UNIVERSITY OF BAHRAIN

College of Science



Postgraduate Programs

Machine Learning Approach for Profiling Petroleum Products FTIR Spectrums

Thesis submitted in partial fulfilment of the requirements for the degree of M.Sc. In Big Data Science and Analytics

Submitted by

Iyad AlAtoom

ID: 20133222

Supervised by

Dr. Riadh Ksantini
Associate Professor
University of Bahrain

Dr. Sawsan Hilal
Assistant Professor
University of Bahrain

Kingdom of Bahrain

March 2023

UNIVERSITY OF BAHRAIN

College of Science



Postgraduate Programs

Machine Learning Approach for Profiling Petroleum Products FTIR Spectrums

Thesis submitted in partial fulfilment of the requirements for the degree of M.Sc. In Big Data Science and Analytics

Submitted by
Iyad AlAtoom
ID: 20133222

Supervised by

Dr. Riadh Ksantini
Associate Professor
University of Bahrain

Dr. Sawsan Hilal
Assistant Professor
University of Bahrain

Kingdom of Bahrain
March 2023

Defense Committee's Approval

University of Bahrain
Deanship of Graduate Studies
And scientific Research

جامعة البحرين
عمادة الدراسات العليا والبحث العلمي

Thesis Discussion Committee Decision

Student Name: Iyad AlAtoom	Student ID: 20133222
Department: Choose an item.	College: **Science**
Program: **M.Sc. in Big Data Science and Analytics**	
Thesis Title: Machine Learning Approach for Profiling Petroleum Products FTIR Spectrums	
Supervisor Name: **Dr. Riadh Ksantini + Dr. Sawsan Hilal**	Student GPA: **3.89**

The Discussion Committee met on 24-May-23 In S41-1002. Upon discussing the student whose details are stated above in public, the discussion committee recommends the following:

The student is awarded a grade Choose an item. in the thesis subject.

☐ Without making any changes to the thesis.

☒ After making the modifications attached with this decision.

Discussion Committee Members Approval:

Name		Signature
1. **Dr. Riadh Ksantini + Dr. Sawsan Hilal**	(Advisor)	SHilal
2. **Dr. Wael Elmedany**	(Internal Examiner)	
3. **Prof. Mohammad Bsoul**	(External Examiner)	MBsoul

Approved at the University Council 6th meeting, 2013
In accordance with Decision No. 736/2013 dated 14/3/2013
Modified by University Council Decision No. 2043/2013 dated 20/11/2013.

Abstract

The refinement of petroleum products of crude oil constitutes a pivotal aspect of modern society, as these products are extensively utilized across numerous industries and sectors. Efficient profiling of petroleum products is of paramount importance, as it plays a crucial role in quality control, environmental protection, and adherence to the ever-evolving market demands.

This study has the potential to substantially influence the oil-based industry through its nuanced profiling of five petroleum products, including Diesel, Kerosene, Gasoline, Fuel, and Lube Base Oil.

For profiling petroleum products through Fourier Transform Infrared (FTIR) data, this study implements a cutting-edge machine learning (ML) methodology, specifically an ensemble of one-class classifiers (OCCs). The OCCs can overcome the imbalanced issue, class overlap, in-sample noise, and outlier identification. Six OCCs were compared, and each consisted of a different base classifier, which are One-class Support Vector Machines (OCSVM), Gaussian Mixture Models (GMM), Isolation Forest (IF), Principal Component Analysis (PCA), Local Outlier Factor (LOF), and K-Nearest Neighbor (KNN). The OCCs were validated against Artificial Neural Networks (ANN).

The accuracy results indicate that the GMM-based ensemble achieved the highest performance across all evaluation metrics, followed by OCSVM-based and LOF-based ensembles with 0.936, IF-based ensemble with 0.908, KNN-based ensemble with 0.899, and PCA-based ensemble with 0.871. Each of the ensembles outperforms the ANN classifier, which struggled with imbalanced data, achieving an accuracy of 0.441. Some of the used methods are sensitive to high dimensional data, therefore the experiments were also done with dimension reduced FTIR data using an

optimized Autoencoder. The results showed a huge drop in the performance of OCSVM-based, a slight drop for GMM-based, PCA-based, and IF-based OCCs, and remained the same for LOF-based and KNN-based.

This study empowers ML for important implications for the oil industry, as it provides versatile, robust, and effective maneuvers for employing it for profiling petroleum products. The results of this study provide valuable insight into the strengths and weaknesses of the different OCCs that may assist practitioners in selecting the most appropriate classifier for their specific problem.

Acknowledgments

I am grateful to Dr. Riadh Ksantini and Dr. Sawsan Hilal for their support and encouragement during this work. Their guidance and feedback were instrumental in shaping my research and refining my writing.

Also, I would like to extend my appreciation to my colleagues who assisted me from the outset of this study and I am immensely thankful to my family for their unwavering endurance and tolerance during the program.

Table of Contents

Defense Committee's Approval ..... A

Abstract ..... B

Acknowledgments ..... D

Table of Contents ..... E

List of Tables ..... G

List of Figures ..... H

List of Abbreviations I

Chapter One Introduction to the Study 2

1.1. Background and Research Problem 2

1.2. Research Objectives 5

1.3. Research Significance 6

1.4. Thesis Structure 7

Chapter Two Literature Review 9

2.1. Introduction to One-Class Classification 9

2.2. One-Class Classification Ensembles 12

2.3. ML with Spectroscopic Data 14

2.4. Summary 18

Chapter Three Methods and Procedures 20

3.1. OCC Approaches 20

3.1.1. Gaussian Mixture Model 20

3.1.2. One-Class Support Vector Machines 21

3.1.3. Isolation Forest 23

3.1.4. Local Outlier Factor.....24
3.1.5. K Nearest Neighbors 25
3.1.6. Principal Component Analysis 25
3.2. Ensemble Technique.....26
3.3. Multiclass Classifier and Dimension Reduction 28
3.4. Evaluation Metrics.....30
Chapter Four Data, Results and Discussion.....33
4.1. Study Data 33
4.1.1. Data Collection and Prerecession.....33
4.2. Experiment Protocol 38
4.3. Results.....40
4.3.1. Training Phase OCCs 40
4.3.2. Computation Time Assessment of OCCs.....47
4.3.3. Testing Phase OCCs 48
4.3.4. ANN Findings.....60
4.4. Discussion.....61
Chapter Five Conclusion and Future Work.....65
5.1. Conclusion 65
5.2. Future Work.....67
References.....68
Appendices.....
الملخص .....

List of Tables

Table (1) FTIR spectra composition regions 4
Table (2) Products data distribution 34
Table (3) Multiclass classification products encoding 40
Table (4) OCCs Training findings with full FTIR 41
Table (5) Average OCC training results using full FTIR 42
Table (6) OCCs Training results using dimensionally reduced FTIR 44
Table (7) Average OCC training results in dimensional reduced FTIR 45
Table (8) methods' computation time 47
Table (9) Testing phase samples 48
Table (10) Full FTIR OCCs ensembles findings 49
Table (11) Full FTIR average OCCs ensembles findings 50
Table (12) Original FTIR OCCs ensembles Mahalanobis distance findings 51
Table (13) Dimensionally reduced FTIR OCCs ensembles findings 55
Table (14) Dimensionally reduced FTIR OCCs ensembles average findings 56
Table (15) Dimensionally reduced FTIR OCCs ensembles Mahalanobis findings 57
Table (16) ANN training findings 60
Table (17) ANN testing findings 60

List of Figures

Fig. 1 OCSVM target class separation (Maglaras & Jiang, 2015)..22

Fig. 2 Isolation process (Esra Kın, n.d.) 24

Fig. 3 OCCs ensemble approach stages.....26

Fig. 4 PCA first and second principal components representations before and after S-G.....36

Fig. 5 Products spectra after preprocessing visualization 37

Fig. 6 Testing and training phase data splitting.....39

Fig. 7 Average OCC training results using full FTIR 42

Fig. 8 Average OCC training results in dimensional reduced FTIR 45

Fig. 9 Full FTIR OCCs ensembles average performance before Mahalanobis.....52

Fig. 10 Full FTIR OCCs ensembles average performance after Mahalanobis.....52

Fig. 11 Reduced FTIR OCCs ensembles average performance before Mahalanobis.....58

Fig. 12 Reduced FTIR OCCs ensembles average performance after Mahalanobis.....58

List of Abbreviations

GC	Gas Chromatography
LC	Liquid Chromatography
IR	Infrared
FTIR	Fourier Transform Infrared
ML	Machine Learning
OCCs	One-Class Classifiers
AUC-ROC	Area Under the Receiver Operating Characteristic Curve
OCSVM	One-class Support Vector Machines
GMM	Gaussian Mixture Models
IF	Isolation Forest
PCA	Principal Component Analysis
LOF	Local Outlier Factor
KNN	One-class K-Nearest Neighbor
ANN	Artificial Neural Networks
OCC	One-Class Classification
SVM	Support Vector Machines
RBF	Radial Based Function
SVDD	Support Vector Domain Description
KW-KPCA	Kernel Principal Component Analysis Kernel Whitening
AANN	Auto-Associative Neural Network
OVO	One Verses One
OVA	One Verse All
ECOC	Error Correcting Outputs Code
DT	Decision Templates
LIBS	Laser-Induced breakdown spectroscopy
LDA	Linear Discriminate Analysis

SVC	Support Vector Classifier
OPLS-DA	Orthogonal Partial Least Square-Discriminant Analysis
BPNN	Back Propagation Neural Networks
S-G	Savitzky–Golay
LSR	Least Squares Regression
GILS	Genetic Inverse Least Squares
SVR	Support Vector Regression
RSME	Root Mean Squared Error
kNN	k nearest neighbour
DS	Direct Standardization
SIMCA	Soft Independent Modelling of Class Analogy
EM	Expectation-Maximization
PDF	Probability Density Function
MLP	Multilayer Perceptron
ReLU	Rectified Linear Unit
CSV	Comma Separated Values
CV	10-fold Cross-Validation

Chapter One

Introduction to the Study

1.1. Background and Research Problem

1.2. Research Objectives

1.3. Research Significance

1.4. Thesis Structure

Chapter One

Introduction to the Study

1.1. Background and Research Problem

Fundamentally, crude oil is a blend of hydrocarbons and other chemical substances that has been formulated over the years from the remains of dead organisms (diatoms) that existed on the Earth's planet millions of years ago. Crude oil is mined from the underground sediments, and then it is passed through an intense refining process to increase the rate of chemical reactions, and it is undergone different operating conditions to extract a variety of petroleum products, including fuels, lubricants, and petrochemicals, each with its unique applications (Peters et al., 2007).

Practically, profiling petroleum products is a crucial task for the oil-based industry with a continuous operation and escalating demanding international market. Indeed, identifying and labeling these products will help ensure safety within the industrial processes of handling, retaining, and storing. Moreover, profiling petroleum products complies with the local regulations of quality control and the specifications of the demanding markets. Undoubtedly environmental protection is another important aspect where the source of leakages and spills is effortlessly determined (Yang et al., 2017). However, Wauquier (1995) emphasizes that profiling petroleum products is a challenging task due to the common chemical nature of these products. The existing approaches usually require analytical investigations such as looking at the physical properties of the products besides their chemical properties or using spectroscopy, Gas Chromatography (GC) or Liquid Chromatography (LC), rheology, etc..

Ultimately, the use of any existing approaches is dependent on the product type and availability of equipment and expertise. This can be

further complicated by the fact that measuring only one or two physicochemical components may not provide accurate classification, leading to additional analytical efforts. One of the commonly used approaches for identifying petroleum products is spectroscopy, which is a powerful tool for analyzing chemical composition. It is still widely used due to its high sensitivity and specificity, such as the Infrared (IR) spectroscopic. Basically, IR is a type of electromagnetic radiation that lies within a certain wavelength range. It transmits the radiation at each wavelength through a defined amount of sample. The energy is absorbed by chemical bonds within the sample, then it prints out the sample's chemical compositions at each wavelength, called spectra. Typically, the wavelength range of IR is between about 0.78 micrometers (μm) and 1000 micrometers (μm), which corresponds to wavenumbers of about 12800 cm^{-1} and 10 cm^{-1} (Skoog et al., 2017). In particular, Fourier Transform Infrared (FTIR) spectroscopy typically operates in the mid-infrared range, which is between 4000 cm^{-1} and 400 cm^{-1}, and it can be applied to different sample forms such as liquids, solids, and gases. It analyzes and detects a variety of functional groups, including hydrocarbons. FTIR spectra can be broken down into four regions, each of which exhibits specific chemical representations. These regions, starting from 4000 cm^{-1} to 400 cm^{-1}, include single, triple, and double bonds stretch, as well as the fingerprint region as demonstrated in **Table 1** (Thomas et al., 2017).

Table (1) FTIR spectra composition regions

Region	Wavenumber Range (cm^{-1})	Bonds and Functional Groups
Single Bond Region	4000-3000	O-H stretching (alcohols, acids)
		N-H stretching (amines, amides)
		C-H stretching (alkanes)
Triple Bond Region	3000-2000	C≡C stretching (alkynes)
		N≡N stretching (nitriles)
Double Bond Region	2000-1500	C=C stretching (aromatics)
		C=O stretching (aldehydes, ketones)
		P=O stretching (phosphates)
Fingerprint Region	1500-400	C-H bending (all compounds)
		C-O stretching (alcohols, esters)
		C-N stretching (amines, amides)
		O-P-O stretching (phosphates)
		Fingerprint region (unique patterns)

The complexity of the output data, the instrument sensitivity, and the background environment hinder the accuracy of FTIR and end up with invalid assistance to an analyst in differentiating between petroleum products. The aforementioned factors trigger the need to improve the current approach within the oil industry to enhance the time and labor consumption, reduce personal bias, and simplify interpretation to achieve the highest possible accuracy. This is the reason behind the recent interest in the advances of artificial intelligence and particularly machine learning (ML) as tools to solve oil industry problems. Specifically, there has been a rising interest in coupling ML with IR spectroscopy for a variety of applications such as adulteration detection, identifying oil origin, chemicals concentrations, products, and chemicals classification (Brown et al., 1975).

It is worth mentioning that the utilization of ML in spectroscopy applications is still limited within the oil industry field, and many of the well-documented issues of the relevant data, such as handling imbalanced data, a small number of samples, and reducing noise in samples are missed

or poorly handled. Moreover, the implemented ML models are restricted to standard classifiers. Therefore, to account for all these deficiencies in the related literature, the current study focuses on classifying five petroleum products, these products are Gasoline, Diesel, Kerosene, Fuel Oil and Base Oil where the products' spectroscopic analysis data has been done and collected by the researcher from a well-recognized oil company and validated by the field experts. The analysis was based on FTIR spectra of the products with the use of an ensemble of multiple One Class Classifiers (OCCs) such that training each classifier on each class provides an efficient technique for handling the imbalance issue of the data instead of using the commonly used technique of implementing artificial data augmentation.

1.2. Research Objectives

The following are the objectives of the current research,

1. To impose ML techniques to profile petroleum products based on their FTIR spectral signatures.
2. To compare different base methods for the OCCs ensemble by evaluating their classification performance across the imbalanced dataset using a range of objective evaluation metrics, such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC).
3. To investigate the impact of the dimension reduction technique, Autoencoder, on the performance of OCCs ensembles in profiling FTIR spectra of petroleum products.
4. To assess the validity of the OCCs ensemble in overcoming imbalanced FTIR data by comparing it against a multiclass classifier.

1.3. Research Significance

The study contributes to the oil industry by assembling multiple OCCs for profiling petroleum products based on their FTIR spectra. More specifically, a comparison between six OCCs ensembles with a variety of function-based categories has been conducted. The ensembles were based on One-class Support Vector Machines (OCSVM), Gaussian Mixture Models (GMM), Isolation Forest (IF), Principal Component Analysis (PCA), Local Outlier Factor (LOF), and K-Nearest Neighbour (KNN). The study obtains the effectiveness of the used OCCs ensemble procedure to mitigate the imbalanced data issue, and it adds more robustness to class overlap. Furthermore, the study procedure incorporates distance approaches to ensure generalization and prevent an equal probability of sample class. Accuracy, F1-Score, AUC-ROC, precision, recall, and model's consumed training time, are the performance evaluation metrics that have been used in this study experiments. To further validate the utility of the OCCs ensemble method, a comparison was conducted against Artificial Neural Network (ANN) multiclass classifier. Overall, this work provides a comprehensive approach to profiling petroleum products using OCCs ensembles with FTIR spectroscopy, which can potentially improve the accuracy and efficiency of the profiling process and contribute to the oil industry. Unlike other studies that use multivariate techniques or standard classifiers that rarely address the imbalance issue, this study takes an innovative approach to profiling petroleum products by utilizing OCCs ensembles with FTIR spectroscopy. This approach addresses common challenges in the field, such as imbalanced data, noise, and sample overlap, which can improve the accuracy and efficiency of the profiling process

1.4. Thesis Structure

The rest of this report is structured as follows. **Chapter Two** reviews the related work. **Chapter Three** describes the implemented methods along with the performance metrics. The main findings of the current study are collected and discussed in **Chapter Four**, while **Chapter Five** concludes the study and highlights some directions for improvement.

Chapter Two

Literature Review

2.1. Introduction to One-Class Classification

2.2. One-Class Classification Ensembles

2.3. ML with Spectroscopic Data

Chapter Two

Literature Review

This chapter presents a comprehensive review of the related work. Specifically, the first section introduces the concept of one-class classification (OCC), followed by the second section, which explores OCCs ensembles. The third section is an overview of the research carried out using spectroscopic data. Finally, the chapter ends with a summary of all the reviewed work.

2.1. Introduction to One-Class Classification

OCC is an unsupervised machine learning approach in most cases, requiring only one class of data to be trained or modeled with. In OCC research, terms such as positive class, target class, inlier, and normal class are used interchangeably, referring to the class of interest, while on the other hand, terms such as negative class, outlier, and anomaly are used to refer to the encounter class, depends on the attacked problem. Wenzhu et al. (2019) emphasize on OCCs are to be classified into three distinct categories: density-based, boundary-based, and reconstruction-based methods. However, the objective of an OCC is strictly to surround the positive class data points together in the most proper form that enables it to distinguish between positive and negative class examples. Therefore, as it only requires one class to work, it becomes effective in cases where noise, outliers, and imbalanced data issue or unknown distribution class within the data is presented. In these cases, most of the typical multiclass classifiers suffer and lack performance. Nevertheless, OCC still can face difficulties as much as multiclass classification methods such as high dimension, small datasets, and data complexity, and hence needs to be handled using the most optimum techniques.

OCC is extensively implemented in detecting irregularities across various domains, including industrial, medical, military, security, food services, and more. OCC has proven effective in detecting frauds, spam, unusual traffic flows, abnormal process conditions, medical diseases, impurities, image classification, unwanted objects, and more.

In 1999 that one of the earliest works on OCC was introduced by Schölkopf et al., where they used the boundary concept of Support Vector Machines (SVM) algorithm to describe the data within a hyperplane to be as one similar class, and whatever is projected outside that hyperplane is considered as an outlier, the approach's named as One-Class Support Vector Machines, OCSVM. The model can handle linear or nonlinear high dimensional data with different kinds of distribution because of the special kernel function property. Although the choice kernel function can be a bit intensive, both Li et al., (2021) and Zheng et al., (2021) in their overview found that Radial Based Function (RBF) to be as the most common and robust choice as a kernel function for OCSVM in complex cases.

The density-based approach, LOF, was proposed by Breunig et al. (2000), where a prototype is considered an outlier if its local density is substantially lower than its neighbors' local density. This functionality based helps in detecting in-class outliers and makes it immune to the high-dimensional curse.

Figueiredo and Jain (2002) presented the GMM as OCC or anomaly detection method, which calculates the probability of the data that fits the target class normal distribution; those with low probability are classified as outliers.

On the other hand, PCA is a well-known unsupervised dimension reduction method, Shyu et al. (2003) demonstrated the effectiveness of PCA in anomaly detection with real high-dimensional datasets. The method catches that maximum variance of the data using principal

components and projects it onto lower-dimensional subspace where a multivariate Gaussian distribution is applied. Then, by using Mahalanobis distance, points that are far from that subspace are considered outliers.

Breunig et al. (2002) presented the OCC approach using KNN. Simply, the concept implies that the anomaly is any point with the highest average distance to its neighbors. Applied by Gyunka and Barda (2020) in Android malware detection, it showed 88.5% overall accuracy.

Binary trees were each constructed by splitting a feature value into two portions, each representing positive and negative classes. This IF method was presented by Liu et al. (2008) and had to be optimized by selecting a fixed splitting point using specific separation criteria between the two portions.

Moreover, Hayashi and Fujita (2022) have provided a comprehensive overview of the performance of OCC models on the aspects of high dimensionality and imbalanced data issues. Their findings indicate that despite the superiority of OCC models over conventional binary classifiers with and without data augmentation techniques, selecting the most suitable OCC model remains a formidable task. Specifically, they compared the performance of GMM, OCSVM, IF, and LOF, and discovered that OCSVM's performance decreases with increasing data dimensions when compared to GMM, IF, and LOF. The effectiveness of OCSVM for petroleum product profiling can be affected by its complexity, as shown in a study by Deng et al. (2018) that compered one-class and binary classifiers in land type identification. The study found that OCSVM failed to produce satisfactory results due to its sensitivity to parameter optimization.

2.2. One-Class Classification Ensembles

Owing to the appearance of multiclass models' low performance and computing complexity in some cases, the interest in implementing OCCs ensemble to boost the multiclass classification performance by easing most of the common issues such as overfitting, imbalanced data, in-class noise, and overlap. OCC allows us to concentrate on each class and its unique characteristics by revolving the model around it. More precisely, the concept relies on training and tuning OCC on one target class and considering other classes to be outliers. Assembling OCCs depends on three factors: classifiers' base, decomposition method of the problem, and final outputs aggregation approach.

In a few studies where the aggregation is dependent on the OCC base model, such as in Ban and Abe (2006) work, the OCCs ensemble implementation was evaluated by experimenting with two algorithms, Support Vector Domain Description (SVDD) and Kernel Principal Component Analysis based Kernel Whitening (KW-KPCA), in solving multiclass problem against pairwise SVM. Pairwise SVM is an ensemble of multiple binary classifiers that have been trained on each pair of classes. Due to the boundary-based of the classifiers, they imposed simple aggregation of the results by computing the distance of the projection to each classifier's hypersphere canter and assigning it to the class with the minimum distance value. However, this approach is applicable when using boundary-based algorithms.

Schaefer (2017) has proposed the use of boundary-based ensembles of one-class classifiers. In this approach, the distances between samples and decision boundaries are transformed into a support function, enabling the fusion of base classifiers based on parameters such as the mean of estimated probabilities.

In their research, Krawczyk and Woźniak (2015) addressed the challenge of imbalanced data in the medical domain by employing a hierarchical ensemble of one-class classifiers. The ensemble was designed to classify patients with essential or secondary hypertension, with a mere 5% of cases being attributed to the latter. These 5% were further sub-categorized into five different types based on their underlying causes. To overcome the issue of imbalanced data, Krawczyk and Wozniak compared various multiclass classifiers with and without a data augmentation process. Their experimental findings confirmed the effectiveness of the proposed hierarchical one-class classifier ensemble, which could be a valuable tool for real-world medical decision support systems.

In Hadjadji et al. (2014) study, the ensemble system included three OCC classifiers from different groups, which are KNN, Auto-Associative Neural Network Classifier (AANN) and OCSVM. Furthermore, due to the classifiers' different bases, the researchers approached the ensemble aggregation by using the maximum prediction value of each classifier. Despite the apparent extensive effort in the training phase, the performance of their system varied when experimenting with different datasets, which indicates a lack of model generalization.

Nevertheless, different OCCs decomposition methods such as One Verses One (OVO), One Verse All (OVA), Error Correcting Outputs Code (ECOC), and Decision templates (DT) were experimented with by Krawczyk et al. in their work published in 2015 and 2018. Their work indicated that the most common and robust one is OVA, of which OVA decomposition includes classifiers as much as the number of classes in the problem. Then assigning a class to any new projection depends on the aggregation methods of the ensemble outputs, which is considered the central challenge of the OCCs ensemble.

2.3. ML with Spectroscopic Data

From the ML view, the spectroscopic is considered a low-data domain with high-dimensional and continuous-value features. Previous works of ML with spectroscopic data were found in many fields for a variety of tasks, quantitative and qualitative, such as classifying, measuring concentrations, clustering unknown items, and more.

A study by Gyftokostas et al., (2021) focused on differentiating olive oil samples from different regions and mixtures of them. Laser-Induced breakdown spectroscopy (LIBS) and absorption spectroscopy data both were experimented with by the researchers using linear discriminate analysis (LDA) and support vector classifier (SVC). Depending on the discrepancy of the samples' geographical features, LDA and SVC succeeded in classifying the samples into four groups, three were the origins, and the fourth indicated the mixture. The study showed that the dimension reduction of both spectroscopic data types would not affect the performance of the classifiers.

In a different application, Calle et al. (2022) conducted a study concerning fruit juice adulteration, which is a vast practice that can either improve the quality and taste or reduce the cost for more profitable gain. In this regard, the researchers experimented with LDA, Random Forest (RF), and SVC on 76 of 3 juice types and 108 adulterated samples. The results showed adequate classification of juices and adulteration detection based on their FTIR data. Similarly, Sen et al. (2021) considered milk adulteration practices by using orthogonal partial least square-discriminant analysis (OPLS-DA) and succeed to discriminate in between FTIR of buffalo, goat, and cow milk, in addition to differentiating pure and mixture milk.

On the other hand, Zhu and Xu (2020) were persistent to back propagation neural networks (BPNN) in classifying FTIR walnut samples, including ten different types from many origins. The study focused on comparing five classifiers using different data subsets. BPNN showed to have the best performance.

Houston et al. (2020) analyzed 230 Raman Spectra of solvent and non-solvent chemical mixtures, including unexpected outliers' data in the test set to evaluate the performance of OCCs on the original data, which are driven instances of a negative class from different distribution forms. They observed that the presence of outliers and the effective detection of these outliers by OCCs such as Autoencoder and denoising Autoencoder led to the development of a two-step classification approach, where OCC models allow inliers to pass through to be classified by binary classifiers later. The researchers noted that binary classifiers can outperform OCCs when trained with efficiently labelled data, regardless of imbalanced issues, but their performance drops when outliers are present in the test set. Although the researchers noted that OCC is more compelling than binary or multiclass classification due to the supported decision-making from only one side, still needed as limited labelled training instances and acquisition challenges are found. Furthermore, the study did not address the issue of imbalanced data.

In the medical sector Korb et al. (2020), prior to binary classifying of FTIR serum data, applied efficient preprocessing for correcting and smoothing the FTIR spectra curves such as Savitzky–Golay (S-G) filter, multiplicative scatter correction, etc., the researchers were able to prove the feasibility of FTIR paired with deep learning strategies to investigate allergic mice or human from healthy ones.

In the microbiology field, FTIR was used to measure sucrose concentrations and identify with or without bacteria samples. At

classifying with/without bacteria, using ANN and SVC, a performance drop was noticed when applying PCA for dimension reduction; other than this, both classifiers performed very well.

A study was conducted by Sakalli (2021) comparing Least Squares Regression (LSR), Genetic Inverse Least Squares (GILS), and a combination of them for discovering pure and blended Gasoline parameters depending on the FTIR spectra. The study results indicated that there was no preprocessing performed, such as smoothing, noise eliminating, etc., except for the extraction of small spectral regions, and a general unsatisfactory with the relevant results could be due to the narrowed amount of data.

Another experiment of LSR and GILS was done by Şentürk (2020) for determining the hydrocarbon composition of a petroleum product, Naphtha, by analyzing FTIR data samples. The researcher excluded certain spectral regions from FTIR data where readings are nonsense, above the limit, and noisy, which can cause a nonlinearity issue that conflicts with Beer's law regarding the chemical absorbance and concentration linearity relationship. PLS showed a more accurate prediction at most of the composition parameters.

Yu et al. (2022) considered the nonlinearity within the spectroscopic data and applies a novel model which has been compared against PLS and support vector regression (SVR) for defining petroleum properties. The model was an improved version of Genetic Programming combined with PLS. It was experimented with three datasets, each representing a specific petroleum product: Diesel, Gasoline, and Kerosene. Based on R2 and Root Mean Squared Error (RSME) reading, the model outperformed PLS on all datasets and two out of three when compared to SVR.

A pattern recognition study enabling crude oil quality evaluation (Lovatti et al., 2019) presented 66.7% compatibility when using PCA and

k nearest neighbour (kNN) when profiling crude oil samples into clusters of similar qualities.

Silva et al. (2015) directed their attention toward differentiating Brazilian Gasoline from foreign ones, specifically those from Venezuela and Peru. They acquired a total of 126 Gasoline samples, which were analyzed using three different FTIR instruments. To align the FTIR data points and reduce misrepresentation, the researchers employed the direct standardization method (DS). Furthermore, in order to identify the Brazilian Gasoline samples, the researchers employed the PLSDA and Soft Independent Modelling of Class Analogy (SIMCA) methods. Prior to the modelling process, the spectra were smoothed using the Savitzky-Golay filter, and PCA was used to optimize the filter's parameter, ensuring that any outliers were minimized during the smoothing process. They approached the experiment in two ways. The first way involved training models on data obtained from the same instrument as the majority of the data, while the second involved training on data obtained from all three instruments. It is noteworthy that SIMCA exhibits a unique ability to identify samples that may belong to more than one type or none at all, thus offering valuable assistance in the smuggling process. However, the imbalanced data issue was overlooked in both ways.

Despite some pilot studies illustrating ML applications in the petroleum field, the majority missed addressing important issues such as handling imbalanced data, dealing with sample noise, and poor spectroscopic data preprocessing. These deficiencies trigger the need for a comprehensive study of ML applications in the petroleum field.

2.4. Summary

Upon review of the literature, OCC serves for identifying normal data points form others, as is trained solely on data points belonging to a single category. OCCs offer several advantages, particularly in cases of imbalanced data or when data points belong to a single class. It pays substantial attention to that signal category, unlike conventional binary or multi-class classifiers that are trained on different categories' data points. OCC is mostly used for novelty detection and identifying outliers, although it can be sensitive to outliers within the dataset. An ensemble of OCCs can effectively replace a conventional binary or multi-class classifier by overcoming common issues such as imbalanced data, noise, and class overlap. However, extra attention must be taken concerning the fusion method and the aggregation of ensemble predictions. The review conducted for the related work highlights the limitation in the applications of OCC in the context of spectroscopic data, besides that imbalanced data has acquired minimal attention in such applications.

Chapter Three

Methods and Procedures

3.1. OCCs Approaches

3.2. Ensembling Technique

3.3. Dimension Reduction Technique

3.4. Evaluation Metrics

Chapter Three

Methods and Procedures

The study adopted OCC for profiling petroleum products and compared a variety of OCC types that are different in their functionality and approach. The first section of this chapter provides an overview of all the OCC methods utilized in this study. The second section describes the fusion of OCCs. The third section introduces using ANN as a multiclass classifier and dimension reduction technique. Lastly, the fourth section is an overview of the used evaluation metrics.

3.1. OCC Approaches

3.1.1. Gaussian Mixture Model

One-Class Gaussian Mixture Model (GMM) is a statistical model used for outlier detection. It assumes that the target class data has a single Gaussian distribution, and the outliers belong to different distributions. In the model's training, the Expectation-Maximization (EM) algorithm is used to estimate the parameters of the Gaussian distribution to asset the decision that is based on the probability density function (PDF) of the Gaussian distribution as in **Equation (1)** (Aggarwal, 2017).

$$f(x) = \frac{1}{\sqrt{(2\pi)^d|\sigma|}} e^{\left(-\frac{1}{2}(x-\mu)^T\sigma^{-1}(x-\mu)\right)} \qquad (1)$$

where d represents data dimension, x input instance, μ mean vector, and σ covariance matrix.

Starting from the expectation step, once the initial values of the model's parameters, namely the mean (μ) and covariance (σ). are set, the

model proceeds to calculate the data responsibilities (r_i), which is a value between 0 and 1, as shown in **Equation (2)**.

$$r_i = \frac{f(x)}{f(x) + \alpha} \tag{2}$$

where α is a regularization parameter, usually set to a small positive value, that ensures that the denominator is always positive.

Then after, the maximization step includes updating μ and σ values using the following formula for each parameter, as shown in **Equation (3)** and **Equation (4)**.

$$\mu = \frac{\sum_{i=1}^{c} r_i x_i}{\sum_{i=1}^{c} r_i} \tag{3}$$

$$\sigma = \frac{\sum_{i=1}^{c} r_i \, (x_i - \mu)(x_i - \mu)^T}{\sum_{i=1}^{c} r_i} \tag{4}$$

The algorithm iterates until the convergence when the change in the model parameters between the iterations is small enough, indicating that the algorithm has found a stable and optimal solution. A perfect model is one where all training data responsibilities are high as they belong to the same Gaussian distribution of target class data (Patel & Kushwaha, 2020). For further details, refer to Deisenroth et al. (2020).

3.1.2. One-Class Support Vector Machines

One-Class Support Vector Machines (OCSVM) is an extended logic of the binary classifier SVM in finding the most informative data points (support vectors) that lie closest to the decision boundary. These support vectors represent parallel hyperplanes that each enclose the class data. In OCSVM, the model assumes all input data points to belong to a signal class and returns +1 for that class (target class) and -1 to elsewhere (outlier), as illustrated in **Figure 1**.

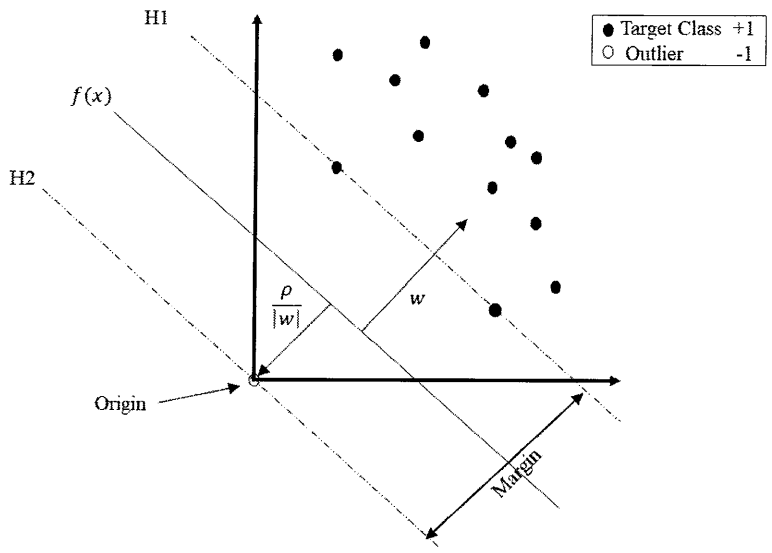


Fig. 1 OCSVM target class separation (Maglaras & Jiang, 2015)

According to Schölkopf et al. (2001), the OCSVM model considers the absence of the second class and assigns the space origin to be its hyperplane. Moreover, the model maps the target data points into a higher dimensional feature space using a kernel function and maximizes the margin between the hyperplane and the origin. The optimization problem described by **Equation (5)** involves finding optimal values of the model's parameters that tend to yield minimal inclusion of outliers.

$$min\left(\frac{1}{2}\|w\|^2 + \frac{1}{c\,nu}\sum_{i=1}^{c}\xi_i - \rho\right)$$
$$subjects\ to \qquad (5)$$
$$w^T\phi(x_i) \geq \rho - \xi_i$$
$$\xi_i \geq 0, i = 1,2,\dots c$$

where w is the hyperplane vector, c is the number of target class samples, nu is the upper bound of outlier fraction as a tradeoff between maximizing the margin and allowing for outliers, ξ is the slack variable, ρ represents the threshold, xi represents the feature vector, and ϕ is the feature map.

Proposing to solve the problem through the Lagrange multiplier and using the kernel function, the decision function will be determined according to **Equation (6)**.

$$f(x) = sign\left(\sum_{i=1}^{c} \alpha_i k(\alpha_i, \alpha_j) - \rho\right) \quad (6)$$

As it was found, the RBF kernel is the most commonly used. Where the RBF kernel in OCSVM is used to measure the Euclidean distance between a test point and the target data points in the training set, and if the distance is larger than a predefined threshold, the test point is classified as an outlier (Maglaras & Jiang, 2015).

3.1.3. Isolation Forest

Isolation Forest (IF) is a nonparametric, fast, and simple algorithm that sets a random value at each feature which uniformly portions the data into two subsets, and the portioning continues until the instances are isolated in a tree. The number of trees is a hyperparameter that is tuned during the training process, and each tree includes random subsets of features to prevent overfitting to any specific feature making the algorithm excels in a case of high dimensional data. The anomaly score is estimated as the average path length of an instance, which is the number of edges an instance traverse to reach a particular tree's terminal node. Instances with a shorter average path length are considered more likely to be outliers, as they require fewer splits to be isolated. Conversely, instances with a longer average path length are considered more likely to be normal, as they require more splits to be isolated as shown in **Figure 2**.

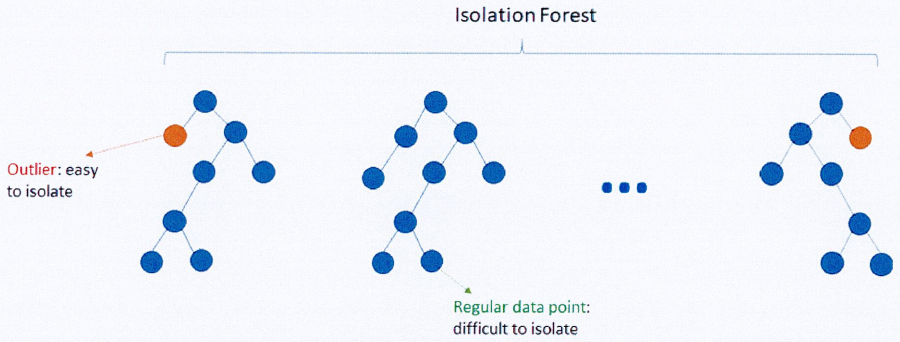


Fig. 2 Isolation process (Esra Kın, n.d.)

The threshold for the isolation score depends on the desired level of contamination (proportion of outliers) in the dataset. The adopted approach for this study was to use a fixed threshold based on the contamination value. For example, suppose the desired level of contamination is 1%. In that case, one can set the threshold as the score corresponding to the 99th percentile of the distribution of anomaly scores for the normal instances. This approach assumes that the distribution of anomaly scores for the normal instances is unknown or cannot be assumed to follow a specific distribution (Liu et al., 2008).

3.1.4. Local Outlier Factor

The Local Outlier Factor (LOF) is a density-based OCC algorithm (Breunig et al., n.d.) that indicates a given instance is an outlier based on a dynamic distance approach involving its k nearest neighbors, where the decision of an object (o) of being an outlier is based on its LOF score, which is imposed by **Equation (7)**.

$$LOF = \sum_{o' \in N_k(o)} Ird_k(o') \times \sum_{o' \in N_k(o)} rd_k(o \to o') \qquad (7)$$

Here Ird is local reachability density, rd is reachability distance, $N_k(o)$ is o neighbors, k is the number of neighbors and o' is an instance that is among $N_k(o)$.

The method is strict to the following stages that are involved in calculating the LOF score of o:

1. Finding rds of the instance, o, to its k nearest neighbors (ko'), which is the maximum value between:
    a. o to the o' distance
    b. distance of $o'$to its k[th] neighbor.
2. Ird is the reciprocal of the average rd between o and $N_k(o')$ as it is shown in **Equation (8)**.

$$Ird_k(o) = \left(\frac{\sum_{o' \in N_k(o)} rd_k(o \to o')}{k} \right)^{-1} \quad (8)$$

The number of neighbors, k, and the LOF score threshold are determined during the model's training process.

3.1.5. K Nearest Neighbors

One-class K Nearest Neighbors (KNN) was initially introduced as an approach to use the traditional KNN as an unsupervised method designed for identifying the positive class instance regarding its distance from the neighbors. It is simple, as it uses global distance instead of local distance, which sets it apart from the LOF method (Ramaswamy et al., 2000).

3.1.6. Principal Component Analysis

Principal Component Analysis (PCA) is an unsupervised learning method that is mostly used as a technique to mitigate the curse of high dimension. PCA as OCC seizes the maximum amount of variance of the target class data points and projects them into a lower-dimensional subspace. The subspace is defined by the eigendecomposition that is involved in computing eigenvalues and eigenvectors of the data's covariance matrix, which are the directions of maximum variance in the

data (Shyu et al., 2003). It defines points as outliers if they lay out the subspace by a significant distance.

3.2. Ensemble Technique

In this study, the adapted approach for profiling petroleum products was through the implementation of an OCCs ensemble as depicted in **Figure 3**. By creating a separate OCC for each product, the approach ensures that each product FTIR data receives equal attention during the OCC construction process, and the final profiling decision is based on the collective response by all the OCCs. The approach is computationally efficient as a multiclass model, where the data are partitioned for each OCC. This aims to improve the performance by mitigating the imbalanced data issue and leveraging the strengths of individual OCC. It contributes to the data representation of each product and its characteristics.

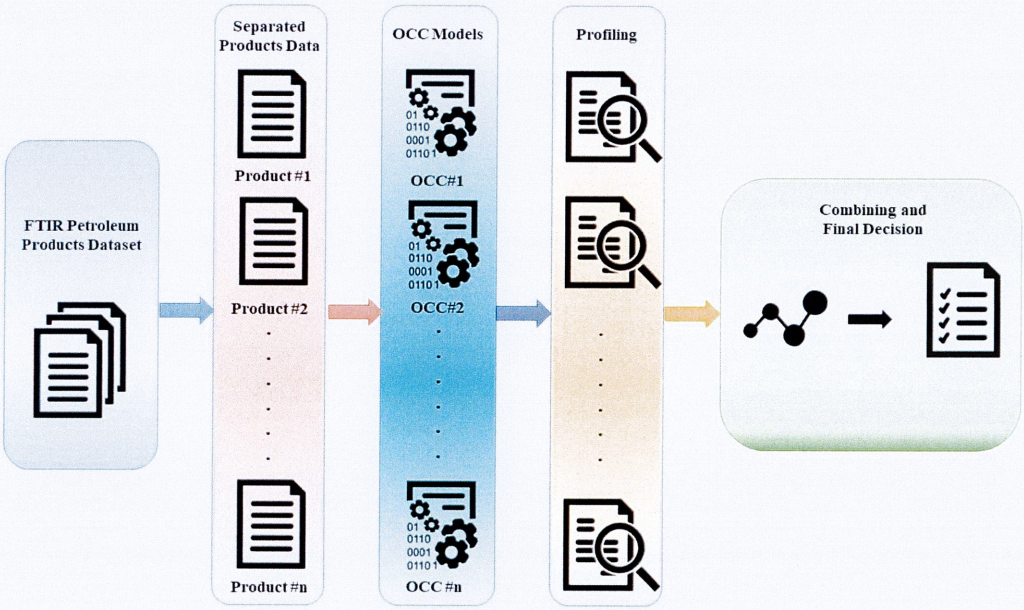


Fig. 3 OCCs ensemble approach stages

In the context of the OCCs ensemble, the final output is typically a combination of the individual OCC predictions. It enables an OCC in identifying an instance to be its target by assisting other OCCs' predictions to it, which adds more confidence to the final output. However, it is not

always straightforward to combine these predictions directly, as certain situations may arise. For instance, there may be cases where two or more OCCs have the same level of certainty of a given instance to be as their target class, ultimately it can be that all OCCs are certain of it being outlier. In such cases, which in this study are called unknowns, it becomes challenging to arrive at a definitive classification for the given instance, and a more nuanced approach may be required to reconcile the differing predictions of the individual OCCs. To address this issue, the Mahalanobis distance has been proposed in the literature as an effective solution in the spectroscopic classification field (Bacci et al., 2001), where spectroscopic data is often highly correlated along various dimensions.

In Mahalanobis (1936) the distance approach was proposed to include the data's covariance structure to calculate the distance between two data points, as presented in **Equation (8)**.

$$MD = \sqrt{(x - y)^T VI^{-1}(x - y)} \tag{8}$$

where MD is the Mahalanobis distance, x and y are the two data points being compared, VI is the covariance matrix of the data, and T denotes the transpose operation.

The covariance matrix VI defined by **Equation (9)** captures the correlations between the different dimensions of the data and allows the Mahalanobis distance to take into account the shape and orientation of the data distribution, which enables a more comprehensive analysis of the underlying relationships between the wavenumbers.

$$VI = \frac{(X - mu)^T (X - mu)}{n} \tag{9}$$

where X is the data matrix with n rows (corresponding to the number of data points), mu is the mean vector of the data, and T denotes the transpose operation.

The unknown instances by the ensemble are profiled to the product with the lowest Mahalanobis distance estimation to the center of its training data points.

3.3. Multiclass Classifier and Dimension Reduction

Artificial Neural Network (ANN) is a mathematical algorithm widely used in the ML field. It is designed to carry out various tasks and works very well when the input variables have complex nonlinear relationships with the event or the class. ANN originated from layers of interconnected processing cells called neurons. The primary layers in an ANN are the input, hidden, and output layers, which imitate the biological structure of the human brain. The neurons are associated with weights that represent the significance of the input information and bias that works as an anchor that moves the model output function horizontally along with the input axis while keeping the shape unaltered to increase the model's flexibility (He et al., 2016). The model repeatedly gains experience during the learning process by comparing the actual. The generated outputs involve adjusting the weights and biases to minimize a loss function between the true and predicted classes. All these weighted information are combined, summed with the bias, and then passed into a transfer function that gives the desired output. The neuron output is obtained by **Equation (10)**.

$$\hat{y} = f\left(b + \sum_{i=1}^{n} x_i \omega_i\right) \qquad (10)$$

where $\hat{y}$ is the predicted output, f is the transfer function, b is the bias, n is the number of input variables, x_i is input variable and ω_i is the weight.

ANN model types correspond to their architecture, i.e., the network's structure, transfer function, learning algorithm, and its specific task. This study uses the multiclass classifier, multilayer perceptron model (MLP) with a backpropagation learning approach to validate against the OCCs ensemble. Additionally, it utilizes the unsupervised ANN model, Autoencoder, for the dimension reduction process.

In MLP, the rectified linear unit (ReLU) transfer function is used at the input and hidden layers, passing the positive inputs, and turning the negatives into zeros. SoftMax is at the output layer that produces a probability distribution over the possible classes or categories. Each layer may consist of one or more neurons.

The gradient descent method minimizes the cross-entropy loss difference between the actual and generated output in backpropagation. This approach substantially depends on two significant extensions, learning rate and momentum (m), to update the model's weights after every epoch. The learning rate represents the proportion in the weight changes to know how far we will step in the weight change. The momentum represents a model's memory from the previous formulation of it to prevent it from the local minimum trap. The epoch is where the model goes through all the data instances. However, the approach in developing the model will be by adding hidden nodes, increasing the momentum, and changing the initial connections' weights to prevent the gradient descent of giving a local minimum instead of a global minimum. Moreover, this approach aims to motivate the model to achieve a viable solution for unseen input data in the future, emphasizing the importance of its generalization capabilities.

Autoencoder is a neural network architecture consisting of encoding and decoding layers. It aims to learn a compressed input data representation

into a lower-dimensional space, the encoder part. In order to maintain the minimal loss of information, the decoder originates the encoder output to the original dimension and compares it with the input. According to Goodfellow et al. (2016), further details about Autoencoders can be found in Chapter 14 of their book, "Deep Learning".

3.4. Evaluation Metrics

In this study, various metrics have been used to evaluate each individual OCC within the ensemble to show a comprehensive assessment of the ability to tackle the imbalanced data issue and profiling FTIR petroleum products. These metrics are accuracy, precision, recall, and F1-score. Accuracy is the ratio of the correct classifications among test examples, which shows a classifier's general performance. Nevertheless, each OCC focuses on classifying the positive class individually or the target class. In this regard, recall, precision, and F1-score measurements are able to provide valid assistance. The recall is the ratio of correct positives to all true positives. Precision, on the other hand, is the ratio of the correct to the classified positives. F1-score is an average rate of both (Swersky et al., 2016).

The following equations, which have been extracted from the confusion matrix, describe the calculation of the evaluation metrics.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{12}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{13}$$

$$\text{F1 Score} = \frac{2x(\text{Precision x Recall})}{\text{Precision} + \text{Recall}} \tag{14}$$

where TP (true positive) is the number of correctly predicted positive samples, FN (false negative) is the number of incorrectly predicted negative samples, FP (false positive) is the number of incorrectly predicted positive samples and TN (true negative) is the number of correctly predicted negative samples.

However, to extend the evaluation into the ensemble abilities on the imbalanced issue, generalization and the comparison on the optimum classifiers' base, AUC-ROC has been utilized, where it is a measure of positive correct and incorrect classification ratios at different classification thresholds. Along with this, time was also employed to assess the computational effort of each OCC type. The evaluation figures for both training and testing are normalized between 0 and 1. A value of 1 indicates perfect performance, while a value of 0 indicates no performance. Intermediate values reflect the degree of performance achieved by the OCC or OCCs ensemble for a given metric and class.

Chapter Four

Data, Results and Discussion

4.1. Study Data

4.2. Experiment Protocol

4.3. Results

4.4. Discussion

Chapter Four

Data, Results and Discussion

In this chapter, the results of the current study on profiling FTIR petroleum products are organized into four sections. The first section is about data collection and preprocessing methods used to obtain the data for the study. This section details the data source, data cleaning and transformation techniques, and other preprocessing steps. The second section describes the experimental protocol used to train and evaluate the performance of different OCC methods in achieving the research objectives and provides information on the experimental design. In the third section, the study's results, which are equipped with graphs and tables, help explain the results and draw conclusions. Finally, in the discussion section, a comprehensive interpretation of the results is provided, accompanied by a discussion of their implications.

4.1. Study Data

4.1.1. Data Collection and Prerecession

In adherence to the ASTM D7575 method for spectroscopy analysis and sample preparation, 545 samples of five diverse petroleum products, Gasoline, Base Oil, Kerosene, Diesel, and Fuel Oil, including their various fractions, were gathered from a well-recognized local organization. The samples were analyzed at an ambient temperature range of 21 to 24 degrees and a resolution of 4cm-1 using Shimadzu Scientific Instrument IRAffinity-1S (ASTM, 2015). A field expert validated the spectroscopy's process and output spectra, expressing the transmittance (%) of electromagnetic radiation over the wavenumber (cm^{-1}) of the FTIR region, which is the rate of infrared light that has been passed through the sample.

The FITR outputs are stored individually in txt form, converted to Comma Separated Values (CSV), and combined into a single CSV file using Python with Pandas (McKinney, 2010).

During the data exploratory, there have been some indications that require extra effort prior to ML modeling, which are discussed in the following subsections.

4.1.1.1. Imbalanced data

Due to the petroleum industry's availability and strict retention process, the data found an unequal distribution of classes, as it is shown in **Table (2)**. This kind of distribution is an eminent issue in any multiclass problem where the model would be more biased toward majorities over others, and its consequences are reflected in the general performance of the model. It motivated the study to propose the OCCs ensemble framework that is built to tackle the imbalanced class issue while preserving the originality of the data.

Table (2) Products data distribution

Product	Sample Count
Gasoline	32
Kerosene	150
Diesel	233
Base Oil	97
Fuel Oil	33
Total	545

4.1.1.2. Missing Values and Noise

It was found that 11 instances were null at 104 (5.56%) wavenumbers. These nulls were filled by the calculated average of each wavenumber at that corresponding instance class.

The FTIR data, though informative, could be inherently noisy. A smoothing approach has been applied to the FTIR spectra to alleviate the fake peaks and noise bumps. S-G filter is known for its effectiveness in

smoothing spectroscopic data (Press & Teukolsky, 1990). Optimizing the weighting factors of S-G, such as window length and polynomial degree, will result in sharper spectra. The window length defines the number of neighboring data points (N), which is used in the discrete calculating of the center of N data points to adopt the polynomial fit. Typically, the window length is an odd number, and the polynomial order is second or third (Gallagher, 2020). Upon conducting the analysis, it was ascertained that the ideal window length to achieve minimal deviation and maximum separability of the products' data points is within the range of 5 to 13. As demonstrated in **Figure 4**, which comprises four distinct representations of the first and second principal components of the data points, both before and after the application of an S-G filter. Each representation shows a different window length used in the S-G filter, (a) shows the data points without the implementation of an S-G filter and the remaining representations feature an S-G filter with varying window lengths, 5 (b), 7 (c), and 9 (d). In addition to this analysis, further investigation was undertaken to identify the window length that minimized the occurrence of transmittance values exceeding 100. A window length of 7 emerged as the most suitable choice, ensuring methodological consistency, and minimizing values exceeding 100. Thus, it was considered the most ideal for subsequent analyses.

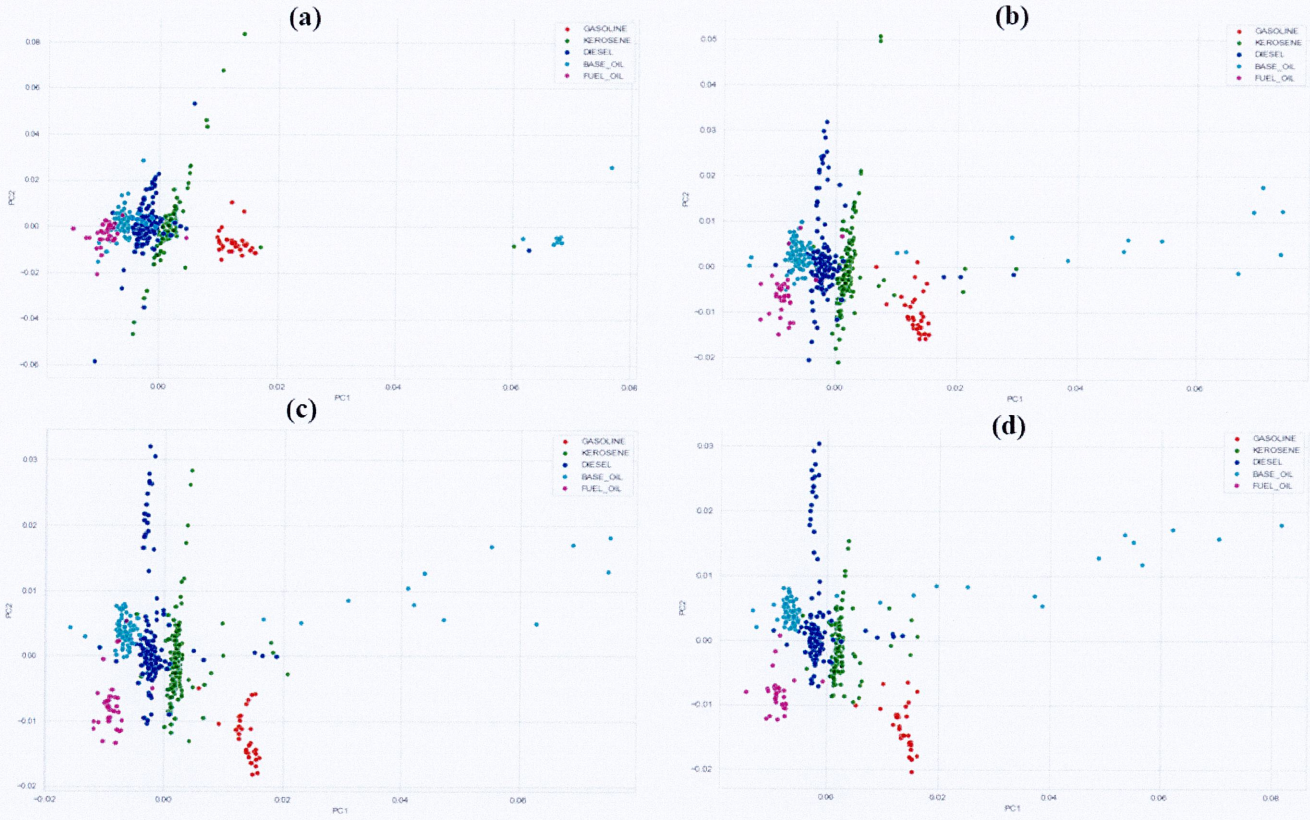


Fig. 4 PCA first and second principal components representations before and after S-G

Furthermore, the transmittance values obtained from the FTIR spectra have been constrained to a maximum of 100, ensuring that the data is consistent and within the expected range. **Figure 5** shows spectra examples of each product after the preprocessing stage.

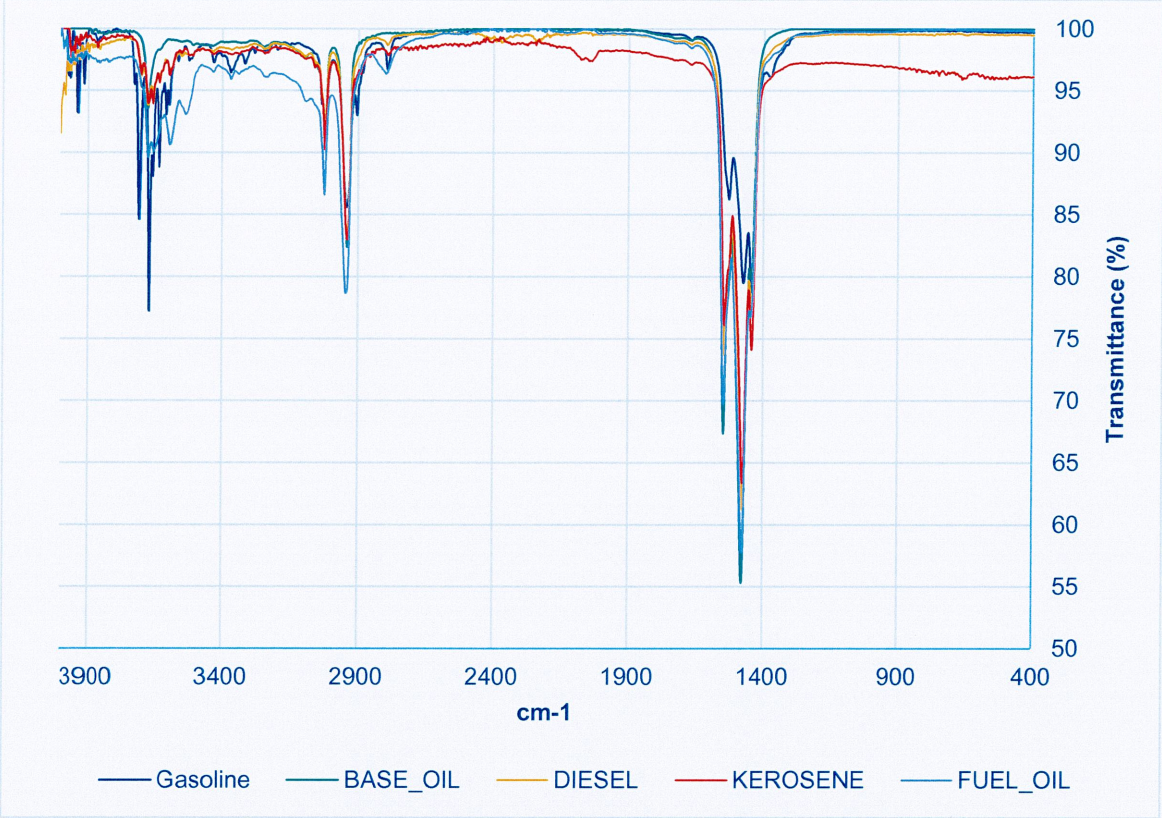


Fig. 5 Products spectra after preprocessing visualization

4.1.1.3. Scaling and High Dimension

Before the modeling part, the data have been normalized to conform with the algorithms' subsequent stages of the process. Normalization is scaling into a range between 0 and 1, where ensuring the consistent distribution of values across all wavenumbers is necessary.

The spectra express a pattern that forms while the wavenumber of the electromagnetic radiation changes. As can be observed, the spectra data are high dimensional, and they have 1868 wavenumbers between 4000 and 400 cm^{-1}, which each wavenumber could be considered as a feature. However, In the context of profiling petroleum products, their spectra pattern places significant emphasis on the inclusion of all data points, where reducing the data may alter the performance of specific models. Nevertheless, a comparative experiment with and without dimension reduction has been included. A standard method for such a task is passing the data through the latent space of the Autoencoder that compresses them and exits them with

a lower dimension. It is worth mentioning that the dimension reduction has not been made to ANN as it is imposed to have experimented to validate the adequate of OCCs ensemble with FTIR original data. Moreover, it is powerful by its nature against the dimensional data problem.

4.2. Experiment Protocol

The study meticulously followed a systematic protocol in the OCCs ensemble's training and testing phases. Wherein each product is trained with an OCC optimized for its unique characteristics and ability to recognize its assigned product. In this study, 5 products are at focus, resulting in the use of 5 OCCs. During the training phase, the accuracy and other metrics are evaluated for each OCC. This allows the study to gain insight into which OCCs are performing well and which ones require further optimization. In the testing phase, where the objective is to measure the performance of the OCCs ensemble in identifying new, unknown instances. As the OCCs ensemble is to aggregate the output of all OCCs and selects the maximum certainty of an OCC in recognizing an instance as its target class.

The data splitting process in this study followed a stringent 80/20 data split ratio, with 80% of the data allocated for training and 20% for testing. The split ratio considered the imbalanced class distribution and aimed to include sufficient one-class data points for each OCC within an ensemble. Additionally, the choice of the 80/20 split ratio upheld reproducibility and consistency in line with industry standards and prior studies. The OCCs data splitting approach of this study into both phases, training, and testing, is shown in **Figure 6**. During the classifiers' optimization stage, a 10-fold cross-validation (CV) technique was employed during the training phase to prevent overfitting and maintain the generalization ability. The 80% of

training data was further split into 75% for training and 25% for validation. Each classifier was trained exclusively on its target class data within the 75% split, while the remaining samples from the other classes were combined with the 25% validation set. In the testing phase, each OCC was meticulously modelled with its target class data from the 80% split, incorporating the tuned hyperparameter values based on the highest median AUC-ROC of the CV process. Finally, the ensembles were tested by utilizing the reserved 20% of the data.

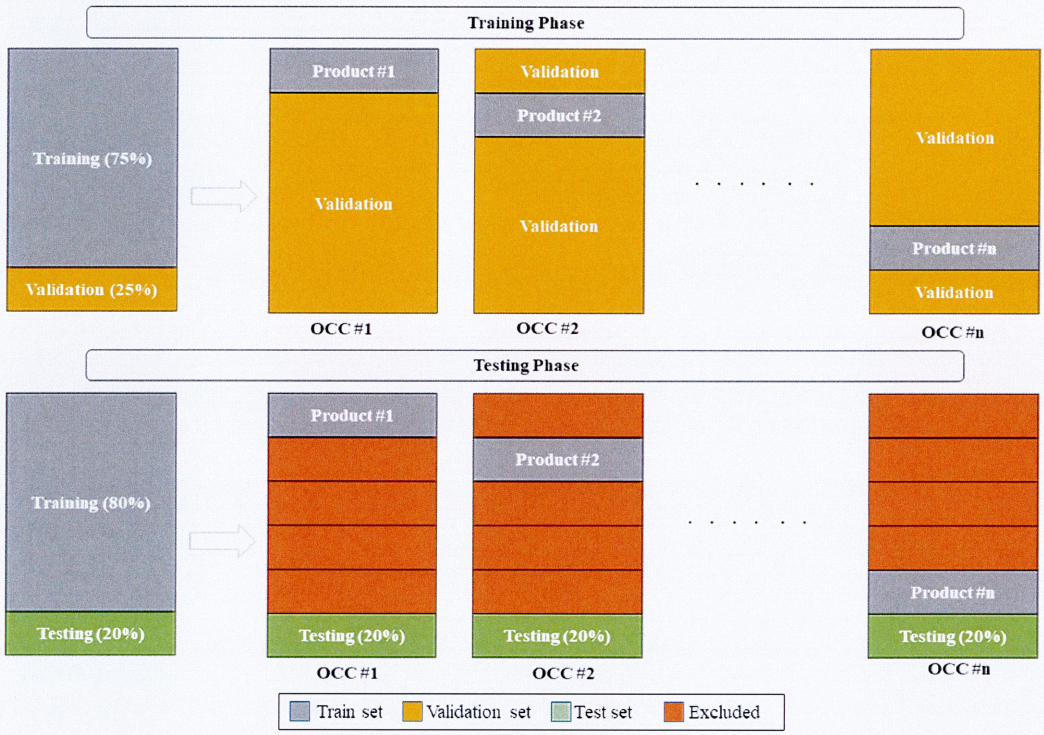


Fig. 6 Testing and training phase data splitting

The PyOD, Python library, provided a comprehensive set of tools for developing and fine-tuning OCCs (Zhao et al., 2019). It is worth mentioning that the data used in this study were encoded as 1 for the target class and 0 for the outlier class, as per the PyOD library's requirements. Additionally, both the evaluation metrics and the Mahalanobis distance incorporated into the OCCs ensemble were computed by the Python scikit-learning library (Pedregosa et al., 2011). Conversely, in the realm of multiclass classification for profiling petroleum products, the Keras library

was employed to implement an ANN classifier with an 80/20 splitting process for optimization and testing (Gulli & Pal, 2017). To this end, each product was encoded with an integer number between 0 and 4, which was specified based on each's physical density. **Table (3)** showcases the encoding scheme for each of the products.

Table (3) Multiclass classification products encoding

Product	Code
Gasoline	0
KEROSENE	1
DIESEL	2
BASE OIL	3
FUEL OIL	4

4.3. Results

This section presents the OCCs ensembles' findings, including Mahalanobis' distance performance on assigning the unknowns to their product class. During the training phase, each OCC is trained on its target class, and the findings represent the best CV performance of each OCC in recognizing the target class from the others. In the testing phase, the OCCs ensemble findings are presented for each OCC in profiling each product, along with the estimated metrics average and overall accuracy before and after profiling the unknowns using the Mahalanobis distance. Additionally, this section presents the findings from training and testing the ANN.

4.3.1. Training Phase OCCs

Table (4) presents the best CV training results obtained by each OCC for each product across the implemented methods with full FTIR data.

Table (4) OCCs Training findings with full FTIR

Method	OCC	Accuracy	F1_Socre	Precision	Recall	AUC-ROC
GMM	Gasoline	0.995	0.997	0.995	1	0.868
	Kerosene	0.974	0.986	0.972	1	0.864
	Diesel	0.965	0.978	0.960	0.997	0.921
	Base Oil	0.982	0.991	0.987	0.994	0.905
	Fuel Oil	0.999	1	0.999	1	0.971
Method	**OCC**	**Accuracy**	**F1_Socre**	**Precision**	**Recall**	**AUC-ROC**
OCSVM	Gasoline	0.999	0.999	0.999	1	0.971
	Kerosene	0.961	0.979	0.978	0.979	0.884
	Diesel	0.715	0.780	0.958	0.669	0.778
	Base Oil	0.980	0.989	0.989	0.989	0.918
	Fuel Oil	0.999	1	0.999	1	0.979
Method	**OCC**	**Accuracy**	**F1_Socre**	**Precision**	**Recall**	**AUC-ROC**
IF	Gasoline	0.995	0.997	0.999	0.996	0.968
	Kerosene	0.904	0.946	0.958	0.936	0.768
	Diesel	0.925	0.954	0.932	0.976	0.855
	Base Oil	0.956	0.976	0.980	0.972	0.853
	Fuel Oil	0.994	0.997	0.994	1	0.814
Method	**OCC**	**Accuracy**	**F1_Socre**	**Precision**	**Recall**	**AUC-ROC**
PCA	Gasoline	0.996	0.998	1	0.997	0.986
	Kerosene	0.968	0.983	0.971	0.995	0.854
	Diesel	0.931	0.958	0.929	0.988	0.853
	Base Oil	0.974	0.986	0.982	0.990	0.873
	Fuel Oil	0.994	0.997	0.993	1	0.807
Method	**OCC**	**Accuracy**	**F1_Socre**	**Precision**	**Recall**	**AUC-ROC**
LOF	Gasoline	1	1	1	1	0.988
	Kerosene	0.954	0.974	0.992	0.957	0.941
	Diesel	0.977	0.986	0.980	0.992	0.957
	Base Oil	0.976	0.987	0.991	0.982	0.932
	Fuel Oil	0.955	0.977	0.998	0.956	0.935
Method	**OCC**	**Accuracy**	**F1_Socre**	**Precision**	**Recall**	**AUC-ROC**
KNN	Gasoline	1	1	1	1	0.994
	Kerosene	0.983	0.991	0.981	1	0.908
	Diesel	0.957	0.973	0.964	0.982	0.923
	Base Oil	0.970	0.984	0.987	0.981	0.904
	Fuel Oil	0.998	0.999	0.999	0.999	0.978

Table (5) and **Figure 9** demonstrate the average training results of the methods across all OCCs. The results demonstrate a highly successful

optimization of each product OCC across all evaluation metrics, with average results exceeding 0.85.

Table (5) Average OCC training results using full FTIR

OCC Method	Accuracy	F1_Socre	Precision	Recall	AUC-ROC
GMM	**0.983**	**0.990**	0.983	**0.998**	0.906
OCSVM	0.931	0.949	0.985	0.927	0.906
IF	0.955	0.974	0.973	0.976	0.852
PCA	0.973	0.984	0.975	0.994	0.875
LOF	0.972	0.985	**0.992**	0.977	**0.951**
KNN	0.981	0.989	0.986	0.992	0.941

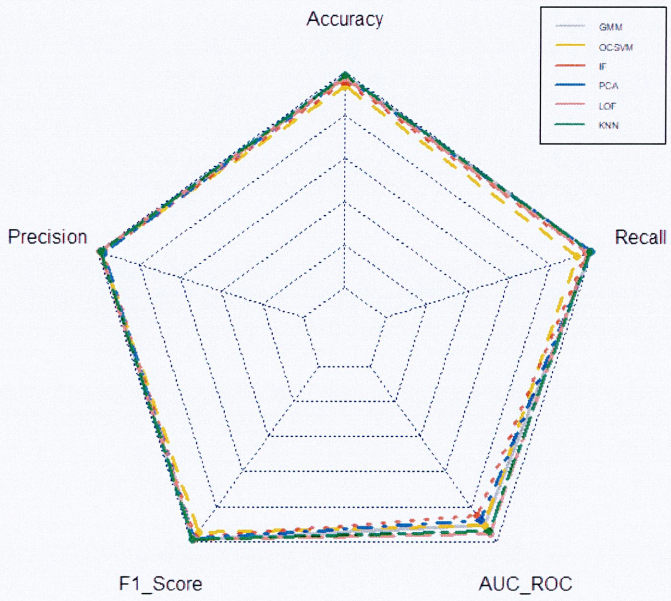


Fig. 7 Average OCC training results using full FTIR

The GMM method was introduced to be the best-performing method among all others, exhibiting impressive average scores of 0.983, 0.990, and 0.998 in accuracy, F1-score, and recall, respectively. It also showed a strong performance in precision and recall, achieving 0.983 and 0.906, respectively, thereby establishing itself as the method of choice for the study task before OCCs ensemble testing.

The OCSVM method exhibited average results of 0.931, 0.949, 0.985, 0.927, and 0.906 in accuracy, F1-score, precision, recall, and AUC-ROC, respectively. However, upon referring to **Table (4),** it is evident that the Diesel OCC of OCSVM hindered its overall performance, with a recall result of 0.669, indicating weak recognition of actual Diesel instances.

The IF algorithm exhibits superior performance in terms of accuracy, F1-score, precision, and recall compared to other methods, with average values of 0.955, 0.974, 0.973, and 0.976, respectively. However, it falls short regarding AUC-ROC, with an average value of 0.852, indicating that the model produces false positives and fails to isolate outliers accurately.

PCA emerges as a promising method, surpassing IF in both scenarios. When trained with full-dimension FTIR data, PCA achieved impressive average results in accuracy, F1-score, precision, recall, and AUC-ROC, scoring 0.973, 0.984, 0.975, 0.994, and 0.875, respectively.

LOF demonstrated exceptional performance in precision and AUC-ROC, achieving an average of 0.992 and 0.951, respectively, highlighting its ability to identify outliers within each product OCC accurately. However, regarding the accuracy, F1-score, and recall, LOF outperformed KNN, which achieved an average of 0.981, 0.989, and 0.977, respectively. KNN also displayed substantial precision and AUC-ROC results, averaging 0.986 and 0.941, respectively.

Table (6) presents the best CV results obtained by each OCC for each product across the implemented methods using dimensionally reduced FTIR data, whereas the Autoencoder did it with a minimum loss value of 0.109.

Table (6) OCCs Training results using dimensionally reduced FTIR

Method	OCC	Accuracy	F1_Socre	Precision	Recall	AUC-ROC
GMM	Gasoline	0.997	0.998	0.997	1	0.915
	Kerosene	0.939	0.966	0.979	0.953	0.878
	Diesel	0.614	0.690	0.941	0.545	0.709
	Base Oil	0.938	0.966	0.989	0.945	0.898
	Fuel Oil	0.999	1	0.999	1	0.979
Method	**OCC**	**Accuracy**	**F1_Socre**	**Precision**	**Recall**	**AUC-ROC**
OCSVM	Gasoline	0.999	1	0.999	1	0.975
	Kerosene	0.964	0.980	0.978	0.982	0.887
	Diesel	0.778	0.842	0.960	0.751	0.816
	Base Oil	0.956	0.976	0.989	0.964	0.910
	Fuel Oil	0.021	0.009	1	0.005	0.502
Method	**OCC**	**Accuracy**	**F1_Socre**	**Precision**	**Recall**	**AUC-ROC**
IF	Gasoline	0.999	0.999	0.999	1	0.964
	Kerosene	0.963	0.980	0.978	0.981	0.884
	Diesel	0.947	0.967	0.941	0.995	0.880
	Base Oil	0.980	0.989	0.989	0.989	0.920
	Fuel Oil	0.994	0.997	0.993	1	0.807
Method	**OCC**	**Accuracy**	**F1_Socre**	**Precision**	**Recall**	**AUC-ROC**
PCA	Gasoline	0.997	0.999	0.997	1	0.928
	Kerosene	0.969	0.983	0.977	0.989	0.883
	Diesel	0.955	0.972	0.947	0.998	0.895
	Base Oil	0.980	0.989	0.988	0.990	0.913
	Fuel Oil	0.988	0.994	0.993	0.995	0.790
Method	**OCC**	**Accuracy**	**F1_Socre**	**Precision**	**Recall**	**AUC-ROC**
LOF	Gasoline	0.997	0.999	0.998	0.999	0.943
	Kerosene	0.966	0.981	0.985	0.977	0.917
	Diesel	0.963	0.977	0.971	0.983	0.936
	Base Oil	0.978	0.988	0.991	0.985	0.932
	Fuel Oil	0.873	0.928	0.999	0.872	0.908
Method	**OCC**	**Accuracy**	**F1_Socre**	**Precision**	**Recall**	**AUC-ROC**
KNN	Gasoline	1	1	1	1	0.988
	KEROSENE	0.978	0.988	0.981	0.995	0.905
	DIESEL	0.960	0.975	0.965	0.985	0.926
	BASE OIL	0.977	0.988	0.986	0.990	0.899
	FUEL OIL	0.999	1	0.999	1	0.971

Table (7) and **Figure 8** exhibit the average results of training OCCs with dimensionally reduced FTIR data, which show a similarly successful optimization of each product OCC across all evaluation metrics, with average results above 0.81. Notably, overlooking the OCSVM performance on the dimensional reduced FTIR data, it could be above 0.87, and the optimization of OCCs remained impressive.

Table (7) Average OCC training results in dimensional reduced FTIR

OCC Method	Accuracy	F1_Socre	Precision	Recall	AUC-ROC
GMM	0.897	0.924	0.981	0.888	0.876
OCSVM	0.744	0.761	0.985	0.740	0.818
IF	0.976	0.986	0.980	0.993	0.891
PCA	0.978	0.987	0.980	**0.994**	0.882
LOF	0.955	0.974	**0.989**	0.963	0.927
KNN	**0.983**	**0.990**	0.986	0.994	**0.938**

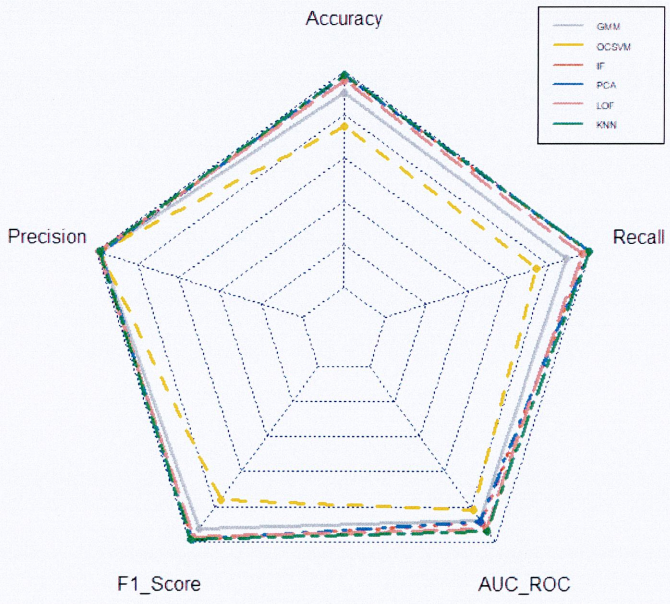


Fig. 8 Average OCC training results in dimensional reduced FTIR

When trained with dimensionally reduced FTIR data, GMM's performance was not as remarkable as it used to be with the full FTIR, as evidenced by its average accuracy of 0.897, F1-score of 0.924, precision of 0.981, recall of 0.888, and AUC-ROC of 0.876. Its performance, which was the second-lowest among all methods, suggests information loss due to missing transmittance at some wavenumbers that affected the model's estimations of covariance, thereby eluding its assumptions of data distribution.

On the other hand, OCSVM products' OCCs yielded an average accuracy of 0.744, F1-score of 0.761, precision of 0.985, recall of 0.74, and AUC-ROC of 0.818, which were the lowest results among all methods. Despite the Diesel OCC performance, it is apparent from **Table (6)** that the model became useless in profiling Fuel Oil, returning an accuracy of 0.021, F1-score of 0.009, precision of 1, recall of 0.005, and AUC-ROC of 0.502. It is worth mentioning that lack of optimization of OCC is not considered as each product OCC is individually optimized through the CV technique.

Despite not suffering from the curse of dimensionality due to its isolation trees baseline, IF's performance is enhanced when trained with dimensionally reduced FTIR data, resulting in an average accuracy of 0.976, F1-score of 0.986, precision of 0.980, recall of 0.993, and AUC-ROC of 0.891.

PCA is averaging at 0.978, 0.987, 0.980, 0.994, and 0.882 for accuracy, F1-score, precision, recall, and AUC-ROC, respectively.

Interestingly, when the dimensionality of FTIR data was reduced, both LOF and KNN demonstrated slightly lower performance across all evaluation metrics, with LOF yielding average results of 0.955, 0.974, 0.989, 0.963, and 0.927 and KNN yielding average results of 0.983, 0.990, 0.986, 0.994, and 0.938 for accuracy, F1-score, precision, recall, and AUC-ROC, respectively.

4.3.2. Computation Time Assessment of OCCs

In comparing one classification method which differs in functionality, the ensemble time consumed was measured for each method as shown in **Table (8)** in both scenarios, full and dimensionally reduced FTIR data.

Table (8) methods' computation time

Method	Methods Computation Time	
	Full	Dimension Reduced
GMM-based	7.54 s +/- 190 ms	0.40 s +/- 0.01 s
OCSVM-based	189 ms +/- 5.67 ms	10.10 ms +/- 0.30 ms
IF-based	2.98 s +/- 84.5 ms	0.16 s +/- 0.005 s
PCA-based	206 ms +/- 9.44 ms	10.99 ms +/- 0.50 ms
LOF-based	**187 ms +/- 6.47 ms**	**9.99 ms +/- 0.34 ms**
KNN-based	213 ms +/- 5.31 ms	11.39 ms +/- 0.28 ms

For full FTIR data, the GMM-based method had the longest computation time, averaging 7.54 s +/- 190 ms, followed by the IF-based method with an average of 2.98 s +/- 84.5 ms, the KNN-based method with an average of 213 ms +/- 5.31 ms, PCA-based method with an average of 206 ms +/- 9.44 ms, OCSVM-based method with an average of 189 ms +/- 5.67 ms, and finally, the LOF-based method with an average of 187 ms +/- 6.47 ms.

For dimensionally reduced FTIR data, the computation time was significantly lower for all methods. The GMM-based method had an average computation time of 0.40 s +/- 0.01 s, followed by the IF-based method with an average of 0.16 s +/- 0.005 s, KNN-based method with an average of 11.39 ms +/- 0.28 ms, PCA-based method with an average of 10.99 ms +/- 0.50 ms, OCSVM-based method with an average of 10.10 ms +/- 0.30 ms, and finally, the LOF-based method with an average of 9.99 ms +/- 0.34 ms.

4.3.3. Testing Phase OCCs

The ensembles' feasibility and effectiveness in profiling new unlabelled petroleum instances is presented in this section. The unknown presence in the testing is where an instance had failed to be recognized by any of the OCCs with an ensemble, or is being admitted by more than one OCC. This instance is later treated by using the Mahalanbios distance.

Table (9) presents the count of modelling samples for each OCC. Collectively, 80% of the FTIR data was used to construct the OCC models prior to the testing phase using the best hyperparameter combinations of the CV for each OCC. The rest, 20%, test samples were all passed to all OCCs to get their predictions.

Table (9) Testing phase samples

Testing Phase		
Target Class	**Modeling Samples**	**Test Samples**
Gasoline	29	109
KEROSENE	112	109
DIESEL	185	109
BASE OIL	85	109
FUEL OIL	25	109

The testing results of each OCCs ensemble are presented in **Table (9)**.

Table (10) Full FTIR OCCs ensembles findings

Method	OCC	F1-Score	Precision	Recall	AUC-ROC	Unknowns
GMM	Gasoline	1	1	1	0.991	2
	Kerosene	1	1	1	0.986	
	Diesel	0.989	1	0.979	0.992	
	Base Oil	0.957	1	0.917	0.995	
	Fuel Oil	1	1	1	0.990	
	Accuracy	**0.982**				
Method	**OCC**	**F1-Score**	**Precision**	**Recall**	**AUC-ROC**	**Unknowns**
OCSVM	Gasoline	1	1	1	0.986	3
	Kerosene	0.935	0.923	0.947	0.931	
	Diesel	0.901	0.953	0.854	0.923	
	Base Oil	0.880	0.846	0.917	0.979	
	Fuel Oil	1	1	1	0.985	
	Accuracy	0.908				
Method	**OCC**	**F1-Score**	**Precision**	**Recall**	**AUC-ROC**	**Unknowns**
IF	Gasoline	1	1	1	1	0
	Kerosene	0.932	0.971	0.895	0.940	
	Diesel	0.957	1	0.917	0.958	
	Base Oil	0.769	0.714	0.833	0.896	
	Fuel Oil	0.762	0.615	1	0.975	
	Accuracy	0.908				
Method	**OCC**	**F1-Score**	**Precision**	**Recall**	**AUC-ROC**	**Unknowns**
PCA	Gasoline	1	1	1	0.986	3
	Kerosene	0.806	0.931	0.711	0.820	
	Diesel	0.851	0.811	0.896	0.879	
	Base Oil	0.880	0.846	0.917	0.979	
	Fuel Oil	1	1	1	0.985	
	Accuracy	0.844				
Method	**OCC**	**F1-Score**	**Precision**	**Recall**	**AUC-ROC**	**Unknowns**
LOF	Gasoline	1	1	1	1	0
	Kerosene	0.944	1	0.895	0.947	
	Diesel	0.947	0.957	0.938	0.952	
	Base Oil	0.923	0.857	1	0.990	
	Fuel Oil	0.842	0.727	1	0.985	
	Accuracy	0.936				
Method	**OCC**	**F1-Score**	**Precision**	**Recall**	**AUC-ROC**	**Unknowns**
KNN	Gasoline	1	1	1	0.995	1
	Kerosene	0.944	1	0.895	0.940	
	Diesel	0.921	1	0.854	0.919	
	Base Oil	0.647	0.500	0.917	0.943	
	Fuel Oil	1	1	1	0.995	
	Accuracy	0.890				

The corresponding average of the testing results in **Table (10)** are presented in **Table (11)**.

Table (11) Full FTIR average OCCs ensembles findings

Method	Accuracy	F1-Score	Precision	Recall	AUC-ROC	Unknowns
GMM	**0.982**	**0.989**	**1**	**0.979**	**0.991**	2
OCSVM	0.908	0.943	0.945	0.944	0.961	3
IF	0.908	0.884	0.86	0.929	0.954	**0**
PCA	0.844	0.907	0.918	0.905	0.93	3
LOF	0.936	0.931	0.908	0.966	0.975	**0**
KNN	0.89	0.903	0.9	0.933	0.959	1

Table (12) shows the testing results including their average update after computing the Mahalanobis distance for the unknowns to the estimated center of each product data points.

Table (12) Original FTIR OCCs ensembles Mahalanobis distance findings

Method	OCC	F1-Score	Precision	Recall	AUC-ROC
GMM	Gasoline	1	1	1	1
	Kerosene	1	1	1	1
	Diesel	1	1	1	1
	Base Oil	1	1	1	1
	Fuel Oil	1	1	1	1
	Accuracy	**1**			
Method	**OCC**	**F1-Score**	**Precision**	**Recall**	**AUC-ROC**
OCSVM	Gasoline	1	1	1	1
	Kerosene	0.935	0.923	0.947	0.953
	Diesel	0.925	0.956	0.896	0.932
	Base Oil	0.923	0.857	1	0.990
	Fuel Oil	1	1	1	1
	Average	0.957	0.947	0.969	0.975
	Accuracy	0.936			
Method	**OCC**	**F1-Score**	**Precision**	**Recall**	**AUC-ROC**
PCA	Gasoline	1	1	1	1
	Kerosene	0.806	0.931	0.711	0.841
	Diesel	0.874	0.818	0.938	0.887
	Base Oil	0.923	0.857	1	0.990
	Fuel Oil	1	1	1	1
	Average	0.921	0.921	0.930	0.944
	Accuracy	0.872			
Method	**OCC**	**F1-Score**	**Precision**	**Recall**	**AUC-ROC**
KNN	Gasoline	1	1	1	1
	Kerosene	0.944	1	0.895	0.947
	Diesel	0.921	1	0.854	0.927
	Base Oil	0.686	0.522	1	0.943
	Fuel Oil	1	1	1	1
	Average	0.910	0.904	0.950	0.964
	Accuracy	0.899			

Figure 9 and **Figure 10** show full FTIR OCCs ensembles' average performance before and after implementing Mahalanobis distance.

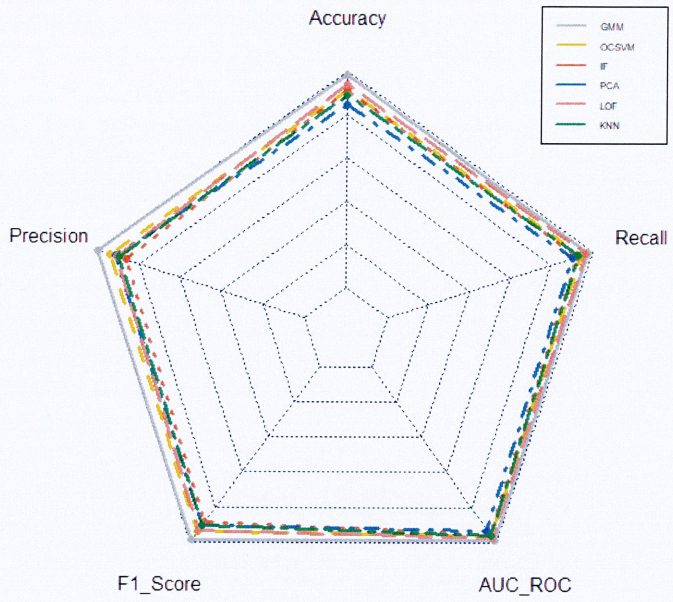


Fig. 9 Full FTIR OCCs ensembles average performance before Mahalanobis

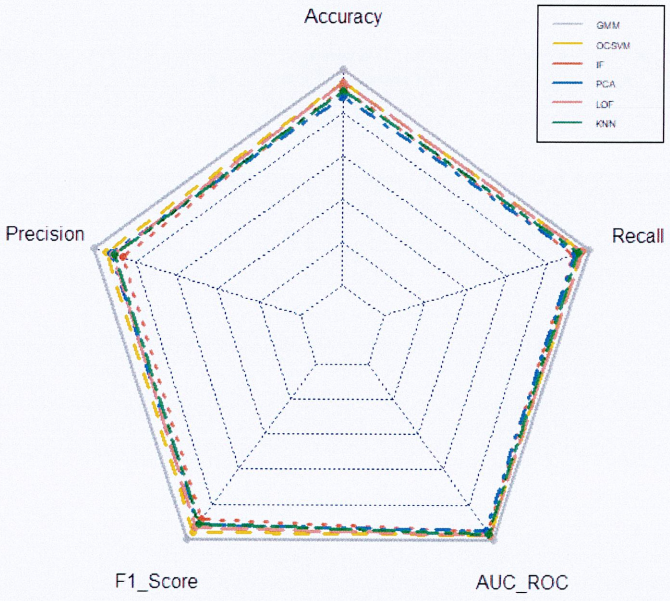


Fig. 10 Full FTIR OCCs ensembles average performance after Mahalanobis

Compared to other methods, the GMM-based ensemble performed the best with an average accuracy of 0.982, F1-score of 0.989, precision of 1, recall of 0.979, and AUC-ROC of 0.991 at full FTIR data dimensions.

However, two instances were not confidently recognized by the GMM-based ensemble. The Mahalanobis distance cleared these two unknowns, resulting in a perfect performance with average accuracy, F1-score, precision, recall, and AUC-ROC of 1.

OCSVM-based ensemble with full FTIR data dimensions showed impressive performance, with average results of 0.908, 0.943, 0.945, 0.944, and 0.961 for accuracy, F1-score, precision, recall, and AUC-ROC, respectively. During the experiment, three unknowns appeared, and the application of Mahalanobis distance improved the performance to average results of 0.936, 0.957, 0.947, 0.969, and 0.975 for accuracy, F1-score, precision, recall, and AUC-ROC, respectively. These results indicate a true advantage of the OCCs ensemble, where OCSVM-based recognized Diesel instances effectively, even though the Diesel OCC was hesitant during the training phase.

On the other hand, the IF-based ensemble demonstrated impressive performance, with an average accuracy of 0.908, F1-score of 0.884, precision of 0.86, recall of 0.929, and AUC-ROC of 0.954. Although not the best-performing method, the IF-based ensemble did not return any unknown FTIR instances, indicating its precise and selective baseline. However, the model struggled with Base Oil and Fuel Oil OCCs, which showed low precision and hindered its performance by misclassifying instances.

PCA-based with the full FTIR dimension data returned average results of 0.844, 0.907, 0.918, 0.905, and 0.930 for accuracy, F1-score, precision, recall, and AUC-ROC, respectively. Nevertheless, three unknowns appeared and were successfully profiled by Mahalanobis distance which improved the performance to average results of 0.872, 0.921, 0.921, 0.930, and 0.944 for accuracy, F1-score, precision, recall, and AUC-ROC, respectively.

The LOF-based approach demonstrated remarkable success in profiling all instances, achieving average scores of 0.936, 0.931, 0.908, 0.966, and 0.975 for accuracy, F1-score, precision, recall, and AUC-ROC, respectively.

The KNN-based achieved an average accuracy of 0.89, F1-score of 0.903, precision of 0.9, recall of 0.933, and AUC-ROC of 0.959. Upon unknown appearance, Mahalanobis distance is applied, and the approach correctly profiled one unknown instance and improved the results to an accuracy of 0.899, F1-score of 0.910, precision of 0.904, recall of 0.950, and AUC-ROC of 0.964.

Table (13) presents the testing results of the ensemble of each OCC with the dimensionally reduced FTIR data.

Table (13) Dimensionally reduced FTIR OCCs ensembles findings

Method	OCC	F1-Score	Precision	Recall	AUC-ROC	Unknowns
GMM	Gasoline	1	1	1	0.981	1
	Kerosene	0.937	0.902	0.974	0.931	
	Diesel	0.886	0.975	0.813	0.902	
	Base Oil	0.800	0.769	0.833	0.974	
	Fuel Oil	1	1	1	0.980	
	Accuracy	0.890				
Method	OCC	F1-Score	Precision	Recall	AUC-ROC	Unknowns
OCSVM	Gasoline	1	1	1	0.995	1
	Kerosene	0.348	1	0.211	0.598	
	Diesel	0	0	0	0.492	
	Base Oil	0.545	0.600	0.500	0.771	
	Fuel Oil	0.147	0.080	0.875	0.537	
	Accuracy	0.220				
Method	OCC	F1-Score	Precision	Recall	AUC-ROC	Unknowns
IF	Gasoline	0.909	1	0.833	0.917	0
	Kerosene	0.915	0.931	0.900	0.937	
	Diesel	0.891	0.953	0.837	0.902	
	Base Oil	0.882	0.882	0.882	0.930	
	Fuel Oil	0.545	0.400	0.857	0.884	
	Accuracy	0.862				
Method	OCC	F1-Score	Precision	Recall	AUC-ROC	Unknowns
PCA	Gasoline	0.857	0.750	1	0.995	1
	Kerosene	0.857	0.938	0.789	0.881	
	Diesel	0.822	0.881	0.771	0.844	
	Base Oil	0.606	0.476	0.833	0.860	
	Fuel Oil	0.889	0.800	1	0.990	
	Accuracy	0.807				
Method	OCC	F1-Score	Precision	Recall	AUC-ROC	Unknowns
LOF	Gasoline	1	1	1	1	0
	Kerosene	0.960	0.973	0.947	0.967	
	Diesel	0.945	1	0.896	0.948	
	Base Oil	0.857	0.750	1	0.979	
	Fuel Oil	0.889	0.800	1	0.990	
	Accuracy	0.936				
Method	OCC	F1-Score	Precision	Recall	AUC-ROC	Unknowns
KNN	Gasoline	1	1	1	0.995	1
	Kerosene	0.944	1	0.895	0.940	
	Diesel	0.909	1	0.833	0.908	
	Base Oil	0.629	0.478	0.917	0.938	
	Fuel Oil	1	1	1	0.995	
	Accuracy	0.881				

The corresponding average of the testing results in **Table (13)** are presented in **Table (14)**.

Table (14) Dimensionally reduced FTIR OCCs ensembles average findings

Method	Accuracy	F1-Score	Precision	Recall	AUC-ROC	Unknowns
GMM	0.89	0.925	**0.929**	0.924	0.954	1
OCSVM	0.22	0.408	0.536	0.517	0.679	1
IF	0.862	0.829	0.833	0.862	0.914	**0**
PCA	0.807	0.806	0.769	0.879	0.914	1
LOF	**0.936**	**0.93**	0.905	**0.969**	**0.977**	**0**
KNN	0.881	0.896	0.896	0.929	0.955	1

Table (15) shows the updated testing results obtained by computing the Mahalanobis distance of the unknowns to the center of each product data points, using the dimensionally reduced FTIR data.

Table (15) Dimensionally reduced FTIR OCCs ensembles Mahalanobis findings

Method	OCC	F1-Score	Precision	Recall	AUC-ROC
GMM	Gasoline	1	1	1	1
	Kerosene	0.925	0.881	0.974	0.952
	Diesel	0.886	0.975	0.813	0.898
	Base Oil	0.786	0.688	0.917	0.933
	Fuel Oil	1	1	1	1
	Average	0.919	**0.909**	0.941	0.956
	Accuracy	0.899			
Method	**OCC**	**F1-Score**	**Precision**	**Recall**	**AUC-ROC**
OCSVM	Gasoline	1	1	1	1
	Kerosene	0.348	1	0.211	0.605
	Diesel	0	0	0	0.500
	Base Oil	0.609	0.636	0.583	0.771
	Fuel Oil	0.147	0.080	0.875	0.541
	Average	0.421	0.543	0.534	0.684
	Accuracy	0.229			
Method	**OCC**	**F1-Score**	**Precision**	**Recall**	**AUC-ROC**
PCA	Gasoline	1	1	1	0.995
	Kerosene	0.857	0.938	0.789	0.874
	Diesel	0.822	0.881	0.771	0.836
	Base Oil	0.606	0.476	0.833	0.902
	Fuel Oil	0.889	0.800	1	0.985
	Average	0.835	0.819	0.879	0.918
	Accuracy	0.807			
Method	**OCC**	**F1-Score**	**Precision**	**Recall**	**AUC-ROC**
KNN	Gasoline	1	1	1	1
	Kerosene	0.944	1	0.895	0.947
	Diesel	0.909	1	0.833	0.917
	Base Oil	0.667	0.500	1	0.938
	Fuel Oil	1	1	1	1
	Average	0.904	0.900	0.946	0.960
	Accuracy	0.890			

Figure 11 and **Figure 12** show reduced FTIR OCCs ensembles average performance before and after implementing Mahalanobis distance.

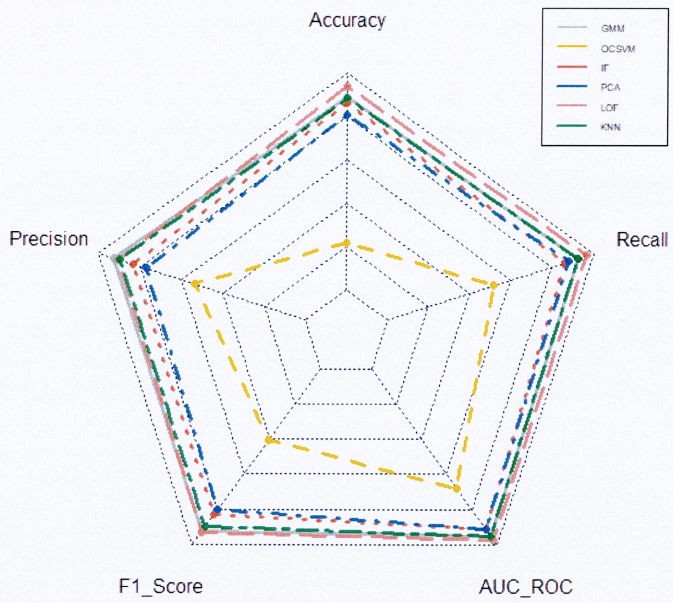


Fig. 11 Reduced FTIR OCCs ensembles average performance before Mahalanobis

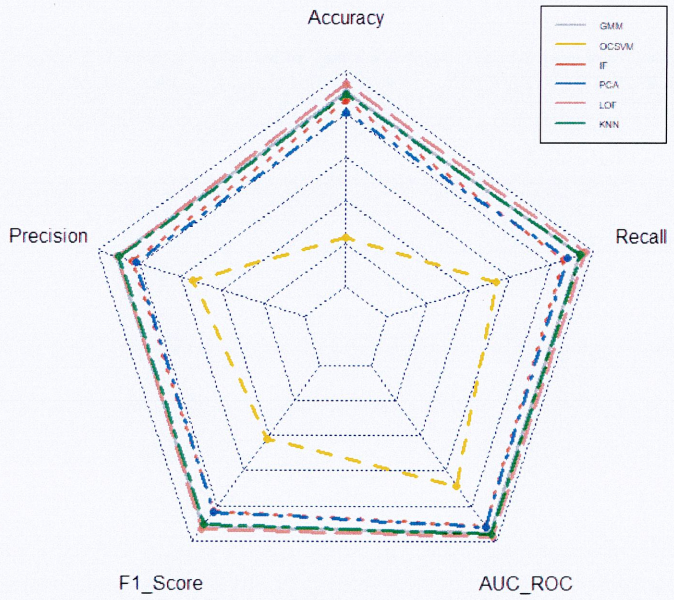


Fig. 12 Reduced FTIR OCCs ensembles average performance after Mahalanobis

In the dimensional reduced FTIR data scenario, the GMM-based ensemble experienced a precision drop in classifying Base Oil from 0.957 to 0.8 but still achieved the highest average precision compared to other methods. The GMM-based ensemble had an average accuracy of 0.89, F1-

score of 0.925, precision of 0.929, recall of 0.924, and AUC-ROC of 0.954, with two unknowns requiring the application of Mahalanobis distance. Hence, after profiling the two unknowns, the average performance results became 0.899, 0.919, 0.909, 0.941, and 0.956 for accuracy, F1-score, precision, recall, and AUC-ROC, respectively.

The OCSVM-based ensemble's performance showed a significant drop compared to its performance with full FTIR data, with an average accuracy of 0.22, F1-score of 0.408, precision of 0.536, recall of 0.517, and AUC-ROC of 0.679, along with one unknown instance.

The IF-based ensemble still returned 0 unknowns, indicating its high dedication to identifying the new instances. However, the missing information affected the Fuel Oil OCC's performance, which scored only 0.4 for precision, even though it performed well during the training phase. This led to a slight drop in the average performance compared to the full FTIR scenario, with an accuracy of 0.862, F1-score of 0.829, precision of 0.833, recall of 0.862, and AUC-ROC of 0.914.

A slight drop in the performance of PCA-based in the dimensional reduced FTIR data scenario with an average accuracy of 0.807, F1-score of 0.806, precision of 0.769, recall of 0.879, and AUC-ROC of 0.914. One unknown instance was profiled by Mahalnbios distance to improve the results to an average accuracy of 0.807, F1-score of 0.835, precision of 0.819, recall of 0.879, and AUC-ROC of 0.918.

Notably, even in the case of dimensional reduced FTIR data, the LOF-based method delivered consistent and impressive results with an average accuracy score of 0.936, an F1-score of 0.930, a precision score of 0.905, a recall score of 0.969, and an AUC-ROC score of 0.977.

Additionally, the KNN-based approach maintained its performance even with reduced dimensions of FTIR data, achieving an average accuracy of 0.881, F1-score of 0.896, precision of 0.896, recall of 0.929, and AUC-

ROC of 0.955. It is worth noting that the approach returned one unknown instance, and applying the Mahalanobis distance technique improved the performance slightly, resulting in an average accuracy of 0.89, an F1-score of 0.904, a precision of 0.900, a recall of 0.946, and an AUC-ROC of 0.960.

4.3.4. ANN Findings

The results of optimizing ANN's parameters, such as neurons, layers, transfer functions, and optimizer types, did not significantly improve the model's performance. The highest average CV results for accuracy, F1-score, precision, recall, and AUC-ROC were 0.675, 0.446, 0.418, 0.504, and 0.692, respectively, as shown in **Table (16)**.

Table (16) ANN training findings

Training phase	F1_Socre	Precision	Recall	AUC-ROC
	0.446	0.418	0.504	0.692
Accuracy	0.675			

The test phase results for ANN, in **Table (16)**, were almost negligible, except for Diesel, due to the higher sample count, as noted in **Table (8)**.

Table (17) ANN testing findings

Testing phase	F1_Socre	Precision	Recall
GASOLINE	0	0	0
KEROSENE	0	0	0
DIESEL	0.61	0.44	1
BASE OIL	0	0	0
FUEL OIL	0	0	0
Average	0.122	0.088	0.2
Accuracy	0.44		

4.4. Discussion

Exploring the study findings in both scenarios, with and without dimension reduction of comparing OCCs ensembles in profiling petroleum products, is paramount.

In a contest, the readings of training OCCs with full FTIR in **Table (4)** bring up Diesel OCSVM to light as it is the only lagging among all OCCs across all methods. One potential limitation of OCSVM is its sensitivity to high-dimensional data and its challenges for tuning complexity. Which sets it apart from other methods and may require more careful consideration in practical applications. Additionally, referring to the performance with the dimensionally reduced FTIR data, as it is demonstrated in **Table (6)**, only OCSVM seems to have missed the specific characteristics of Fuel Oil. Nevertheless, the method's high capability in other OCCs covered these hindrances at the OCCs ensemble level. Moreover, given its original property in lower dimension space projection, PCA's performance slightly improved when trained with dimensionally reduced FTIR data.

However, it might not be clear in the training stage, but in the ensembles testing with fully FTIR data, KNN-based and IF-based showed a lack of precision at profiling Base Oil as presented in **Table (10)**, with F1-score of 0.647 and 0.769, respectively. The ensembles tend to falsy accept instances such as Base Oil, which hinders other products OCC from profiling their own. Additionally, IF-based had an F1-score of 0.762 at Fuel Oil profiling. However, IF has a limitation in that the threshold of trees in isolating the outliers is not flexible; all trees in an IF-OCC model have the same threshold value. On the other hand, when looking at the other methods' high performance, it shows that the problem is neither the preprocessing of it nor Basel oil FTIR data, and it is not the sample size as

Fuel Oil nearly equals to Gasoline samples size. Therefore, it emphasizes that the nature of the problem demands consideration of the OCC methods' interpretability, performance, and computational efficiency. GMM-based shows perfect performance, indicating a possible correct assumption of Gaussian distributions among the data instances.

Moreover, the readings in **Table (13)** present the testing performance of the ensembles when FTIR data have been reduced; the OCSVM-based has failed as a system, clearly mixing between Diesel, Base Oil, and Fuel Oil. Despite the successful application of Mahalanobis distance, the results only showed a slight improvement, with an average accuracy of 0.229, F1-score of 0.421, precision of 0.543, recall of 0.534, and AUC-ROC of 0.684. Unfortunately, the model's weak performance remained unchanged even after eliminating the unknown instance. The results show the inconvenience of reducing FTIR data dimension for OCSVM, which is inconsistence with previous works. GMM-based had a performance drop compared to other methods. Base Oil became a problem for PCA and KNN based. Interestingly, IF-based overcame the Base Oil profiling problem and had more decrease in Fuel Oil precision, where its F1-score is 0.545. The performance across all evaluation metrics suggests that dimensionality reduction negatively impacted the model's ability to profile petroleum products more accurately.

It is worth mentioning that uncertainty and returning unprofiled unknowns have been reduced to one instance only after lowering the dimension to indicate a possible confusion occurred by some features or wavenumbers, but that does not justify the performance drop for these methods. **Table (12)** and **Table (15)** show how well-performing Mahalabios distance is in assigning these unknown instances. However, comparing after and before applying the distance approach and investigating the precision and recall trade-offs across the OCCs, the

instance showed to be Base Oil. It can be considered an in-data outlier. Moreover, IF-based and LOF-based show superiority in overcoming such outliers in both scenarios, full and reduced FTIR data, compared to other methods.

Overall, the study findings suggest that LOF can achieve a high level of performance, not as perfect as GMM-based in the first scenario, but it demonstrates high performance in both cases. Furthermore, it suggests that the LOF-based method can be a suitable choice for applications where computation time is a critical factor. Underscoring its effectiveness in profiling data regardless of dimensionality reduction, where its ability to detect low-density regions helped to overcome the products overlap.

In contrast, OCCs ensembles demonstrated impressive overall performance in addressing the imbalanced data issue with the FTIR data and inconsistency with the literature regarding dimension reduction effectiveness in raising up the performance. Moreover, compared to ANN, the ensembles showed much better performance.

Chapter Five

Conclusion and Future Work

5.1. Conclusion

5.2. Future work

Chapter Five

Conclusion and Future Work

5.1. Conclusion

The problem of profiling petroleum products using FTIR was handled in the current work taking into account the issue of imbalanced data using the advances in the ML field. It showed how carefully selecting the proper OCC algorithm in the ensemble implementation for profiling petroleum products using FTIR data is crucial, considering specific performance metrics. Based on the study findings, all the explored models demonstrated some levels of effectiveness, and particular algorithms showed superior performance. The GMM-based ensemble was found to be the most effective, despite requiring significant computational effort. In contrast, the LOF-based ensemble was the fastest but encountered challenges with Fuel Oil profiling, and its reachability distance process made its interpretation challenging.

Nevertheless, it was able to identify all test instances, unlike the second-fastest, OCSVM-based ensemble, which returned unknown instances, showing its selectivity and uncertainty in a few cases. Overall, LOF-based and OCSVM-based methods demonstrated very similar performance with full FTIR data, but LOF held on when the FTIR dimension was reduced. On the other hand, while PCA-based and KNN-based methods were uncertain in some instances, they demonstrated adequate performance but still lagged behind the other methods.

When the FTIR data was dimensionally reduced, several methods, particularly the OCSVM-based method, were significantly impacted due to specific information loss that affected Fuel Oil OCC profiling. However, other methods performed well despite the change. GMM-based, PCA-

based, and IF-based methods showed only a slight drop in performance. The access to the neighbors within the data points of KNN-based and LOF-based allowed them to maintain almost the same level of performance across all evaluation metrics. However, the dimension reduction of the Autoencoder with spectroscopic data made it difficult to interpret why some methods and not others missed specific characteristics.

Further analysis revealed that specific fractions in the Diesel data confused the OCSVM-based method but not other methods, despite its majority in sample count. The dimensional reduction affected the performance of Base Oil and Fuel Oil OCCs, which was not observed during training but became evident during testing, indicating specific information loss for some methods. An ensemble that consists of different OCC-based methods, with each method being superior in profiling a particular product, may be a potential improvement.

In contrast, it can be concluded that the performance of the ensembles is highly dependent on the quality of data available, and the impact of dimensional reduction on some methods cannot be ignored. A possible improvement could be performing dimensional reduction separately for each product, leading to a more focused and accurate analysis. Furthermore, it was observed that the Mahalanobis distance approach showed superior performance in assisting the ensembles' decisions in profiling unknowns. However, the distance approach is not optimized for the data and cannot be generalized. Nevertheless, when validating against the multiclass ANN, the OCC ensembles demonstrated superior performance in profiling FTIR petroleum products and addressing the imbalanced data issue. Overall, the findings suggest that using OCC ensembles can lead to more accurate and effective profiling of petroleum products. This study not only efficiently addresses the challenges of profiling petroleum products based on their FTIR spectra outputs but also

introduces a unique profiling approach in the FTIR and petroleum field. By employing an ensemble of OCCs, the issue of imbalanced class distribution is effectively tackled, surpassing a conventional multiclass classifier. The significant improvement in classification accuracy achieved by this innovative approach enables precise identification and profiling of petroleum products. The proposed methodology not only demonstrates its efficiency in handling the unique data problems in this field but also marks a distinctive advancement in the profiling of petroleum products using FTIR spectroscopy.

5.2. Future Work

The focus of the current work was on the use of OCC-based ensembles for profiling petroleum products using FTIR data, and the results showed the effectiveness of these techniques for the application under consideration, but this work could be enhanced in several directions. First, the current work used a relatively small dataset, limiting the findings' generalizability. Therefore, it is worth expanding the dataset to include a broader range of petroleum products and a larger sample size to improve the representativeness of the results. Additionally, it is worth investigating different spectroscopic data types to explore the impact on the performance of OCC-based ensembles. Second, it is worth considering other applications and trying other approaches, such as deep learning architectures or hybrid models that combine OCC with other techniques or varieties of the OCC methods baseline within the ensemble, in the way each OCC is imposed to a product.

References

Aggarwal, C. C. (2017). *Outlier Analysis*. Springer International Publishing. https://doi.org/10.1007/978-3-319-47578-3

ASTM. (2015). Standard Test Method for Solvent-Free Membrane Recoverable Oil and Grease by Infrared Determination. *Astm*.

Bacci, M., Fabbri, M., Picollo, M., & Porcinai, S. (2001). Non-invasive fibre optic Fourier transform-infrared reflectance spectroscopy on painted layers: Identification of materials by means of principal component analysis and Mahalanobis distance. *Analytica Chimica Acta*, *446*(1–2). https://doi.org/10.1016/S0003-2670(01)01057-1

Ban, T., & Abe, S. (n.d.). *Implementing Multi-class Classifiers by One-class Classification Methods*.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (n.d.). *LOF: Identifying Density-Based Local Outliers*.

Brown, C. W., Lynch, P. F., & Ahmadjian, M. (1975). Applications of Infrared Spectroscopy in Petroleum Analysis and Oil Spill Identification. *Applied Spectroscopy Reviews*, *9*(1), 223–248. https://doi.org/10.1080/05704927508081491

Calle, J. L. P., Ferreiro-González, M., Ruiz-Rodríguez, A., Fernández, D., & Palma, M. (2022). Detection of Adulterations in Fruit Juices Using Machine Learning Methods over FT-IR Spectroscopic Data. *Agronomy*, *12*(3). https://doi.org/10.3390/agronomy12030683

Deisenroth, M. P., Faisal, A. A., & Ong, C. S. (2020). *Mathematics for machine learning*. Cambridge University Press & Assessment.

Deng, X., Li, W., Liu, X., Guo, Q., & Newsam, S. (2018). One-class remote sensing classification: One-class vs. Binary classifiers. *International Journal of Remote Sensing*, *39*(6). https://doi.org/10.1080/01431161.2017.1416697

DETERMINATION OF CRUCIAL PARAMETERS IN GASOLINE BLENDS BY USING INFRARED SPECTROSCOPY COUPLED WITH MULTIVARIATE CALIBRATION METHODS. (n.d.).

DETERMINATION OF HYDROCARBON COMPOSITION OF NAPHTHA BY USING FOURIER TRANSFORM INFRARED SPECTROSCOPY AND MULTIVARIATE CALIBRATION. (2020).

Figueiredo, M. A. T., & Jain, A. K. (2002). Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(3). https://doi.org/10.1109/34.990138

Gallagher, N. B. (2020). *Savitzky-Golay Smoothing and Differentiation Filter*. www.Eigenvector.com

Gulli, A., & Pal, S. (2017). Deep learning with Keras. *Packt Publishing Ltd*.

Gyftokostas, N., Nanou, E., Stefas, D., Kokkinos, V., Bouras, C., & Couris, S. (2021). Classification of Greek olive oils from different regions by machine learning-aided laser-induced breakdown spectroscopy and absorption spectroscopy. *Molecules*, *26*(5). https://doi.org/10.3390/molecules26051241

Gyunka, B. A., & Barda, S. I. (2020). Anomaly detection of android malware using One-Class K-Nearest Neighbours (OC-KNN). *Nigerian Journal of Technology*, *39*(2). https://doi.org/10.4314/njt.v39i2.25

Hadjadji, B., Chibani, Y., & Guerbai, Y. (2014). Multiple one-class classifier combination for multi-class classification. *Proceedings - International Conference on Pattern Recognition*, 2832–2837. https://doi.org/10.1109/ICPR.2014.488

Hayashi, T., & Fujita, H. (2022). One-class ensemble classifier for data imbalance problems. *Applied Intelligence*, *52*(15). https://doi.org/10.1007/s10489-021-02671-1

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016-December*. https://doi.org/10.1109/CVPR.2016.90

Houston, J., Glavin, F. G., & Madden, M. G. (2020). Robust Classification of High-Dimensional Spectroscopy Data Using Deep Learning and Data Synthesis. *Journal of Chemical Information and Modeling, 60*(4). https://doi.org/10.1021/acs.jcim.9b01037

Ian Goodfellow, Yoshua Bengio, A. C. (2016). Deep Learning - Ian Goodfellow, Yoshua Bengio, Aaron Courville - Google Books. In *MIT Press*.

Korb, E., Bağcıoğlu, M., Garner-Spitzer, E., Wiedermann, U., Ehling-Schulz, M., & Schabussova, I. (2020). Machine learning-empowered ftir spectroscopy serum analysis stratifies healthy, allergic, and sit-treated mice and humans. *Biomolecules, 10*(7), 1–17. https://doi.org/10.3390/biom10071058

Krawczyk, B., Galar, M., Woźniak, M., Bustince, H., & Herrera, F. (2018). Dynamic ensemble selection for multi-class classification with one-class classifiers. *Pattern Recognition, 83*, 34–51. https://doi.org/10.1016/j.patcog.2018.05.015

Krawczyk, B., & Woźniak, M. (2015). Hypertension Type Classification Using Hierarchical Ensemble of One-Class Classifiers for Imbalanced Data. *Advances in Intelligent Systems and Computing, 311*. https://doi.org/10.1007/978-3-319-09879-1_34

Krawczyk, B., Woźniak, M., & Herrera, F. (2015). On the usefulness of one-class classifier ensembles for decomposition of multi-class problems. *Pattern Recognition, 48*(12). https://doi.org/10.1016/j.patcog.2015.06.001

Li, J., Zuo, L., Su, T., & Guo, Z. (2021). The Choice of Kernel Function for One-Class Support Vector Machine. *Proceedings - 2021 2nd International Conference on Big Data and Artificial Intelligence and Software Engineering, ICBASE 2021.* https://doi.org/10.1109/ICBASE53849.2021.00044

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*, 413–422. https://doi.org/10.1109/ICDM.2008.17

Lovatti, B. P. O., Silva, S. R. C., Portela, N. de A., Sad, C. M. S., Rainha, K. P., Rocha, J. T. C., Romão, W., Castro, E. V. R., & Filgueiras, P. R. (2019). Identification of petroleum profiles by infrared spectroscopy and chemometrics. *Fuel, 254.* https://doi.org/10.1016/j.fuel.2019.115670

Maglaras, L. A., & Jiang, J. (2015). A novel intrusion detection method based on OCSVM and K-means recursive clustering. *ICST Transactions on Security and Safety*, *2*(3). https://doi.org/10.4108/sesa.2.3.e5

Mahalanobis P. C. (1936). On the generalized distance in statistics. *Proceedings of the National Institute of Sciences of India*, *2*, 49–55.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference.* https://doi.org/10.25080/majora-92bf1922-00a

Patel, E., & Kushwaha, D. S. (2020). Clustering Cloud Workloads: K-Means vs Gaussian Mixture Model. *Procedia Computer Science, 171.* https://doi.org/10.1016/j.procs.2020.04.017

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M.,

& Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12.*

Peters Kenneth E., Walters Clifford C., & Moldowan J. Michael. (2007). *The biomarker guide: Biomarkers and isotopes in the environment and human history volume 1 (2nd ed.).*

Press, W. H., & Teukolsky, S. A. (1990). Savitzky-Golay Smoothing Filters. *Computers in Physics, 4*(6), 669. https://doi.org/10.1063/1.4822961

Ramaswamy, S., Rastogi, R., & Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. *ACM SIGMOD Record, 29*(2), 427–438. https://doi.org/10.1145/335191.335437

Schaefer, G. (2017). Classifier ensembles for imbalanced classification. *7th International Conference on Innovative Computing Technology, INTECH 2017*, 1–2. https://doi.org/10.1109/INTECH.2017.8102455

Schölkopf, B., Williamson, R., Smola, A., Shawe-Taylor, J., & Piatt, J. (2000). Support vector method for novelty detection. *Advances in Neural Information Processing Systems.*

Sen, S., Dundar, Z., Uncu, O., & Ozen, B. (2021). Potential of Fourier-transform infrared spectroscopy in adulteration detection and quality assessment in buffalo and goat milks. *Microchemical Journal, 166.* https://doi.org/10.1016/j.microc.2021.106207

Shyu, M. L., Chen, S. C., Sarinnapakorn, K., & Chang, L. (2003). A Novel Anomaly Detection Scheme Based on Principal Component Classifier. *3rd IEEE International Conference on Data Mining.*

Silva, N. C. da, Pimentel, M. F., Honorato, R. S., Talhavini, M., Maldaner, A. O., & Honorato, F. A. (2015). Classification of Brazilian and foreign gasolines adulterated with alcohol using infrared spectroscopy. *Forensic Science International, 253.* https://doi.org/10.1016/j.forsciint.2015.05.011

Skoog, D. A., Holler, F. J., & Crouch, S. R. (2017). *Principles of Instrumental Analysis* (7th ed.). Cengage Learning.

Swersky, L., Marques, H. O., Sander, J., Campello, R. J. G. B., & Zimek, A. (2016). On the Evaluation of Outlier Detection and One-Class Classification Methods. *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 1–10. https://doi.org/10.1109/DSAA.2016.8

Thomas, S., Thomas, R., Zachariah, A. K., & Mishra, R. K. (2017). Spectroscopic Methods for Nanomaterials Characterization. In *Spectroscopic Methods for Nanomaterials Characterization* (Vol. 2).

Wauquier, J.-P. (1995). *Petroleum Refining: Crude oil, petroleum products, process flowsheets* (Vol. 1). Institut Francais du Petrole Publication: Petroleum Refining.

Wenzhu, S., Wenting, H., Zufeng, X., & Jianping, C. (2019). Overview of one-class classification. *2019 IEEE 4th International Conference on Signal and Image Processing, ICSIP 2019*. https://doi.org/10.1109/SIPROCESS.2019.8868559

What are Isolation Forests?. Quick Recap for the Best Outlier... | by Nigar Esra Kın | Teknasyon Engineering. (n.d.). Retrieved March 14, 2023, from https://engineering.teknasyon.com/what-are-isolation-forests-151d8e98ef5f

Yang, C., Brown, C. E., Hollebone, B., Yang, Z., Lambert, P., Fieldhouse, B., Landriault, M., & Wang, Z. (2017). Chemical Fingerprints of Crude Oils and Petroleum Products. In *Oil Spill Science and Technology: Second Edition*. https://doi.org/10.1016/B978-0-12-809413-6.00004-7

Yu, H., Wang, X., Shen, F., Long, J., & Du, W. (2022). Novel automatic model construction method for the rapid characterization of petroleum

properties from near-infrared spectroscopy. *Fuel, 316.* https://doi.org/10.1016/j.fuel.2021.123101

Zhao, Y., Nasrullah, Z., & Li, Z. (2019). PyOD: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research, 20.*

Zheng, Y., Zhou, Z., Li, Z., & Li, D. (2021). Kernel Choices in One-class Support Vector Machines for Novelty Detection. *Proceedings - 2021 International Conference on Computers and Automation, CompAuto 2021.* https://doi.org/10.1109/CompAuto54408.2021.00013

Zhu, H., & Xu, J. L. (2020). Authentication and provenance of walnut combining fourier transform mid-infrared spectroscopy with machine learning algorithms. *Molecules, 25*(21). https://doi.org/10.3390/molecules25214987

Appendices

Appendix A: OCSVM OCCs Main Codes

1 Full FTIR

```
from pyod.models.ocsvm import OCSVM

import pandas as pd, numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, RobustScaler, MinMaxScaler, Norma
lizer
from sklearn.metrics import classification_report
import seaborn as sns
from sklearn import metrics
import warnings
warnings.filterwarnings("ignore")

df = pd.read_csv('AllCLASSES.csv')
df = df.rename(columns={'CLASS': 'Class'})
df[df.isna().any(axis=1)]["Class"].value_counts()

sns.countplot(df['Class'])

df[df.isnull().any(axis=1)]
```

1.1 Filling null values with avarage value of the coulmn class based

```
cc = df.columns[0:1868]
for column in cc:
    df[column] = df[column].fillna(df.groupby('Class')[column].transform('mean'))

df[df.isna().any(axis=1)]["Class"].value_counts()

from scipy.signal import savgol_filter

df2 = df.iloc[:, 0:1868]

df2

a = df2[df2 > 100 ].count()

sum(a)

#c = []
#for i in range(5,500,2):
#  df_filtered = df2.apply(savgol_filter,  window_length= i, polyorder=3)
#  b = a - (df_filtered[df_filtered > 100 ].count())
#  c.append({"window_length":i, "diffrence":sum(b)})

#cc = pd.DataFrame(c)

#cc = cc.sort_values(by=['diffrence'], ascending=True)
#cc['diffrence']>0

#cc[cc['diffrence']>0]

df_filtered = df2.apply(savgol_filter,  window_length= 7, polyorder=3)

df_filtered[df_filtered > 100] = 100
```

```
df_filtered[df_filtered > 100 ].count()

df_filtered['Class'] = df['Class'].values

df_filtered[df_filtered["Class"] == 'BASE_OIL']

cols = df.columns[0:1868]
df[cols][df[cols] > 100] = 100
```

1.2 Converting df to array

```
array_ = df_filtered.to_numpy()

np.random.seed(42)
np.random.shuffle(array_)
array_
```

1.3 Scalling usjng Normalizer, proven to give better results

```
array_[:,0:1868] = Normalizer().fit_transform(array_[:,0:1868]) #changed the scaler
array_

training, testing0 = train_test_split(array_, test_size=0.40, random_state=32)

testing, validation = train_test_split(testing0, test_size=0.50, random_state=42)

print(training.shape, testing.shape, validation.shape)
```

1.4 Functions for OCSVM hyperparamerts tunning with K folds CV.

```
def FindPortions(k, y):
    Length = len(y)
    A = []
    L = Length//k
    c = Length - ((Length//k)*k)

    for i in range (0, k):
        A.append(L)
    B = A.copy()

    while(c != 0):
        while (c > k):
            loop = int(c/k)
            for i in range(loop):
                for i in range(0, k):
                    B[i] += 1
                    c -=1
        for i in range(0, c):
            #m = k-i-1
            B[i] += 1
            c -= 1
    Com = []
    for i in range(0,k):
```

```
        Com.append(sum(B[0:i]))
    return B, Com

def Average(lst):
    return sum(lst) / len(lst)

def OCSVclasifier(X_train,X_test, y_test, kk, nn, gg):

        predicted_y = []
        expected_y = []
        classifier = OCSVM(kernel=kk, nu=nn,gamma=gg, degree=3, coef0=0.02) ## kern
el:kk, nu:nn, gamma: gg, uncertian about other parameters.
        classifier.fit(X_train)
        predicted_y.extend(classifier.predict(X_test))
        expected_y.extend(y_test)

        accuracy = metrics.accuracy_score(expected_y, predicted_y)
        F_Score = metrics.f1_score(expected_y, predicted_y)
        Pre = metrics.precision_score(expected_y, predicted_y)
        recl = metrics.recall_score(expected_y, predicted_y)
        roc = metrics.roc_auc_score(expected_y, predicted_y)

        return accuracy, F_Score, Pre, recl, roc

def Main(k,y,test, B, Com, kk, nn, gg):
    j = 0
    FFF = []
    ACC = []
    ppre = []
    recl = []
    a_roc = []
    while(j < k):
        starting = Com[j]
        ending = starting + B[j]
        if (j == k-1):
            ending = starting + B[j]
        #print("starting = ", starting, ",ending = ", ending, "diffrence = ", endin
g-starting)


        listtt = np.arange(starting, ending, 1).tolist()
        X_train = np.delete(y,listtt, 0)
        X_train = X_train[:,0:1868]
        NN = y[starting:ending]
        Testing = np.concatenate((test, NN), axis = 0)
        y_test = Testing[:,1868].astype(int)
        X_test = Testing[:,0:1868]



        accuracy, F_Score, pp, rrr, aroc = OCSVclasifier(X_train,X_test, y_test, kk
, nn, gg)

        FFF.append(F_Score)
        ACC.append(accuracy)
        ppre.append(pp)
        recl.append(rrr)
        a_roc.append(aroc)

        #Thread_accuracy.append(accuracy)

        j += 1
```

```
    return (ACC,FFF, ppre, recl, a_roc)
        #my_queue.put(accuracy)
```

1.5 SETUP

```
nonGasoline = training[training[:,1868] != "GASOLINE"]
Gasoline = training[training[:,1868] == "GASOLINE" ]
nonGasoline[:,1868] = 1
Gasoline[:,1868] = 0 # Target Class. 03/11/2022
G_testing= testing.copy()
G_testing = np.concatenate((G_testing, nonGasoline),axis=0)
np.random.seed(42)
np.random.shuffle(G_testing)
G_testing[:,1868][G_testing[:,1868] != "GASOLINE"] = 1
G_testing[:,1868][G_testing[:,1868] == "GASOLINE"] = 0


#Gasoline_lables = Gasoline[:,1868].astype(int) # all 0s it is just here, not to be
used unless we want to combine it with testing.
#X_test = np.concatenate((tetsing[:,0:1868], Gasoline[:,0:1868]), axis=0)
#y_test = np.concatenate((tetsing[:,1868], Gasoline[:,1868]), axis=0)
#y_test = y_test.astype(int)

nonBASE_OIL = training[training[:,1868] != "BASE_OIL"]
BASE_OIL = training[training[:,1868] == "BASE_OIL" ]
#nonBASE_OIL[:,50] = 1
BASE_OIL[:,1868] = 0
B_testing = testing.copy()
B_testing = np.concatenate((B_testing, nonBASE_OIL),axis=0)

np.random.seed(42)
np.random.shuffle(B_testing)

B_testing[:,1868][B_testing[:,1868] != "BASE_OIL"] = 1
B_testing[:,1868][B_testing[:,1868] == "BASE_OIL"] = 0

nonKEROSENE = training[training[:,1868] != "KEROSENE"]
KEROSENE = training[training[:,1868] == "KEROSENE" ]
#nonKEROSENE[:,50] = 1
KEROSENE[:,1868] = 0
K_testing = testing.copy()
K_testing = np.concatenate((K_testing, nonKEROSENE), axis=0)

np.random.seed(42)
np.random.shuffle(K_testing)

K_testing[:,1868][K_testing[:,1868] != "KEROSENE"] = 1
K_testing[:,1868][K_testing[:,1868] == "KEROSENE"] = 0

nonDIESEL = training[training[:,1868] != "DIESEL"]
DIESEL = training[training[:,1868] == "DIESEL" ]
#nonDIESEL[:,50] = 1
DIESEL[:,1868] = 0
D_testing = testing.copy()
D_testing = np.concatenate((D_testing, nonDIESEL), axis=0)

np.random.seed(42)
np.random.shuffle(D_testing)

D_testing[:,1868][D_testing[:,1868] != "DIESEL"] = 1
D_testing[:,1868][D_testing[:,1868] == "DIESEL"] = 0
```

```
nonFUEL_OIL = training[training[:,1868] != "FUEL_OIL"]
FUEL_OIL = training[training[:,1868] == "FUEL_OIL" ]
#nonFUEL_OIL[:,50] = 1
FUEL_OIL[:,1868] = 0
F_testing = testing.copy()
F_testing = np.concatenate((F_testing, nonFUEL_OIL), axis=0)

np.random.seed(42)
np.random.shuffle(F_testing)

F_testing[:,1868][F_testing[:,1868] != "FUEL_OIL"] = 1
F_testing[:,1868][F_testing[:,1868] == "FUEL_OIL"] = 0

print('Gasoline: ', Gasoline.shape[0], "Gasoline_testing: ", G_testing.shape[0], '\
n',
      "BASE_OIL: ", BASE_OIL.shape[0], "BASE_OIL.testing: ", B_testing.shape[0], '\
n',
      "KEROSENE: ", KEROSENE.shape[0], "KEROSENE.testing: ", K_testing.shape[0], '\
n',
      "DIESEL: ", DIESEL.shape[0], "DIESEL.testing: ", D_testing.shape[0], '\n',
      "FUEL_OIL: ", FUEL_OIL.shape[0], "FUEL_OIL.testing: ", F_testing.shape[0])
```

1.5.1 Parameters GridSearch

```
# Define parameter possibilities as lists
OSVM_kernel = ['rbf']
OSVM_gamma = np.arange(1, 1000, 10).tolist() #['scale', 'auto',0.5, 60, 900]
OSVM_nu = np.arange(0.05, 0.9, 0.05).tolist()

training.shape

array_.shape
```

1.5.2 Classifier 1 (Gasoline)

```
%%time
# to find starting and ending of each fold.
B, Com = FindPortions(k=10, y= Gasoline) #didn't make it to go for more than data s
amples folds, not necessary.


# The scores will go here+987\][poirwy]
results = []

# Nested loops - we need to test for all combinations
for kk in OSVM_kernel:
    for gg in OSVM_gamma:
        for nn in OSVM_nu:

            Accuracy , F1_Socre, ppre, recl, a_roc = Main(10,Gasoline,G_testing, B,
Com, kk, nn, gg) ## the k should be same as it in portionfunction
            # Append current results
            results.append({
                'Accuracy': round(Average(Accuracy), 5),
                'F1_Socre': round(Average(F1_Socre), 5),
                'Precession': round(Average(ppre), 5),
                'Recall': round(Average(recl), 5),
                'roc': round(Average(a_roc), 5),
                'Kernel': kk,
                'Gamma': gg,
                'nu': nn
            })
```

```
# Convert to Pandas DataFrame and sort descendingly by accuracy
G_results = pd.DataFrame(results)
G_results = G_results.sort_values(by=['Accuracy','F1_Socre','Precession','Recall'],
ascending=False)
G_results
```

1.5.3 Classifier 2 (BASE_OIL)

```
%%time
# to find starting and ending of each fold.
B, Com = FindPortions(k=10, y= BASE_OIL) #didn't make to go for more than data samp
les folds, not necessary.

# The scores will go here+987\][poirwy]
results = []

# Nested loops - we need to test for all combinations
for kk in OSVM_kernel:
    for gg in OSVM_gamma:
        for nn in OSVM_nu:

            Accuracy , F1_Socre, ppre, recl, a_roc = Main(10,BASE_OIL,B_testing, B,
Com, kk, nn, gg) ## the k should be same as it in portionfunction
            # Append current results
            results.append({
                'Accuracy': round(Average(Accuracy), 5),
                'F1_Socre': round(Average(F1_Socre), 5),
                'Precession': round(Average(ppre), 5),
                'Recall': round(Average(recl), 5),
                'roc': round(Average(a_roc), 5),
                'Kernel': kk,
                'Gamma': gg,
                'nu': nn
            })

# Convert to Pandas DataFrame and sort descendingly by accuracy
B_results = pd.DataFrame(results)
B_results = B_results.sort_values(by=['Accuracy','F1_Socre','Precession','Recall'],
ascending=False)
B_results
```

1.5.4 Classifier 3 (KEROSENE)

```
%%time
# to find starting and ending of each fold.
B, Com = FindPortions(k=10, y= KEROSENE) #didn't make to go for more than data samp
les folds, not necessary.


# The scores will go here
results = []

# Nested loops - we need to test for all combinations
for kk in OSVM_kernel:
    for gg in OSVM_gamma:
        for nn in OSVM_nu:

            Accuracy , F1_Socre, ppre, recl, a_roc = Main(10,KEROSENE,K_testing, B,
Com, kk, nn, gg) ## the k should be same as it in portionfunction
            # Append current results
            results.append({
                'Accuracy': round(Average(Accuracy), 5),
                'F1_Socre': round(Average(F1_Socre), 5),
```

```
                'Precession': round(Average(ppre), 5),
                'Recall': round(Average(recl), 5),
                'roc': round(Average(a_roc), 5),
                'Kernel': kk,
                'Gamma': gg,
                'nu': nn
            })
# Convert to Pandas DataFrame and sort descendingly by accuracy
K_results = pd.DataFrame(results)
K_results = K_results.sort_values(by=['Accuracy','F1_Socre','Precession','Recall'],
ascending=False)
K_results
```

1.5.5 Classifier 4 (DIESEL)

```
%%time
# to find starting and ending of each fold.
B, Com = FindPortions(k=10, y= DIESEL) #didn't make to go for more than data sample
s folds, not necessary.


# The scores will go here
results = []

# Nested loops - we need to test for all combinations
for kk in OSVM_kernel:
    for nn in OSVM_nu:
        for gg in OSVM_gamma:

            Accuracy , F1_Socre, ppre, recl, a_roc = Main(10,DIESEL,D_testing, B, C
om, kk, nn, gg) ## the k should be same as it in portionfunction
            # Append current results
            results.append({
                'Accuracy': round(Average(Accuracy), 5),
                'F1_Socre': round(Average(F1_Socre), 5),
                'Precession': round(Average(ppre), 5),
                'Recall': round(Average(recl), 5),
                'roc': round(Average(a_roc), 5),
                'Kernel': kk,
                'Gamma': gg,
                'nu': nn
            })
# Convert to Pandas DataFrame and sort descendingly by accuracy
D_results = pd.DataFrame(results)
D_results = D_results.sort_values(by=['roc'], ascending=False)
D_results
```

1.5.6 Classifier 5 (FUEL_OIL)

```
%%time
# to find starting and ending of each fold.
B, Com = FindPortions(k=10, y= FUEL_OIL) #didn't make to go for more than data samp
les folds, not necessary.


# The scores will go here
results = []

# Nested loops - we need to test for all combinations
for kk in OSVM_kernel:
    for nn in OSVM_nu:
        for gg in OSVM_gamma:
```

```
            Accuracy , F1_Socre, ppre, recl, a_roc = Main(10,FUEL_OIL,F_testing, B,
Com, kk, nn, gg) ## the k should be same as it in portionfunction
            # Append current results
            results.append({
                'Accuracy': round(Average(Accuracy), 5),
                'F1_Socre': round(Average(F1_Socre), 5),
                'Precession': round(Average(ppre), 5),
                'Recall': round(Average(recl), 5),
                'roc': round(Average(a_roc), 5),
                'Kernel': kk,
                'Gamma': gg,
                'nu': nn
            })
# Convert to Pandas DataFrame and sort descendingly by accuracy
F_results = pd.DataFrame(results)
F_results = F_results.sort_values(by=['roc'], ascending=False)
F_results
```

1.6 SAVE and TEST

```
#G_results.to_csv("G_results_PYOD_OCSVM.csv")
#B_results.to_csv("B_results_PYOD_OCSVM.csv")
#K_results.to_csv("K_results_PYOD_OCSVM.csv")
D_results.to_csv("D_results_PYOD_OCSVM.csv")
F_results.to_csv("F_results_PYOD_OCSVM.csv")

all_testing = validation.copy()

training = np.concatenate((training, testing), axis = 0)

%%time
#Training:
#Gasoline:
nonGasoline = training[training[:,1868] != "GASOLINE"]
Gasoline = training[training[:,1868] == "GASOLINE" ]
G_train = Gasoline[:,0:1868]
G_classifier = OCSVM(kernel='rbf', nu=0.1,gamma=11, degree=3, coef0=0.02)
G_classifier.fit(G_train)

#BASEOIL:
nonBASE_OIL = training[training[:,1868] != "BASE_OIL"]
BASE_OIL = training[training[:,1868] == "BASE_OIL" ]
B_train = BASE_OIL[:,0:1868]
B_classifier =  OCSVM(kernel='rbf', nu=0.1,gamma=671, degree=3, coef0=0.02)
B_classifier.fit(B_train)

#KEROSENE:
nonKEROSENE = training[training[:,1868] != "KEROSENE"]
KEROSENE = training[training[:,1868] == "KEROSENE" ]
K_train = KEROSENE[:,0:1868]
K_classifier = OCSVM(kernel='rbf', nu=0.1,gamma=991, degree=3, coef0=0.02)
K_classifier.fit(K_train)

#Deisel:
nonDIESEL = training[training[:,1868] != "DIESEL"]
DIESEL = training[training[:,1868] == "DIESEL" ]
D_train = DIESEL[:,0:1868]
D_classifier = OCSVM(kernel='rbf', nu=0.1,gamma=981, degree=3, coef0=0.02)
D_classifier.fit(D_train)

#FUEL_OIL
nonFUEL_OIL = training[training[:,1868] != "FUEL_OIL"]
FUEL_OIL = training[training[:,1868] == "FUEL_OIL" ]
```

```
F_train = FUEL_OIL[:,0:1868]
F_classifier =  OCSVM(kernel='rbf', nu=0.15,gamma=771, degree=3, coef0=0.02)
F_classifier.fit(F_train)

bb = []
it = 0
m = all_testing.shape[0]
for i in range(m):
    G_decision_function = G_classifier.decision_function(all_testing[i:(i+1),0:1868
])
    B_decision_function = B_classifier.decision_function(all_testing[i:(i+1),0:1868
])
    K_decision_function = K_classifier.decision_function(all_testing[i:(i+1),0:1868
])
    D_decision_function = D_classifier.decision_function(all_testing[i:(i+1),0:1868
])
    F_decision_function = F_classifier.decision_function(all_testing[i:(i+1),0:1868
])

    if G_decision_function == B_decision_function == K_decision_function == D_decis
ion_function == F_decision_function:
        bb.append("outlier")
        it += 1
        print(i, "outlier")
    elif G_decision_function == min(G_decision_function, B_decision_function, K_dec
ision_function, D_decision_function, F_decision_function):
        bb.append("GASOLINE")
    elif B_decision_function == min(G_decision_function, B_decision_function, K_dec
ision_function, D_decision_function, F_decision_function):
        bb.append("BASE_OIL")
    elif K_decision_function == min(G_decision_function, B_decision_function, K_dec
ision_function, D_decision_function, F_decision_function):
        bb.append("KEROSENE")
    elif D_decision_function == min(G_decision_function, B_decision_function, K_dec
ision_function, D_decision_function, F_decision_function):
        bb.append("DIESEL")
    elif F_decision_function == min(G_decision_function, B_decision_function, K_dec
ision_function, D_decision_function, F_decision_function):
        bb.append("FUEL_OIL")

y_act = all_testing[:,1868]

report = classification_report(y_act, bb, output_dict=True)
preport = pd.DataFrame(report).transpose()
preport

from sklearn.metrics import roc_auc_score
def roc_auc_score_multiclass(actual_class, pred_class, average = "macro"):

  #creating a set of all the unique classes using the actual class list
    unique_class = set(actual_class)
    roc_auc_dict = {}
    for per_class in unique_class:
        #creating a list of all the classes except the current class
        other_class = [x for x in unique_class if x != per_class]

        #marking the current class as 1 and all other classes as 0
        new_actual_class = [0 if x in other_class else 1 for x in actual_class]
        new_pred_class = [0 if x in other_class else 1 for x in pred_class]

        #using the sklearn metrics method to calculate the roc_auc_score
        roc_auc = roc_auc_score(new_actual_class, new_pred_class, average = average
)
        roc_auc_dict[per_class] = roc_auc
```

```
    return roc_auc_dict

lr_roc_auc_multiclass = roc_auc_score_multiclass(y_act, bb)
pd.DataFrame(lr_roc_auc_multiclass.items(), columns=['Class', 'AUC'])

G_Cen = np.mean(G_train, axis=0)
B_Cen = np.mean(B_train, axis=0)
K_Cen = np.mean(K_train, axis=0)
D_Cen = np.mean(D_train, axis=0)
F_Cen = np.mean(F_train, axis=0)

bb = []
it = 0
m = all_testing.shape[0]
for i in range(m):
    G_prob = G_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
    B_prob = B_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
    K_prob = K_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
    D_prob = D_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
    F_prob = F_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]

    if G_prob == B_prob == K_prob == D_prob == F_prob:
        bb.append("outlier")
        it += 1
        print(i, "outlier")
    elif G_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("GASOLINE")
    elif B_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("BASE_OIL")
    elif K_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("KEROSENE")
    elif D_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("DIESEL")
    elif F_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("FUEL_OIL")

report = classification_report(y_act, bb, output_dict=True)
preport = pd.DataFrame(report).transpose()
y = list(y_act)
predictionnnn_a = pd.DataFrame(data={"prediction": bb, "actual": y})
predictionnnn_a.to_csv("pred vs act.csv")
preport

lr_roc_auc_multiclass = roc_auc_score_multiclass(y_act, bb)
pd.DataFrame(lr_roc_auc_multiclass.items(), columns=['Class', 'AUC'])

from scipy.spatial.distance import hamming, mahalanobis, chebyshev, cdist

bb = []
it = 0
m = all_testing.shape[0]
for i in range(m):
    G_prob = G_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
    B_prob = B_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
    K_prob = K_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
    D_prob = D_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
    F_prob = F_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]

    if G_prob == B_prob == K_prob == D_prob == F_prob:
        G_dis = chebyshev(all_testing[i,0:1868], G_Cen)
        B_dis = chebyshev(all_testing[i,0:1868], B_Cen)
        K_dis = chebyshev(all_testing[i,0:1868], K_Cen)
        D_dis = chebyshev(all_testing[i,0:1868], D_Cen)
        F_dis = chebyshev(all_testing[i,0:1868], F_Cen)
```

```
        if G_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
            bb.append("GASOLINE")
        elif B_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
            bb.append("BASE_OIL")
        elif K_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
            bb.append("KEROSENE")
        elif D_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
            bb.append("DIESEL")
        elif F_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
            bb.append("FUEL_OIL")
        it += 1
        print(i, "outlier")
    elif G_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("GASOLINE")
    elif B_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("BASE_OIL")
    elif K_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("KEROSENE")
    elif D_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("DIESEL")
    elif F_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("FUEL_OIL")

report = classification_report(y_act, bb, output_dict=True)
preport = pd.DataFrame(report).transpose()
preport

lr_roc_auc_multiclass = roc_auc_score_multiclass(y_act, bb)
pd.DataFrame(lr_roc_auc_multiclass.items(), columns=['Class', 'AUC'])

predictionnnn_a = pd.DataFrame(data={"prediction": bb, "actual": y})
predictionnnn_a.to_csv("pred vs act.csv")

G_predict = G_classifier.predict(all_testing[:,0:1868]).reshape(-1,1)
B_predict = B_classifier.predict(all_testing[:,0:1868]).reshape(-1,1)
K_predict = K_classifier.predict(all_testing[:,0:1868]).reshape(-1,1)
D_predict = D_classifier.predict(all_testing[:,0:1868]).reshape(-1,1)
F_predict = F_classifier.predict(all_testing[:,0:1868]).reshape(-1,1)
prediction = np.concatenate([G_predict, B_predict, K_predict, D_predict, F_predict]
,axis=1)

bb = []
it = 0
m = prediction.shape[0]
for i in range(m):
    a = prediction[i,:]
    if a[0] == 0 and a[1] == 1 and a[2] == 1 and a[3] == 1 and a[4] == 1:
        bb.append("GASOLINE")
    elif a[0] == 1 and a[1] == 0 and a[2] == 1 and a[3] == 1 and a[4] == 1:
        bb.append("BASE_OIL")
    elif a[0] == 1 and a[1] == 1 and a[2] == 0 and a[3] == 1 and a[4] == 1:
        bb.append("KEROSENE")
    elif a[0] == 1 and a[1] == 1 and a[2] == 1 and a[3] == 0 and a[4] == 1:
        bb.append("DIESEL")
    elif a[0] == 1 and a[1] == 1 and a[2] == 1 and a[3] == 1 and a[4] == 0:
        bb.append("FUEL_OIL")
    else:
        G_prob = G_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
        B_prob = B_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
        K_prob = K_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
        D_prob = D_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
        F_prob = F_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
        it += 1
        if G_prob == B_prob == K_prob == D_prob == F_prob:
            bb.append("outlier")
            print(i, "outlier")
```

```
        elif G_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("GASOLINE")
            print(i)
        elif B_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("BASE_OIL")
            print(i)
        elif K_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("KEROSENE")
            print(i)
        elif D_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("DIESEL")
            print(i)
        elif F_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("FUEL_OIL")
            print(i)

y_act = all_testing[:,1868]

report = classification_report(y_act, bb, output_dict=True)
preport = pd.DataFrame(report).transpose()
preport

lr_roc_auc_multiclass = roc_auc_score_multiclass(y_act, bb)
pd.DataFrame(lr_roc_auc_multiclass.items(), columns=['Class', 'AUC'])

from scipy.spatial.distance import hamming, mahalanobis, chebyshev, cdist

bb = []
it = 0
m = prediction.shape[0]
for i in range(m):
    a = prediction[i,:]
    if a[0] == 0 and a[1] == 1 and a[2] == 1 and a[3] == 1 and a[4] == 1:
        bb.append("GASOLINE")
    elif a[0] == 1 and a[1] == 0 and a[2] == 1 and a[3] == 1 and a[4] == 1:
        bb.append("BASE_OIL")
    elif a[0] == 1 and a[1] == 1 and a[2] == 0 and a[3] == 1 and a[4] == 1:
        bb.append("KEROSENE")
    elif a[0] == 1 and a[1] == 1 and a[2] == 1 and a[3] == 0 and a[4] == 1:
        bb.append("DIESEL")
    elif a[0] == 1 and a[1] == 1 and a[2] == 1 and a[3] == 1 and a[4] == 0:
        bb.append("FUEL_OIL")
    else:
        G_prob = G_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
        B_prob = B_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
        K_prob = K_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
        D_prob = D_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
        F_prob = F_classifier.predict_proba(all_testing[i:(i+1),0:1868])[0][0]
        it += 1
        if G_prob == B_prob == K_prob == D_prob == F_prob:
            G_dis = mahalanobis(all_testing[i,0:1868], G_Cen)
            B_dis = mahalanobis(all_testing[i,0:1868], B_Cen)
            K_dis = mahalanobis(all_testing[i,0:1868], K_Cen)
            D_dis = mahalanobis(all_testing[i,0:1868], D_Cen)
            F_dis = mahalanobis(all_testing[i,0:1868], F_Cen)
            if G_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
                bb.append("GASOLINE")
            elif B_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
                bb.append("BASE_OIL")
            elif K_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
                bb.append("KEROSENE")
            elif D_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
                bb.append("DIESEL")
            elif F_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
                bb.append("FUEL_OIL")
            print(i, "outlier")
```

```
        elif G_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("GASOLINE")
            print(i)
        elif B_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("BASE_OIL")
            print(i)
        elif K_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("KEROSENE")
            print(i)
        elif D_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("DIESEL")
            print(i)
        elif F_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("FUEL_OIL")
            print(i)

report = classification_report(y_act, bb, output_dict=True)
preport = pd.DataFrame(report).transpose()
preport

lr_roc_auc_multiclass = roc_auc_score_multiclass(y_act, bb)
pd.DataFrame(lr_roc_auc_multiclass.items(), columns=['Class', 'AUC'])
```

2 Reduced FTIR

```
from pyod.models.ocsvm import OCSVM

import pandas as pd, numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, RobustScaler, MinMaxScaler, Norma
lizer
from sklearn.metrics import classification_report
import seaborn as sns
from sklearn import metrics
import warnings
warnings.filterwarnings("ignore")

df = pd.read_csv('AllCLASSES.csv')
df = df.rename(columns={'CLASS': 'Class'})
df[df.isna().any(axis=1)]["Class"].value_counts()

sns.countplot(df['Class'])

df[df.isnull().any(axis=1)]
```

2.1 Filling null values with avarage value of the coulmn class based

```
cc = df.columns[0:1868]
for column in cc:
    df[column] = df[column].fillna(df.groupby('Class')[column].transform('mean'))

df[df.isna().any(axis=1)]["Class"].value_counts()

from scipy.signal import savgol_filter

df2 = df.iloc[:, 0:1868]

df2
```

```
a = df2[df2 > 100 ].count()

sum(a)

#c = []
#for i in range(5,500,2):
#  df_filtered = df2.apply(savgol_filter,  window_length= i, polyorder=3)
#  b = a - (df_filtered[df_filtered > 100 ].count())
#  c.append({"window_length":i, "diffrence":sum(b)})

#cc = pd.DataFrame(c)

#cc = cc.sort_values(by=['diffrence'], ascending=True)
#cc['diffrence']>0

#cc[cc['diffrence']>0]

df_filtered = df2.apply(savgol_filter,  window_length= 7, polyorder=3)

df_filtered[df_filtered > 100] = 100

df_filtered[df_filtered > 100 ].count()

df_filtered['Class'] = df['Class'].values

df_filtered[df_filtered["Class"] == 'BASE_OIL']

cols = df.columns[0:1868]
df[cols][df[cols] > 100] = 100

from keras.layers import Input, Dense
from keras.models import Model
# Extract the features and labels
features = df_filtered.iloc[:,:-1].values
labels = df_filtered.iloc[:,-1].values
# Normalized the features
normalizer = Normalizer()
scaled_features = normalizer.fit_transform(features)
# Define the autoencoder model
inputs = Input(shape=(1868,))
encoded = Dense(100, activation='relu')(inputs)
decoded = Dense(1868, activation='sigmoid')(encoded)
autoencoder = Model(inputs, decoded)
# Compile the model
autoencoder.compile(optimizer='Adam', loss='binary_crossentropy')
# Fit the model on the scaled features
autoencoder.fit(scaled_features, scaled_features, epochs=100, batch_size=32, shuffl
e=True)
# Get the encoding layer
encoder = Model(inputs, encoded)

# Get the reduced features
reduced_features = encoder.predict(scaled_features)
# Convert the reduced features to a dataframe
reduced_features = pd.DataFrame(reduced_features)

# Add the labels to the reduced features dataframe
reduced_features['label'] = labels
```

2.2 Converting df to array

```
array_ = reduced_features.to_numpy()
```

```
np.random.seed(42)
np.random.shuffle(array_)
array_

training, testing0 = train_test_split(array_, test_size=0.40, random_state=32)

testing, validation = train_test_split(testing0, test_size=0.50, random_state=42)

print(training.shape, testing.shape, validation.shape)
```

2.3 Functions for OCSVM hyperparamerts tunning with K folds CV.

```
def FindPortions(k, y):
    Length = len(y)
    A = []
    L = Length//k
    c = Length - ((Length//k)*k)

    for i in range (0, k):
        A.append(L)
    B = A.copy()

    while(c != 0):
        while (c > k):
            loop = int(c/k)
            for i in range(loop):
                for i in range(0, k):
                    B[i] += 1
                    c -=1
        for i in range(0, c):
            #m = k-i-1
            B[i] += 1
            c -= 1
    Com = []
    for i in range(0,k):
        Com.append(sum(B[0:i]))
    return B, Com

def Average(lst):
    return sum(lst) / len(lst)

def OCSVclasifier(X_train,X_test, y_test, kk, nn, gg):

        predicted_y = []
        expected_y = []
        classifier = OCSVM(kernel=kk, nu=nn,gamma=gg, degree=3, coef0=0.02) ## kern
el:kk, nu:nn, gamma: gg, uncertian about other parameters.
        classifier.fit(X_train)
        predicted_y.extend(classifier.predict(X_test))
        expected_y.extend(y_test)

        accuracy = metrics.accuracy_score(expected_y, predicted_y)
        F_Score = metrics.f1_score(expected_y, predicted_y)
        Pre = metrics.precision_score(expected_y, predicted_y)
        recl = metrics.recall_score(expected_y, predicted_y)
        roc = metrics.roc_auc_score(expected_y, predicted_y)

        return accuracy, F_Score, Pre, recl, roc
```

```
def Main(k,y,test, B, Com, kk, nn, gg):
    j = 0
    FFF = []
    ACC = []
    ppre = []
    recl = []
    a_roc = []
    while(j < k):
        starting = Com[j]
        ending = starting + B[j]
        if (j == k-1):
            ending = starting + B[j]
        #print("starting = ", starting, ",ending = ", ending, "diffrence = ", endin
g-starting)


        listtt = np.arange(starting, ending, 1).tolist()
        X_train = np.delete(y,listtt, 0)
        X_train = X_train[:,0:100]
        NN = y[starting:ending]
        Testing = np.concatenate((test, NN), axis = 0)
        y_test = Testing[:,100].astype(int)
        X_test = Testing[:,0:100]



        accuracy, F_Score, pp, rrr, aroc = OCSVclasifier(X_train,X_test, y_test, kk
, nn, gg)

        FFF.append(F_Score)
        ACC.append(accuracy)
        ppre.append(pp)
        recl.append(rrr)
        a_roc.append(aroc)

        #Thread_accuracy.append(accuracy)

        j += 1

    return (ACC,FFF, ppre, recl, a_roc)
        #my_queue.put(accuracy)
```

2.4 SETUP

```
nonGasoline = training[training[:,100] != "GASOLINE"]
Gasoline = training[training[:,100] == "GASOLINE" ]
nonGasoline[:,100] = 1
Gasoline[:,100] = 0 # Target Class. 03/11/2022
G_testing= testing.copy()
G_testing = np.concatenate((G_testing, nonGasoline),axis=0)
np.random.seed(42)
np.random.shuffle(G_testing)
G_testing[:,100][G_testing[:,100] != "GASOLINE"] = 1
G_testing[:,100][G_testing[:,100] == "GASOLINE"] = 0


#Gasoline_lables = Gasoline[:,100].astype(int) # all 0s it is just here, not to be
used unless we want to combine it with testing.
#X_test = np.concatenate((tetsing[:,0:100], Gasoline[:,0:100]), axis=0)
#y_test = np.concatenate((tetsing[:,100], Gasoline[:,100]), axis=0)
#y_test = y_test.astype(int)
```

```
nonBASE_OIL = training[training[:,100] != "BASE_OIL"]
BASE_OIL = training[training[:,100] == "BASE_OIL" ]
#nonBASE_OIL[:,50] = 1
BASE_OIL[:,100] = 0
B_testing = testing.copy()
B_testing = np.concatenate((B_testing, nonBASE_OIL),axis=0)

np.random.seed(42)
np.random.shuffle(B_testing)

B_testing[:,100][B_testing[:,100] != "BASE_OIL"] = 1
B_testing[:,100][B_testing[:,100] == "BASE_OIL"] = 0

nonKEROSENE = training[training[:,100] != "KEROSENE"]
KEROSENE = training[training[:,100] == "KEROSENE" ]
#nonKEROSENE[:,50] = 1
KEROSENE[:,100] = 0
K_testing = testing.copy()
K_testing = np.concatenate((K_testing, nonKEROSENE), axis=0)

np.random.seed(42)
np.random.shuffle(K_testing)

K_testing[:,100][K_testing[:,100] != "KEROSENE"] = 1
K_testing[:,100][K_testing[:,100] == "KEROSENE"] = 0

nonDIESEL = training[training[:,100] != "DIESEL"]
DIESEL = training[training[:,100] == "DIESEL" ]
#nonDIESEL[:,50] = 1
DIESEL[:,100] = 0
D_testing = testing.copy()
D_testing = np.concatenate((D_testing, nonDIESEL), axis=0)

np.random.seed(42)
np.random.shuffle(D_testing)

D_testing[:,100][D_testing[:,100] != "DIESEL"] = 1
D_testing[:,100][D_testing[:,100] == "DIESEL"] = 0

nonFUEL_OIL = training[training[:,100] != "FUEL_OIL"]
FUEL_OIL = training[training[:,100] == "FUEL_OIL" ]
#nonFUEL_OIL[:,50] = 1
FUEL_OIL[:,100] = 0
F_testing = testing.copy()
F_testing = np.concatenate((F_testing, nonFUEL_OIL), axis=0)

np.random.seed(42)
np.random.shuffle(F_testing)

F_testing[:,100][F_testing[:,100] != "FUEL_OIL"] = 1
F_testing[:,100][F_testing[:,100] == "FUEL_OIL"] = 0

print('Gasoline: ', Gasoline.shape[0], "Gasoline_testing: ", G_testing.shape[0], '\
n',
      "BASE_OIL: ", BASE_OIL.shape[0], "BASE_OIL.testing: ", B_testing.shape[0], '\
n',
      "KEROSENE: ", KEROSENE.shape[0], "KEROSENE.testing: ", K_testing.shape[0], '\
n',
      "DIESEL: ", DIESEL.shape[0], "DIESEL.testing: ", D_testing.shape[0], '\n',
      "FUEL_OIL: ", FUEL_OIL.shape[0], "FUEL_OIL.testing: ", F_testing.shape[0])
```

2.4.1 Parameters GridSearch

```
# Define parameter possibilities as lists
OSVM_kernel = ['rbf']
OSVM_gamma = np.arange(1, 1000, 10).tolist() #['scale', 'auto',0.5, 60, 900]
OSVM_nu = np.arange(0.05, 0.9, 0.05).tolist()

training.shape

array_.shape
```

2.4.2 Classifier 1 (Gasoline)

```
%%time
# to find starting and ending of each fold.
B, Com = FindPortions(k=10, y= Gasoline) #didn't make it to go for more than data s
amples folds, not necessary.


# The scores will go here+987\][poirwy]
results = []

# Nested loops - we need to test for all combinations
for kk in OSVM_kernel:
    for gg in OSVM_gamma:
        for nn in OSVM_nu:

            Accuracy , F1_Socre, ppre, recl, a_roc = Main(10,Gasoline,G_testing, B,
Com, kk, nn, gg) ## the k should be same as it in portionfunction
            # Append current results
            results.append({
                'Accuracy': round(Average(Accuracy), 5),
                'F1_Socre': round(Average(F1_Socre), 5),
                'Precession': round(Average(ppre), 5),
                'Recall': round(Average(recl), 5),
                'roc': round(Average(a_roc), 5),
                'Kernel': kk,
                'Gamma': gg,
                'nu': nn
            })

# Convert to Pandas DataFrame and sort descendingly by accuracy
G_results = pd.DataFrame(results)
G_results = G_results.sort_values(by=['Accuracy','F1_Socre','Precession','Recall'],
ascending=False)
G_results
```

2.4.3 Classifier 2 (BASE_OIL)

```
%%time
# to find starting and ending of each fold.
B, Com = FindPortions(k=10, y= BASE_OIL) #didn't make to go for more than data samp
les folds, not necessary.

# The scores will go here+987\][poirwy]
results = []

# Nested loops - we need to test for all combinations
for kk in OSVM_kernel:
    for gg in OSVM_gamma:
        for nn in OSVM_nu:

            Accuracy , F1_Socre, ppre, recl, a_roc = Main(10,BASE_OIL,B_testing, B,
```

```
Com, kk, nn, gg) ## the k should be same as it in portionfunction
            # Append current results
            results.append({
                'Accuracy': round(Average(Accuracy), 5),
                'F1_Socre': round(Average(F1_Socre), 5),
                'Precession': round(Average(ppre), 5),
                'Recall': round(Average(recl), 5),
                'roc': round(Average(a_roc), 5),
                'Kernel': kk,
                'Gamma': gg,
                'nu': nn
            })

# Convert to Pandas DataFrame and sort descendingly by accuracy
B_results = pd.DataFrame(results)
B_results = B_results.sort_values(by=['Accuracy','F1_Socre','Precession','Recall'],
ascending=False)
B_results
```

2.4.4 Classifier 3 (KEROSENE)

```
%%time
# to find starting and ending of each fold.
B, Com = FindPortions(k=10, y= KEROSENE) #didn't make to go for more than data samp
les folds, not necessary.


# The scores will go here
results = []

# Nested loops - we need to test for all combinations
for kk in OSVM_kernel:
    for gg in OSVM_gamma:
        for nn in OSVM_nu:

            Accuracy , F1_Socre, ppre, recl, a_roc = Main(10,KEROSENE,K_testing, B,
Com, kk, nn, gg) ## the k should be same as it in portionfunction
            # Append current results
            results.append({
                'Accuracy': round(Average(Accuracy), 5),
                'F1_Socre': round(Average(F1_Socre), 5),
                'Precession': round(Average(ppre), 5),
                'Recall': round(Average(recl), 5),
                'roc': round(Average(a_roc), 5),
                'Kernel': kk,
                'Gamma': gg,
                'nu': nn
            })
# Convert to Pandas DataFrame and sort descendingly by accuracy
K_results = pd.DataFrame(results)
K_results = K_results.sort_values(by=['Accuracy','F1_Socre','Precession','Recall'],
ascending=False)
K_results
```

2.4.5 Classifier 4 (DIESEL)

```
%%time
# to find starting and ending of each fold.
B, Com = FindPortions(k=10, y= DIESEL) #didn't make to go for more than data sample
s folds, not necessary.


# The scores will go here
```

```
results = []

# Nested loops - we need to test for all combinations
for kk in OSVM_kernel:
    for nn in OSVM_nu:
        for gg in OSVM_gamma:

            Accuracy , F1_Socre, ppre, recl, a_roc = Main(10,DIESEL,D_testing, B, C
om, kk, nn, gg) ## the k should be same as it in portionfunction
            # Append current results
            results.append({
                'Accuracy': round(Average(Accuracy), 5),
                'F1_Socre': round(Average(F1_Socre), 5),
                'Precession': round(Average(ppre), 5),
                'Recall': round(Average(recl), 5),
                'roc': round(Average(a_roc), 5),
                'Kernel': kk,
                'Gamma': gg,
                'nu': nn
            })
# Convert to Pandas DataFrame and sort descendingly by accuracy
D_results = pd.DataFrame(results)
D_results = D_results.sort_values(by=['roc'], ascending=False)
D_results
```

2.4.6 Classifier 5 (FUEL_OIL)

```
%%time
# to find starting and ending of each fold.
B, Com = FindPortions(k=10, y= FUEL_OIL) #didn't make to go for more than data samp
les folds, not necessary.


# The scores will go here
results = []

# Nested loops - we need to test for all combinations
for kk in OSVM_kernel:
    for nn in OSVM_nu:
        for gg in OSVM_gamma:

            Accuracy , F1_Socre, ppre, recl, a_roc = Main(10,FUEL_OIL,F_testing, B,
Com, kk, nn, gg) ## the k should be same as it in portionfunction
            # Append current results
            results.append({
                'Accuracy': round(Average(Accuracy), 5),
                'F1_Socre': round(Average(F1_Socre), 5),
                'Precession': round(Average(ppre), 5),
                'Recall': round(Average(recl), 5),
                'roc': round(Average(a_roc), 5),
                'Kernel': kk,
                'Gamma': gg,
                'nu': nn
            })
# Convert to Pandas DataFrame and sort descendingly by accuracy
F_results = pd.DataFrame(results)
F_results = F_results.sort_values(by=['roc'], ascending=False)
F_results
```

2.5 SAVE and TEST

```
#G_results.to_csv("G_results_PYOD_OCSVM.csv")
#B_results.to_csv("B_results_PYOD_OCSVM.csv")
#K_results.to_csv("K_results_PYOD_OCSVM.csv")
D_results.to_csv("D_results_PYOD_OCSVM.csv")
F_results.to_csv("F_results_PYOD_OCSVM.csv")

all_testing = validation.copy()

training = np.concatenate((training, testing), axis = 0)

%%time
#Training:
#Gasoline:
nonGasoline = training[training[:,100] != "GASOLINE"]
Gasoline = training[training[:,100] == "GASOLINE" ]
G_train = Gasoline[:,0:100]
G_classifier = OCSVM(kernel='rbf', nu=0.05,gamma=1, degree=3, coef0=0.02)
G_classifier.fit(G_train)

#BASEOIL:
nonBASE_OIL = training[training[:,100] != "BASE_OIL"]
BASE_OIL = training[training[:,100] == "BASE_OIL" ]
B_train = BASE_OIL[:,0:100]
B_classifier =  OCSVM(kernel='rbf', nu=0.1,gamma=1, degree=3, coef0=0.02)
B_classifier.fit(B_train)

#KEROSENE:
nonKEROSENE = training[training[:,100] != "KEROSENE"]
KEROSENE = training[training[:,100] == "KEROSENE" ]
K_train = KEROSENE[:,0:100]
K_classifier = OCSVM(kernel='rbf', nu=0.1,gamma=91, degree=3, coef0=0.02)
K_classifier.fit(K_train)

#Deisel:
nonDIESEL = training[training[:,100] != "DIESEL"]
DIESEL = training[training[:,100] == "DIESEL" ]
D_train = DIESEL[:,0:100]
D_classifier = OCSVM(kernel='rbf', nu=0.1,gamma=31, degree=3, coef0=0.02)
D_classifier.fit(D_train)

#FUEL_OIL
nonFUEL_OIL = training[training[:,100] != "FUEL_OIL"]
FUEL_OIL = training[training[:,100] == "FUEL_OIL" ]
F_train = FUEL_OIL[:,0:100]
F_classifier =  OCSVM(kernel='rbf', nu=0.2,gamma=1, degree=3, coef0=0.02)
F_classifier.fit(F_train)

bb = []
it = 0
m = all_testing.shape[0]
for i in range(m):
    G_decision_function = G_classifier.decision_function(all_testing[i:(i+1),0:100]
)
    B_decision_function = B_classifier.decision_function(all_testing[i:(i+1),0:100]
)
    K_decision_function = K_classifier.decision_function(all_testing[i:(i+1),0:100]
)
    D_decision_function = D_classifier.decision_function(all_testing[i:(i+1),0:100]
)
    F_decision_function = F_classifier.decision_function(all_testing[i:(i+1),0:100]
)
```

```
    if G_decision_function == B_decision_function == K_decision_function == D_decis
ion_function == F_decision_function:
        bb.append("outlier")
        it += 1
        print(i, "outlier")
    elif G_decision_function == min(G_decision_function, B_decision_function, K_dec
ision_function, D_decision_function, F_decision_function):
        bb.append("GASOLINE")
    elif B_decision_function == min(G_decision_function, B_decision_function, K_dec
ision_function, D_decision_function, F_decision_function):
        bb.append("BASE_OIL")
    elif K_decision_function == min(G_decision_function, B_decision_function, K_dec
ision_function, D_decision_function, F_decision_function):
        bb.append("KEROSENE")
    elif D_decision_function == min(G_decision_function, B_decision_function, K_dec
ision_function, D_decision_function, F_decision_function):
        bb.append("DIESEL")
    elif F_decision_function == min(G_decision_function, B_decision_function, K_dec
ision_function, D_decision_function, F_decision_function):
        bb.append("FUEL_OIL")

y_act = all_testing[:,100]

report = classification_report(y_act, bb, output_dict=True)
preport = pd.DataFrame(report).transpose()
preport

from sklearn.metrics import roc_auc_score
def roc_auc_score_multiclass(actual_class, pred_class, average = "macro"):

  #creating a set of all the unique classes using the actual class list
    unique_class = set(actual_class)
    roc_auc_dict = {}
    for per_class in unique_class:
        #creating a list of all the classes except the current class
        other_class = [x for x in unique_class if x != per_class]

        #marking the current class as 1 and all other classes as 0
        new_actual_class = [0 if x in other_class else 1 for x in actual_class]
        new_pred_class = [0 if x in other_class else 1 for x in pred_class]

        #using the sklearn metrics method to calculate the roc_auc_score
        roc_auc = roc_auc_score(new_actual_class, new_pred_class, average = average
)
        roc_auc_dict[per_class] = roc_auc

    return roc_auc_dict

lr_roc_auc_multiclass = roc_auc_score_multiclass(y_act, bb)
pd.DataFrame(lr_roc_auc_multiclass.items(), columns=['Class', 'AUC'])

G_Cen = np.mean(G_train, axis=0)
B_Cen = np.mean(B_train, axis=0)
K_Cen = np.mean(K_train, axis=0)
D_Cen = np.mean(D_train, axis=0)
F_Cen = np.mean(F_train, axis=0)

bb = []
it = 0
m = all_testing.shape[0]
for i in range(m):
    G_prob = G_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
    B_prob = B_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
    K_prob = K_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
```

```
    D_prob = D_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
    F_prob = F_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]

    if G_prob == B_prob == K_prob == D_prob == F_prob:
        bb.append("outlier")
        it += 1
        print(i, "outlier")
    elif G_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("GASOLINE")
    elif B_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("BASE_OIL")
    elif K_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("KEROSENE")
    elif D_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("DIESEL")
    elif F_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("FUEL_OIL")

report = classification_report(y_act, bb, output_dict=True)
preport = pd.DataFrame(report).transpose()
y = list(y_act)
predictionnnn_a = pd.DataFrame(data={"prediction": bb, "actual": y})
predictionnnn_a.to_csv("pred vs act.csv")
preport

lr_roc_auc_multiclass = roc_auc_score_multiclass(y_act, bb)
pd.DataFrame(lr_roc_auc_multiclass.items(), columns=['Class', 'AUC'])

from scipy.spatial.distance import hamming, mahalanobis, chebyshev, cdist

bb = []
it = 0
m = all_testing.shape[0]
for i in range(m):
    G_prob = G_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
    B_prob = B_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
    K_prob = K_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
    D_prob = D_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
    F_prob = F_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]

    if G_prob == B_prob == K_prob == D_prob == F_prob:
        G_dis = chebyshev(all_testing[i,0:100], G_Cen)
        B_dis = chebyshev(all_testing[i,0:100], B_Cen)
        K_dis = chebyshev(all_testing[i,0:100], K_Cen)
        D_dis = chebyshev(all_testing[i,0:100], D_Cen)
        F_dis = chebyshev(all_testing[i,0:100], F_Cen)
        if G_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
            bb.append("GASOLINE")
        elif B_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
            bb.append("BASE_OIL")
        elif K_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
            bb.append("KEROSENE")
        elif D_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
            bb.append("DIESEL")
        elif F_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
            bb.append("FUEL_OIL")
        it += 1
        print(i, "outlier")
    elif G_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("GASOLINE")
    elif B_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("BASE_OIL")
    elif K_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
        bb.append("KEROSENE")
    elif D_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
```

```
            bb.append("DIESEL")
        elif F_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("FUEL_OIL")

report = classification_report(y_act, bb, output_dict=True)
preport = pd.DataFrame(report).transpose()
preport

lr_roc_auc_multiclass = roc_auc_score_multiclass(y_act, bb)
pd.DataFrame(lr_roc_auc_multiclass.items(), columns=['Class', 'AUC'])

predictionnnn_a = pd.DataFrame(data={"prediction": bb, "actual": y})
predictionnnn_a.to_csv("pred vs act.csv")

G_predict = G_classifier.predict(all_testing[:,0:100]).reshape(-1,1)
B_predict = B_classifier.predict(all_testing[:,0:100]).reshape(-1,1)
K_predict = K_classifier.predict(all_testing[:,0:100]).reshape(-1,1)
D_predict = D_classifier.predict(all_testing[:,0:100]).reshape(-1,1)
F_predict = F_classifier.predict(all_testing[:,0:100]).reshape(-1,1)
prediction = np.concatenate([G_predict, B_predict, K_predict, D_predict, F_predict]
,axis=1)

bb = []
it = 0
m = prediction.shape[0]
for i in range(m):
    a = prediction[i,:]
    if a[0] == 0 and a[1] == 1 and a[2] == 1 and a[3] == 1 and a[4] == 1:
        bb.append("GASOLINE")
    elif a[0] == 1 and a[1] == 0 and a[2] == 1 and a[3] == 1 and a[4] == 1:
        bb.append("BASE_OIL")
    elif a[0] == 1 and a[1] == 1 and a[2] == 0 and a[3] == 1 and a[4] == 1:
        bb.append("KEROSENE")
    elif a[0] == 1 and a[1] == 1 and a[2] == 1 and a[3] == 0 and a[4] == 1:
        bb.append("DIESEL")
    elif a[0] == 1 and a[1] == 1 and a[2] == 1 and a[3] == 1 and a[4] == 0:
        bb.append("FUEL_OIL")
    else:
        G_prob = G_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
        B_prob = B_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
        K_prob = K_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
        D_prob = D_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
        F_prob = F_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
        it += 1
        if G_prob == B_prob == K_prob == D_prob == F_prob:
            bb.append("outlier")
            print(i, "outlier")
        elif G_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("GASOLINE")
            print(i)
        elif B_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("BASE_OIL")
            print(i)
        elif K_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("KEROSENE")
            print(i)
        elif D_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("DIESEL")
            print(i)
        elif F_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("FUEL_OIL")
            print(i)

y_act = all_testing[:,1868]
```

```
report = classification_report(y_act, bb, output_dict=True)
preport = pd.DataFrame(report).transpose()
preport

lr_roc_auc_multiclass = roc_auc_score_multiclass(y_act, bb)
pd.DataFrame(lr_roc_auc_multiclass.items(), columns=['Class', 'AUC'])

from scipy.spatial.distance import hamming, mahalanobis, chebyshev, cdist

bb = []
it = 0
m = prediction.shape[0]
for i in range(m):
    a = prediction[i,:]
    if a[0] == 0 and a[1] == 1 and a[2] == 1 and a[3] == 1 and a[4] == 1:
        bb.append("GASOLINE")
    elif a[0] == 1 and a[1] == 0 and a[2] == 1 and a[3] == 1 and a[4] == 1:
        bb.append("BASE_OIL")
    elif a[0] == 1 and a[1] == 1 and a[2] == 0 and a[3] == 1 and a[4] == 1:
        bb.append("KEROSENE")
    elif a[0] == 1 and a[1] == 1 and a[2] == 1 and a[3] == 0 and a[4] == 1:
        bb.append("DIESEL")
    elif a[0] == 1 and a[1] == 1 and a[2] == 1 and a[3] == 1 and a[4] == 0:
        bb.append("FUEL_OIL")
    else:
        G_prob = G_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
        B_prob = B_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
        K_prob = K_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
        D_prob = D_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
        F_prob = F_classifier.predict_proba(all_testing[i:(i+1),0:100])[0][0]
        it += 1
        if G_prob == B_prob == K_prob == D_prob == F_prob:
            G_dis = mahalanobis(all_testing[i,0:100], G_Cen)
            B_dis = mahalanobis(all_testing[i,0:100], B_Cen)
            K_dis = mahalanobis(all_testing[i,0:100], K_Cen)
            D_dis = mahalanobis(all_testing[i,0:100], D_Cen)
            F_dis = mahalanobis(all_testing[i,0:100], F_Cen)
            if G_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
                bb.append("GASOLINE")
            elif B_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
                bb.append("BASE_OIL")
            elif K_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
                bb.append("KEROSENE")
            elif D_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
                bb.append("DIESEL")
            elif F_dis == min(G_dis, B_dis, K_dis, D_dis, F_dis):
                bb.append("FUEL_OIL")
            print(i, "outlier")
        elif G_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("GASOLINE")
            print(i)
        elif B_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("BASE_OIL")
            print(i)
        elif K_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("KEROSENE")
            print(i)
        elif D_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("DIESEL")
            print(i)
        elif F_prob == max(G_prob, B_prob, K_prob, D_prob, F_prob):
            bb.append("FUEL_OIL")
            print(i)
```

```
report = classification_report(y_act, bb, output_dict=True)
preport = pd.DataFrame(report).transpose()
preport

lr_roc_auc_multiclass = roc_auc_score_multiclass(y_act, bb)
pd.DataFrame(lr_roc_auc_multiclass.items(), columns=['Class', 'AUC'])
```

Appendix B: Radar Charts R code

```
library(fmsb)
library(ggplot2)

data <- data.frame(Accuracy = c(1, 0, 0.899, 0.229, 0.862, 0.807, 0.936,
0.89),
                   Precision = c(1, 0, 0.909, 0.543, 0.833, 0.819, 0.905, 0.9),
                   F1_Score = c(1, 0,0.919, 0.421, 0.829, 0.835, 0.93,
0.904),
                   AUC_ROC = c(1, 0,0.956, 0.684, 0.914, 0.918, 0.977,
0.96),
                   Recall = c(1, 0, 0.941, 0.534, 0.862, 0.879, 0.969, 0.946),
                   row.names = c("max", "min", "GMM", "OCSVM", "IF",
"PCA", "LOF", "KNN"))

    colors_line <- c(scales::alpha("darkgray", 0.9),
                     scales::alpha("gold", 0.9),
                     scales::alpha("tomato", 0.9),
                     scales::alpha("royalblue", 0.9),
                     scales::alpha("salmon", 0.9),
                     scales::alpha("seagreen", 0.9))



    p <- radarchart(data, seg = 5, pcol = colors_line, lty = 1,  plwd = 3,
cex.axis = 0.7)

    # Add a legend
    legend_data <- rownames(data[-c(1,2),])
```

```
legend_colors <- colors_line
legend(x = 0.75, y = 1.25, legend = legend_data, col = legend_colors,
lty = 1, lwd = 2, cex = 0.5 )

# Save the plot with the legend
ggsave("radarchart.png", plot = p, device = "png", dpi = 300, width
= 8, height = 6)
```

تمكّن هذه الدراسة من إمداد الذكاء الاصطناعي بتأثيرات هامة على صناعة النفط، حيث توفر خططًا متعددة وفعالة لتطبيقه في تصنيف المنتجات النفطية بطريقة قوية ومتينة. تقدّم نتائج هذه الدراسة نظرة قيمة على نقاط القوة والضعف في الأساليب المختلفة لمساعدة الممارسين في اختيار المصنف الأنسب لمشكلتهم المحددة.

الملخص

تشكل تكرير منتجات النفط الخام جانبًا حيويًا في المجتمع الحديث، حيث يتم استخدام هذه المنتجات على نطاق واسع في العديد من الصناعات والقطاعات. ولذلك فإن الإجراء الفعال لتحليل منتجات النفط هو أمر حاسم للغاية، حيث يلعب دورًا حاسمًا في مراقبة الجودة وحماية البيئة والالتزام بمتطلبات السوق المتطورة باستمرار.

هذه الدراسة لديها القدرة على التأثير بشكل كبير على صناعة النفط من خلال التعرف الدقيق على خمسة منتجات بترولية ، بما في ذلك الديزل والكيروسين والبنزين والوقود وزيت اللوب الأساسي. لتحليل المنتجات البترولية من خلال البيانات الإشعاعية للتحويل الفوريي للتحويل (FTIR) ، تنفذ هذه الدراسة منهجية تعلّم الآلة الحديثة (ML) وتحديدًا مجموعة من المصنفات من فئة واحدة (OCCs). يمكن للOCCs التغلب على مشكلة عدم التوازن وتداخل الفئات وضوضاء العينات وتحديد القيم الشاذة. تم مقارنة ستة مصنفات من فئة واحدة ، وكان لكل منها مصنف أساسي مختلف ، وهي دعم أقليّة الفصل الخطيّ الدعمي (OCSVM)، نماذج خليط غاوسيّ (GMM)، غابات العزلة (IF)، تحليل العناصر الرئيسية (PCA)، عامل الخطأ المحلي (LOF)، الجار الأقرب (KNN). تم التحقق من OCCs مقابل الشبكات العصبية الاصطناعية (ANN).

تشير نتائج الدقة إلى أن المجموعة التي تعتمد على نماذج الاختلال الخليطية الغوصية (GMM)حققت أعلى أداء عبر جميع مقاييس التقييم، يليها المجموعة التي تعتمد على آلات الدعم النوعية من فئة واحدة (OCSVM) وعامل الضوضاء الشذوذي المحلي (LOF) بدرجة 0.936 و 0.908 على التوالي، والمجموعة التي تعتمد على غابة العزل (IF) بدرجة 0.899 ومجموعة جار القريبين (KNN) بدرجة 0.871. تفوق كل المجموعات المصنف الشبكي العصبي الاصطناعي (ANN) الذي واجه صعوبة في معالجة بيانات غير متوازنة، وتحقيق دقة بنسبة 0.441.

بعض الأساليب المستخدمة حساسة للبيانات ذات الأبعاد العالية، لذلك تم إجراء التجارب أيضًا على بيانات FTIR التي تم تقليل أبعادها باستخدام Autoencoder محسن. أظهرت النتائج انخفاضًا كبيرًا في أداء الأسلوب المبني علىOCSVM ، وانخفاضًا طفيفًا في حالة الأساليب المبنية على GMM و PCAوIF، وظلت النتائج على حالها في حالة الأساليب المبنية على LOF و KNN.

جامعة البحرين

كلية العلوم



برامج الدراسات العليا

تقنية التعلم الآلي لتحليل طيف الأشعة تحت الحمراء الفورية لتوصيف منتجات البترول

أطروحة مقدمة كجزء من متطلبات الحصول على درجة الماجستير

في علوم وتحليل البيانات الضخمه

إعـــداد

إياد العتوم

الرقم الجامعي: ٢٠١٣٣٢٢٢

إشـــراف

د. رياض قسنطيني
أستاذ مشارك
جامعة البحرين

د. سوسن هلال
أستاذ مساعد
جامعة البحرين

مملكة البحرين

مارس ٢٠٢٣م