Machine Learning For Credit Card Fraud Detection

line 1: İrfan Erdem Şenyener
line 2: *Engineering Faculty Computer Engineering*
line 3: *Dokuz Eylül University*
line 4: İzmir, Türkiye
line 5: irfanerdem.senyener@ogr.deu.edu.tr

line 1: Ali Şiyar Arslan
line 2: *Engineering Faculty Computer Engineering*
line 3: *Dokuz Eylül University*
line 4: İzmir, Türkiye
line 5: alisiyar.arslan@ogr.deu.edu.tr

***Abstract*—This document gives insight into the process of data preprocessing techniques used in this project that aims to create a model to detect the probability of credit card fraud.**

Keywords—categorize, model, data, probability, database, statistics, fraud, information, features, preprocessing.

I. Introduction

This report contains information about the steps taken when data preprocessing, related works and our projects difference from these works, the detailed description of our dataset and our feature selection methods, with the aim to create a model that can detect and categorize credit card fraud with high accuracy and precision. To achieve this goal, many categorisation models were compared with eachoder to find the model with the best results.

II. Literature Studies

A. *A Comprehensive Survey of Data Mining-Based Fraud Detection Research*

"This survey paper categorises, compares, and summarises from almost all published technical and review articles in automated fraud detection within the last 10 years. It defines the professional fraudster, formalises the main types and subtypes of known fraud, and presents the nature of data evidence collected within affected industries. Within the business context of mining the data to achieve higher cost savings, this research presents methods and techniques together with their problems. Compared to all related reviews on fraud detection, this survey covers much more technical articles and is the only one, to the best of our knowledge, which proposes alternative data and solutions from related domains." [1].

B. *The Application of Data Mining Techniques in Financial Fraud Detection: A Classification Framework and an Academic Review of Literature*

"This paper presents a review of — and classification scheme for — the literature on the application of data mining techniques for the detection of financial fraud. Although financial fraud detection (FFD) is an emerging topic of great importance, a comprehensive literature review of the subject has yet to be carried out. This paper thus represents the first systematic, identifiable and comprehensive academic literature review of the data mining techniques that have been applied to FFD." [2].

C. *Financial Fraud Detection Applying Data Mining Techniques: A Comprehensive Review From 2009 to 2019*

"This paper gives a comprehensive revision of the state-of-the-art research in detecting financial fraud from 2009 to 2019 inclusive and classifying them based on their types of fraud and data mining technology utilized in detecting financial fraud. The review result yielded a sample of 75 relevant articles (58 conference papers with 17 peer-reviewed journal articles) that are categorized into four main groups (bank fraud, insurance fraud, financial statement fraud, and cryptocurrency fraud). The study shows that 34 data mining techniques were used to identify fraud throughout various financial applications. The SVM is found to be one of the most widely used financial fraud detection techniques that carry about 23% of the overall study, followed by both Naïve Bayes and Random Forest, resulting in 15%." [3].

D. *Distributed Data Mining in Credit Card Fraud Detection*

"Credit card transactions continue to grow in number, taking an ever-larger share of the US payment system and leading to a higher rate of stolen account numbers and subsequent losses by banks. Improved fraud detection thus has become essential to maintain the viability of the US payment system. Banks have used early fraud warning systems for some years. Large scale data-mining techniques can improve the state of the art in commercial practice. Scalable techniques to analyze massive amounts of transaction data that efficiently compute fraud detectors in a timely manner is an important problem, especially for e-commerce. Besides scalability and efficiency, the fraud-detection task exhibits technical problems that include skewed distributions of training data and nonuniform cost per error, both of which have not been widely studied in the knowledge-discovery and data mining community." [4].

E. *Data Mining Application in Credit Card Fraud Detection System*

"Nationwide, financial losses due to financial statement frauds (FSF) are mounting. The industry recognizes the problem and is just now starting to act. Although prevention is the best way to reduce frauds, fraudsters are adaptive and will usually find ways to circumvent such measures. Detecting fraud is essential once prevention mechanism has failed. Several data mining algorithms have been developed that allow one to extract relevant knowledge from a large amount of data like fraudulent financial statements to detect FSF. Detecting FSF is a new attempt; thus, several research questions have often being asked: (1) Can FSF be detected?

How likely and how to do it? (2) What data features can be used to predict FSF? (3) What kinds of algorithm can be used to detect FSF? (4) How to measure the performance of the detection? And (5) How effective of these algorithms in terms of fraud detection? To help answer these questions, we conduct an extensive review on literatures. We present a generic framework to guide our analysis. Critical issues for FSF detection are identified and discussed." [5].

The difference of our project, from these other projects, is that our project uses commonly used categorisation models rather than using spesific models for every situtations.

III. Description of the Dataset

Our dataset contains information about transactions that happened in two days. There are four types of features in our dataset, which are : V1…V28, time, amount and our target class.

A. *V1…V28*

These features are named this way to safeguard the confidentality of credit card users. They contain values between -1 and 1 with exceptions. These features can be categorised as nominal data because they are not connected to or ordered by eachoder, and have discrete numerical values.

B. *Time*

This feature measures the time passed since the first transaction, in seconds. This feature can be categorized as a continuous interval data because it has order and time is continuous.

C. *Amount*

This feature is the amount that was transacted. This feature can be categorized as discrete or interval data because money is not continuous and our numeric data has order.

D. *Target Feature*

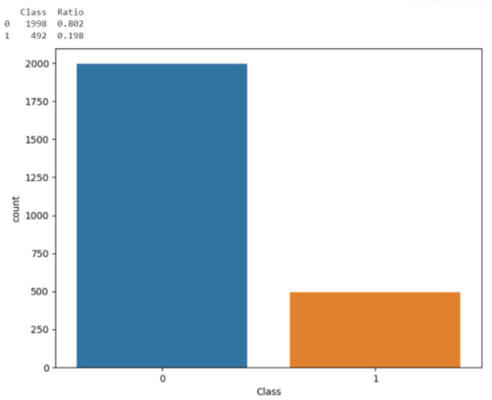
The target value is named "Class" in the data set. This feature is our target feature to determine if our transaction is fraud or not. It can both be categorized as categorical data or binary data because It has only two values it can have, 1 and 0 and It categorizes data.

index	count	mean	std	min	25%	50%	75%	max
Time	2490.0	92403.39959839357	47510.91072724603	51.0	51476.0	83709.0	138019.25	172756.0
V1	2490.0	-0.9216883632114193	3.9560665652964158	-30.552380043581	-1.38572782601905	-0.2823605224261875	1.2427418996827575	2.37085221987414
V2	2490.0	0.7173751619842884	2.7738219793759775	-20.4591497044095	-0.4366518524611295	0.2849409097832535	1.1946526089427927	22.0577289904909
V3	2490.0	-1.4037013148096746	4.414941811561839	-31.1036848245812	-1.6647808260375399	-0.2534532765829255	0.7892795671357611	4.02942831931052
V4	2490.0	0.8838762224382197	2.5154428027867377	-4.31297585255669	-0.6310878325669953	0.33243916345708	1.429397447139465	12.1146718424589
V5	2490.0	-0.6279565986149471	2.8772613888038856	-22.105531524316	-0.9617081874420835	-0.2108382859153335	0.5741870713837924	12.47787472067
V6	2490.0	-0.2966282159084988	1.549377760785178	-8.99836811420914	-0.9917130388083805	-0.4098434967969796	0.2612432117950274	12.0587633474039
V7	2490.0	-1.091611582875663	4.045958216989986	-43.5572415712451	-0.9538207393782512	-0.1116530340605935	0.4992221576995245	16.3340913350001
V8	2490.0	0.12155118472562357	3.183589308716119	-41.0442609210741	-0.20802214431381225	0.0466697351923852	0.48430678173325875	20.0072083651213
V9	2490.0	-0.5133351061643068	1.7880791581511046	-13.4340663182301	-1.075509518488005	-0.2143527712954535	0.456028149929073	6.26368804244595
V10	2490.0	-1.1186217408525987	3.254057298102145	-24.5882624372475	-0.9707637988996103	-0.2244475776693998	0.26834623751173625	11.0941773364143
V11	2490.0	0.7080660031863527	2.1356197087519617	-3.54883863818073	-0.6749733467193683	0.20569007796371	1.2621762423318326	12.0189131816199
V12	2490.0	-1.2591033219076828	3.3349931802258364	-18.6837146333443	-1.1269310976323026	-0.1044240379164725	0.45461589056892976	3.42576939866866
V13	2490.0	-0.04771979708293178	1.0173917167871802	-3.74981604344172	-0.7419540007289288	-0.0376340826312599	0.6368455779621862	3.57498522100317
V14	2490.0	-1.3450480764236563	3.465165977275394	-19.2143254902614	-0.963870489451719	-0.1330979396789155	0.39772268812190576	5.70142730649443
V15	2490.0	0.010043127433183335	0.9307910371811798	-4.49894467676621	-0.5506793374968123	0.08141628214195125	0.6631119158375602	2.73691064643217
V16	2490.0	-0.7904320793999212	2.5074960578406825	-14.1298645174931	-0.9468974358371473	-0.0652146800069329	0.461581027594954	3.59487468342043
V17	2490.0	-1.3034689304351297	4.131308827313711	-25.1627993693248	-0.7176360988056455	-0.166770301746001	0.362688188853831	6.73938438478335
V18	2490.0	-0.4496055119271561	1.7303161855772786	-9.49874592104677	-0.7572120323033767	-0.11558162557422799	0.4182153111299875	3.79031621184375
V19	2490.0	0.1324455135916901	1.0286452563264425	-3.68190355226504	-0.447136422551002	0.05172997243741829	0.6138327729679455	5.2283417900513
V20	2490.0	0.070294573038112	0.864892554849109	-7.97160572659173	-0.211758743113453	-0.04240179086281574	0.20450642834871424	11.0590042933942
V21	2490.0	0.14763217626215214	1.847750393124327	-22.7976039065519	-0.216293272008818	0.0037051468339863	0.29190907063414673	27.2028391573154
V22	2490.0	-0.0061617309611673565	0.9258945079204841	-8.88701714094871	-0.564168795191628	-0.006309996891072599	0.538069917559792	6.36198519168435
V23	2490.0	-0.017170307595752003	0.886721109889362	-19.2543276173719	-0.173540357113333	-0.013676882994104 35	0.16842933857471448	6.99152859401764
V24	2490.0	-0.008223039513759299	0.5967377892573984	-2.14427290116573	-0.36593183672130875	0.035193731194132896	0.4206678643334662	3.95701282112281
V25	2490.0	0.005417293676176529	0.5998287178537112	-4.78160552206407	-0.324549343758526	0.010532693604006101	0.3537891627962165	2.60967812174146
V26	2490.0	0.008325139781685998	0.4715186217325084	-1.31184461009895	-0.3069976850426055	-0.0388759716563044 5	0.26036511417258323	2.7452606721766
V27	2490.0	0.02336206275871349	0.7222406187597032	-7.26348214633855	-0.06908056226980015	0.01688362397691915	0.18560757581795576	3.05235768679424
V28	2490.0	0.01380749538372362	0.3475667521631957	-2.92585506309395	-0.0529226994521483	0.017207179727116502	0.11296046926778725	4.74779192901984
Amount	2490.0	95.78038955823293	269.029032396219	0.0	2.9125	19.73	81.7025	4350.0
Class	2490.0	0.19759036144578312	0.39826135909995	0.0	0.0	0.0	0.0	1.0

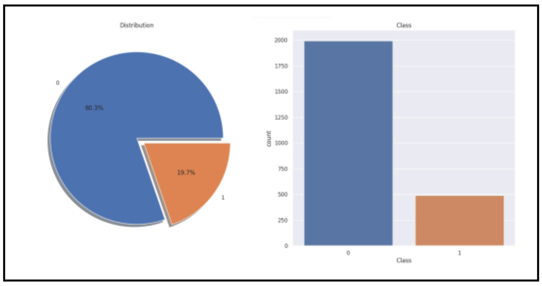
IV. Data Preprocessing Techniques

The dataset that was have chosen to test the model was noisy and had low connections between its features. As a results, many data preprocessing methods had to be used. They are, in order:

- Only 492 of the 284000 records in the data set correspond to the fraud class, which means that the ratio of all transactions is 0.172 fraud.
- To prevent this imbalance, the undersampling technique was first applied. From 284000 records, 2000 non-fraud data were sampled and 492 were concatenated with fraud transactions.



- Numerical values were analyzed to find out the noisiness of the features.
- Since there were only integer variables in the data set, no encoding process was applied except for the class feature.
- The distribution of features was examined using graphical representation techniques such as histograms and barplots



- When performing Outlier analysis, threshold values were selected as 0.05 and 0.95 The reason for this is not to interfere too much with data integrity.
- Multivariate Outlier Analysis was performed using Local Outlier Factor.

- Since the point where the slope changed the most was the 10th point, data with values lower than this point were deleted.
-

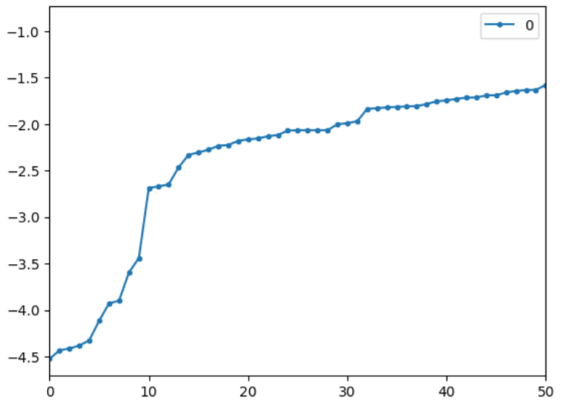
```
# Calculate the first and third quartiles
quartile1 = dataframe[col_name].quantile(q1)
quartile3 = dataframe[col_name].quantile(q3)

# Calculate the interquartile range (IQR)
interquartile = quartile3 - quartile1

# Calculate the upper and lower thresholds for outliers using 1.5 times the IQR
up = quartile3 + 1.5 * interquartile
low = quartile1 - 1.5 * interquartile
```

```
# Replace values below the lower threshold with the lower threshold
dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit

# Replace values above the upper threshold with the upper threshold
dataframe.loc[(dataframe[variable] > up_limit), variable] = up_limit
```




```
# The data with threshold value 4 was chosen because it had the greatest change in slope.
th = np.sort(df_scores)[4]
```

- There were no missing values in the dataset, because of that random values were turned into missing values, median was used to fill them. (Median was used because the data had too many outliners, which made mean values unusable.).

```
def assign_random_nulls(df, num_nulls):

    col_set = set()

    while len(col_set) < 5:
        col_set.add(np.random.randint(0,df.shape[1]))

    #for _ in range(num_nulls):
    while df.isnull().sum().sum() < num_nulls:
        # Choose a random row and column index
        row_index = np.random.randint(0, df.shape[0])

        col_index = np.random.choice(list(col_set))

        # Assign null value to the selected cell
        df.iat[row_index, col_index] = np.nan

# Assign n random null values to the DataFrame
num_nulls = 400 # The number of null values can be changed here.
assign_random_nulls(df, num_nulls)
```

```
na_cols = missing_values_table(df,True)
```

	n_miss	ratio
V16	95	3.821
V19	87	3.500
V28	81	3.258
V8	71	2.856
V23	66	2.655

```
for col in na_cols:
    # Null values in the dataset are filled with median.
    df[col] = df[col].fillna(df[col].median())
```

- New features were created by combining features with other features, using mathemathical calculations, such as NEW_V1*V2 feature which is the multiplication of V1 by V2.

- Since the names of the features were not given, meaning could not be derived, the features were randomly multiplied and added as a new variable.

```
"NEW_V1*V2" = "V1" * "V2"
"NEW_V3*V4" = "V3" * "V4"
"NEW_V5*V6" = "V5" * "V6"
"NEW_V7*V8*V9" = "V7" * "V8" * "V9"
"NEW_V10*V11" = "V10" * "V11"
"NEW_V12*V13*V14" = "V12" * "V13" * "V14"
"NEW_V15*V16" = "V15" * "V16"
"NEW_V17*V18" = "V17" * "V18"
"NEW_V19*V20*V21" = "V19" * "V20" * "V21"
"NEW_V22*V23" = "V22" * "V23"
"NEW_V24*V25*V26" = "V24" * "V25" * "V26"
"NEW_V27*V28" = "V27" * "V28"
"NEW_V2*V3" = "V2" * "V3"
"NEW_V4*V5*V6" = "V4" * "V5" * "V6"
"NEW_V7*V8" = "V7" * "V8"
"NEW_V9*V10*V11" = "V9" * "V10" * "V11"
"NEW_V12*V13" = "V12" * "V13"
"NEW_V14*V15*V16" = "V14" * "V15" * "V16"
"NEW_V17*V18*V19" = "V17" * "V18" * "V19"
```

NEW_V1*V2	NEW_V3*V4	NEW_V5*V6	NEW_V7*V8*V9
-0.939	0.113	-0.024	-0.321
-2.307	-18.873	6.578	14.470
-2.781	2.673	0.797	0.061
0.658	-23.934	-3.858	-3.387
-0.379	1.510	0.036	-0.066

- Normalization of the data was done using Robust Scaler, because regular normalization methods would give us bad results because of the high outliner count
- Using the SMOTE technique Oversampling was applied.

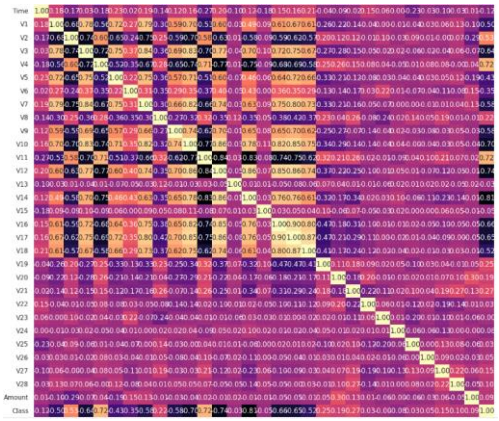
```
# Smote application (Applied to the training set)

oversample = SMOTE(random_state = 17)#
X_smote, y_smote = oversample.fit_resample(X, y)
```

```
# Number of classes of the training set after smote
y_smote.value_counts()
```

```
1    1996
0    1996
Name: Class, dtype: int64
```

- 1500 new fraud data were generated using 492 fraud data, thus preventing imbalance.
- Smote Oversampling had to be used because the dataset was still too imbalanced. This method made the dataset %50 %50 fraud and not fraud.
- When correlation analysis was performed on the data, some significant correlation was observed between two features.



Correlation Matrix

- The data was split into %10 test and %90 training data and also n-folds (n = 10) to use for cross-validation.

V. Apllied Machine Learning Methods For Classification

Four different models were used in this project, all of them from the Sckit-Learn Library of Python programming language.

A. Support Vector Machine – SVC

Support Vectors Machines are machine learning algorithms that are used for classification and regression tasks. It's aim is to separate data into different classes. It does so by trying to find an optimal hyperplane that can separate maximum amount of data into different classes. In our 2D dataset, that hyperplane is a line that separates data.

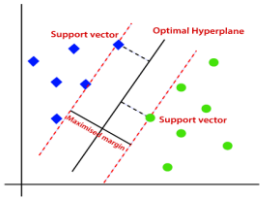


Figure 1. SVM Example

To find this hyperplane, the algorithm uses Support Vectors, which are datapoints that have the maximum margin values. Margin values are determined by the distance the closest datapoint of each class has to the optimal hyperplane. The algorithm is complete when it finds the maximum margin.

For complex datasets like ours, where we have to find a non-linear hyperspace, we need to map our datapoints into a higher dimensional space. Kernels are used for this purpose.

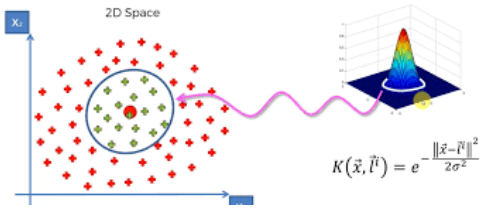


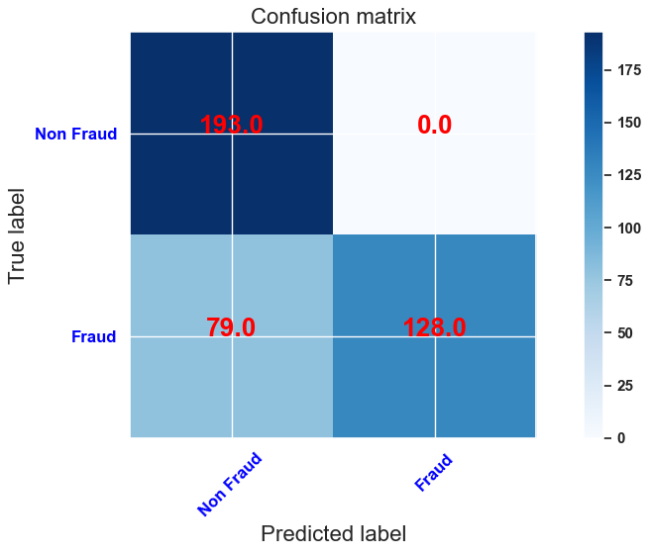
Figure 2. Gaussian Radial Basis Function (RBF) kernel

The kernel that is used uses the formula on Figure 2 to find the distance of datapoints from a selected landmark and then to map them into a higher dimensional space, like a cone. The new optimal hyperplane is then calculated using this higher dimensional space.

Python library scklit-learn's SVM library was used for this model.

First of all, using the hold-out technique, the data set was divided into 90% train and 10% test, and the SVC model was trained on this. The precision value was 1, the recall value was 0.62, 0.76, and the accuracy value was 0.80.

	precision	recall	f1-score	support
0	0.71	1.00	0.83	193
1	1.00	0.62	0.76	207
accuracy			0.80	400
macro avg	0.85	0.81	0.80	400
weighted avg	0.86	0.80	0.80	400



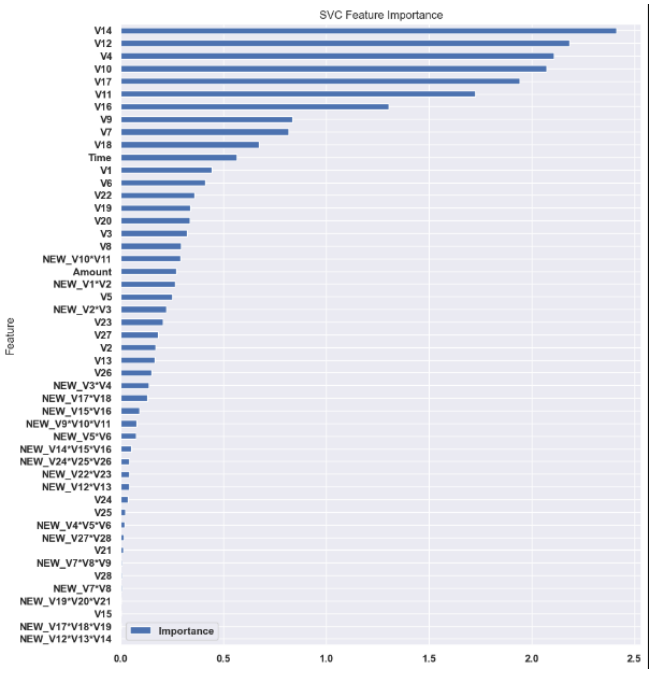
In the last step, 10-fold cross validation was performed. Accuracy: 0.81, F1-Score: 0.77, Roc-Auc: 0.94, Precison: 0.99, Recall: 0.63 values were obtained.

In the next step, hyperparameter optimization was performed.

svc_params = {'C': np.arange(1,10),
'gamma': [1, 5,'scale','auto'],
'kernel': ['linear', 'rbf']}

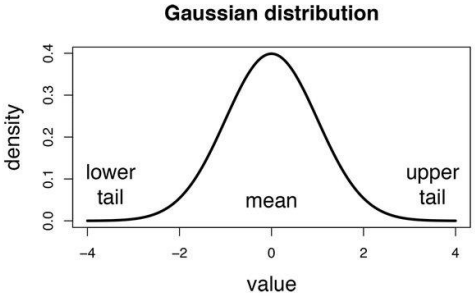
As a result of 10-fold GridSearch using the svc_params hyperparameter list, the optimum hyperparameters were obtained as {'C': 9, 'gamma': 1, 'kernel': 'linear'} and the final model was established with these parameters. In the last case, results of Accuracy: 0.95, F1-Score: 0.95, Roc-Auc: 0.99, Precison: 0.98, Recall: 0.93 were obtained.

When the features were examined by removing the Feature Importance graphics, it was observed that the V14, V12, V4, V10 features contributed well to the model.



B. Naïve-Bayes – GaussianNB

Naïve-Bayes is a Bayesian model used for probabilistic classification machine learning algorithms.



$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$

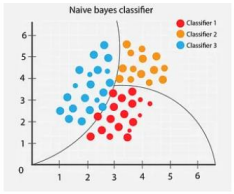


Figure 3. Naive-Bayes Classification

As seen in the formula in Figure 3., the bayesian algorithm uses prior knowledge for classification. Naïve part represents Naïve Assumption, which is the assumption that independent from one another. For this reason, its best use is in datasets that have low correlation.

Gaussian Naïve-Bayes, the one used in the project, has a different assumption. It assumes the datapoints follow a Gaussian distribution like in Figure 4.. To learn, it not only takes the prior information like in regular Naïve-Bayes method, it also calculates the z-score and the standart deviation of the datapoint. The formula used is in Figure 5.

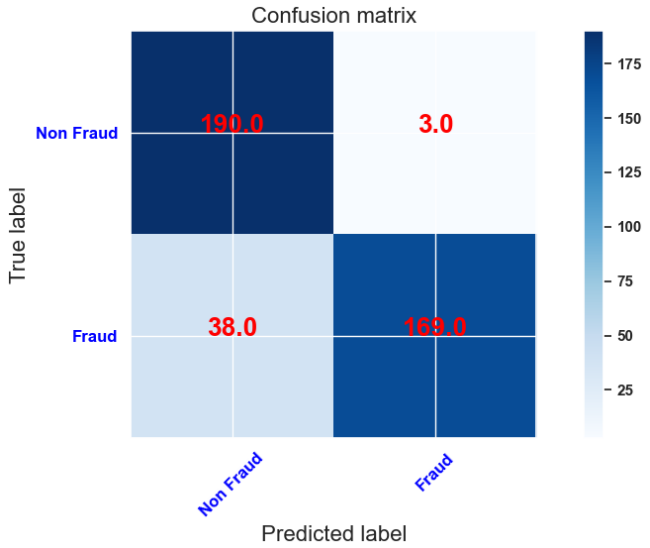
$$p(x_i|y_j) = \frac{1}{\sqrt{2\pi\sigma_j^2}} e^{-\frac{(x_i-\mu_j)^2}{2\sigma_j^2}}$$

Figure 5. Gaussian Naïve-Bayes Algorithm

Python library scklit-learn GaussianNB was used for this model.

First of all, using the hold-out technique, the data set was divided into 90% train and 10% test, and the naive Bayes model was trained on this. The precision value was 0.98, the recall value was 0.82, 0.89, and the accuracy value was 0.90.

	precision	recall	f1-score	support
0	0.83	0.98	0.90	193
1	0.98	0.82	0.89	207
accuracy			0.90	400
macro avg	0.91	0.90	0.90	400
weighted avg	0.91	0.90	0.90	400



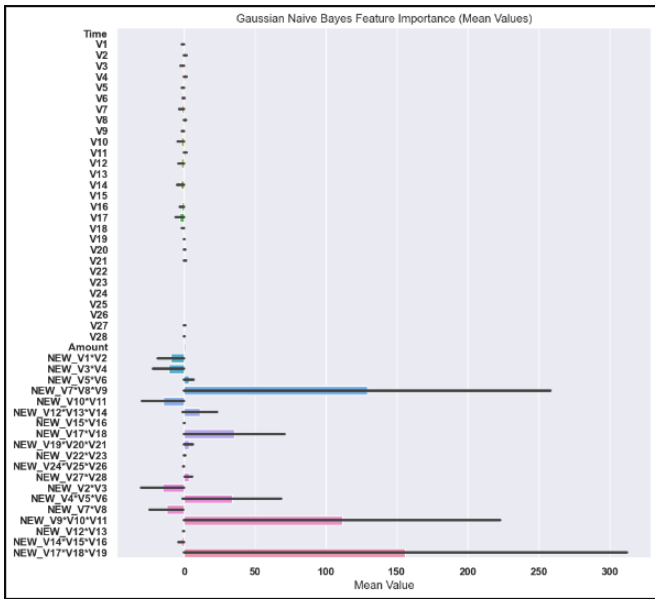
In the next step, 10-fold cross validation was performed. Accuracy: 0.90 F1-Score: 0.90 Roc-Auc: 0.95 Precison: 0.97 Recall: 0.83 values were obtained.

In the last step, hyperparameter optimization was performed.

nb_params = {'var_smoothing': [1e-09, 1e-08, 1e-07, 1e-06, 1e-05] }

As a result of 10-fold GridSearch using the nb_params hyperparameter list, the optimum hyperparameters were obtained as {'var_smoothing': 1e-09} and the final model was established with these parameters. In the last case, results of Accuracy: 0.90, F1-Score: 0.90, Roc-Auc: 0.95, Precison: 0.97 Recall: 0.83 were obtained.

When the features were examined by removing the Feature Importance graphics, it was observed that the NEW_V7*V8*V9, NEW_V17*V18*V19, , NEW_V9*V10*V11 features contributed well to the model.



C. Regression - Logistical Regression

Regression is a learning model, designed to predict a numerical output based on the input space.

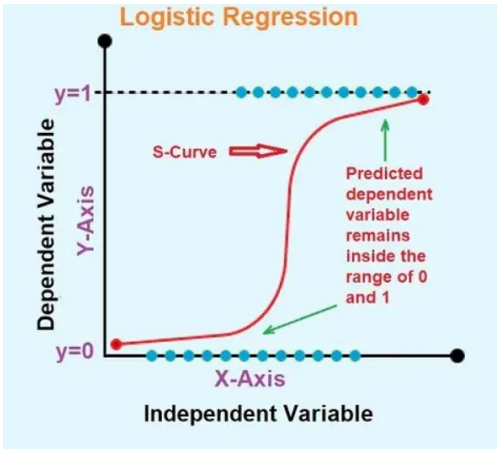


Figure 6. Logistic Regression

The objective of this learning model is to map the input datapoint into a continuous output data. The way it learns data is dependent on the chosen method and most common regression models are not classification models.

Logistic Regression (Figure 6.) was used in the project. This type of regression is mainly used for binary classification tasks. If calculates the probability of an instance belonging to a class. It works with a threshold system where, for classes 0 and 1, any probability bigger or equal to 0.5 is assigned to class 1 and rest are assigned to class 0. Objective of this model is to find the maximum likelihood estimation, which calculates the likelihood of the data, by finding the parameters that do so.

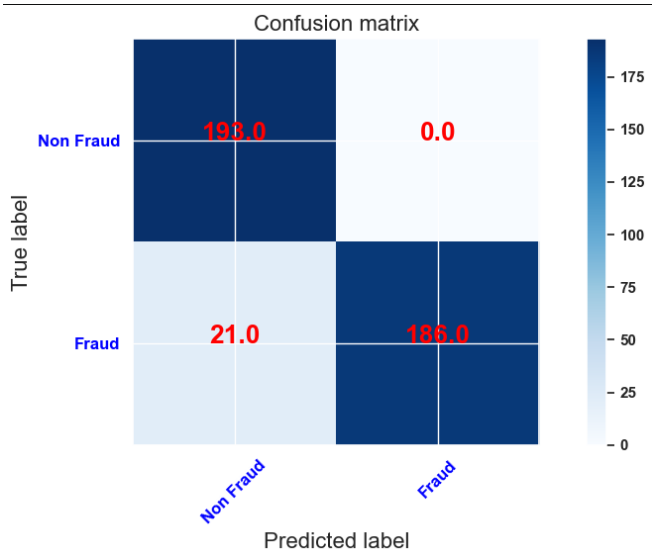
$$\frac{1}{1+e^{-x}}$$

Figure 7. Sigmoid (Logistic) Function

Python library Sckit-learn LogisticRegression was used for this model.

First, using the hold-out technique, the data set was divided into 90% train and 10% test, and the logistic regression model was trained on this. The precision value was 1.00, the recall value was 0.90, 0.95, and the accuracy value was 0.95.

	precision	recall	f1-score	support
0	0.90	1.00	0.95	193
1	1.00	0.90	0.95	207
accuracy			0.95	400
macro avg	0.95	0.95	0.95	400
weighted avg	0.95	0.95	0.95	400



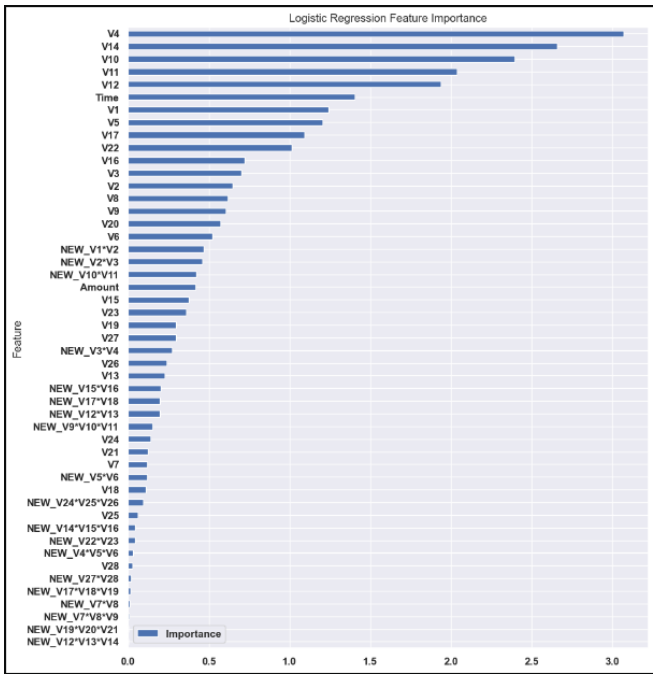
In the next step, 10-fold cross validation was performed. Accuracy: 0.95, F1-Score: 0.95, Roc-Auc: 0.99, Precison: 0.97, Recall: 0.92 values were obtained.

In the last step, hyperparameter optimization was performed.

```
logistic_regression_params = {
    'C': [0.001, 0.01, 0.1, 1, 10, 20 ,30 ,50], # Different values for regularization strength
    'max_iter': [50, 100, 200 , 250 ,300 ,400, 500], # Maximum number of iterations
    'solver': ['lbfgs', 'liblinear'], # Solver for optimization
}
```

As a result of 10-fold GridSearch using the logistic_regression_params hyperparameter list, the optimum hyperparameters were obtained as {'C': 10, 'max_iter': 250, 'solver': 'lbfgs'} and the final model was established with these parameters. In the last case, results of Accuracy: 0.95, F1-Score: 0.95, Roc-Auc: 0.99, Precison: 0.97, Recall: 0.93 were obtained.

When the features were examined by removing the Feature Importance graphics, it was observed that the V4, V14, V10, V11 features contributed well to the model.



D. Neural Networks – MLPClassifier

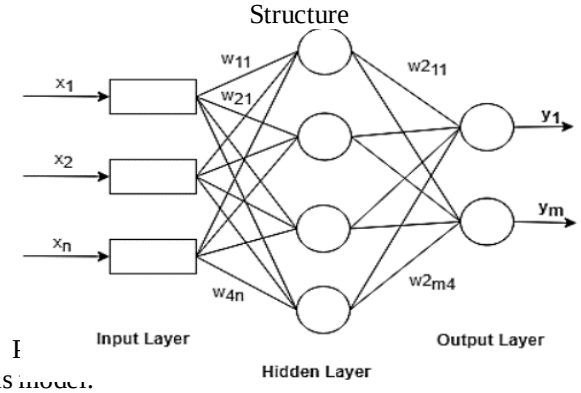
Multi-Layer Perceptron is a type of neural network classifier. It consists of three layers; input layer, hidden layer and output layer.

- Input layer consists of nodes that correspond to each feature of the input data. These nodes then connect to the hidden layer of the model.
- Hidden layer can consist of many node chains, each chain having a weight between nodes. Certain activation functions are used in this layer to introduce non-linearity into the model. Relu is the activation function used in this project, which assigns every negative number to 0.
- Output layer is the last layer of the model and determines the class of the input datapoint using the values it acquired from the hidden layer.

The model learns by a process called backpropagation. A loss value is calculated every time the model predicts, and then the prediction is compared with the real value. The object of the model is to minimize the loss value by

changing the weights and other parameters to achieve this goal.

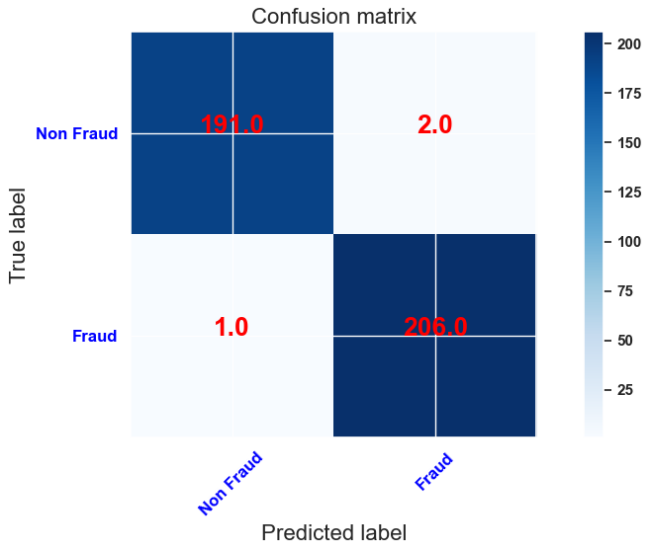
Figure 8. Structure



I for this model.

First, using the hold-out technique, the data set was divided into 90% train and 10% test, and the logistic regression model was trained on this. The precision value was 0.99, the recall value was 1.00, 0.99, and the accuracy value was 0.99.

	precision	recall	f1-score	support
0	0.99	0.99	0.99	193
1	0.99	1.00	0.99	207
accuracy			0.99	400
macro avg	0.99	0.99	0.99	400
weighted avg	0.99	0.99	0.99	400



In the next step, 10-fold cross validation was performed. Accuracy: 0.98 F1-Score: 0.98 Roc-Auc: 1.00 Precison: 0.98 Recall: 0.98 values were obtained.

In the last step, hyperparameter optimization was performed.

mlpc_params = {"alpha": [0.02, 0.005, 0.001,],
"hidden_layer_sizes": [(50,), (100,)],
"solver" : ["lbfgs","adam","sgd"],
"activation": ["relu", "logistic"]}

Optimal hyperparameters as a result of 10-fold GridSearch using mlpc_params hyperparameter list {'activation': 'relu', 'alpha': 0.005,
'hidden_layer_sizes': (100,), 'solver': 'adam'} and the final model was built with these parameters. In the last case, the results of Accuracy: 0.98 F1-Score: 0.98 Roc-Auc: 1.00 Precison: 0.98 Recall: 0.99 were obtained.

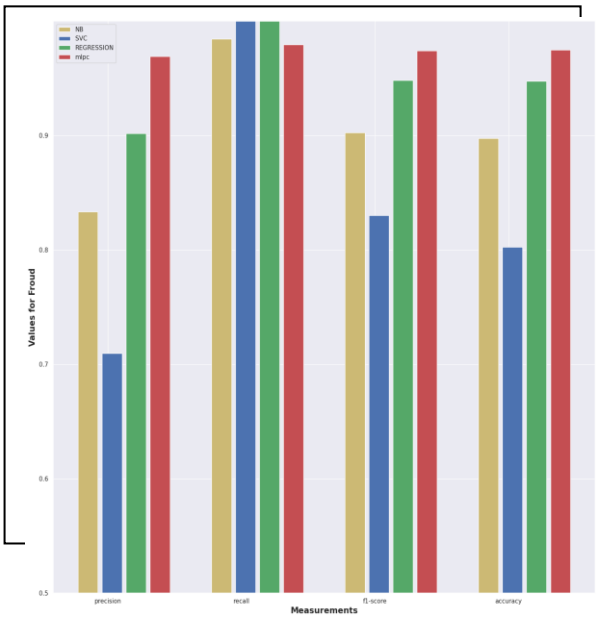
VI. Results of the Experiments

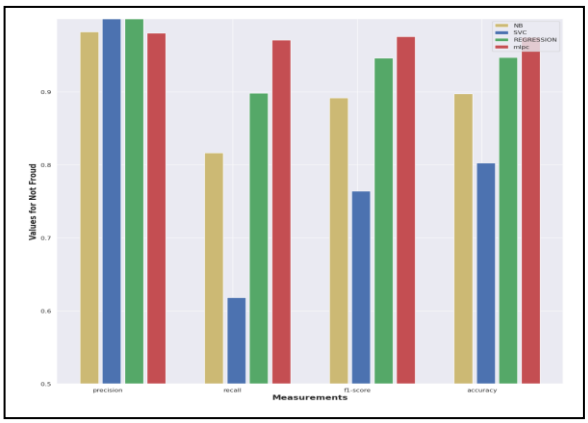
For our classification project, the most important evaluation metrics are accuracy, precision, f1-score and recall values.

There are two testing methods in this project, one with 90%-10%/train-test split and n-fold (n=10) cross-validation.

A. 90%-10%/Train-Test Split

We have two separate values for if the class is Fraud or Not Fraud





As can be seen, svc has 1.0 precision when it is Fraud and 1.0 Recall when it is Not Fraud. This means that it is not fit for this database. Mlpc on the other hand, has the best overall results across all measurements. Logistic Regression and NB models also have acceptable results.

These observations are supported by these confusion matrixes for every model

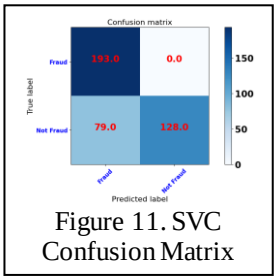


Figure 11. SVC Confusion Matrix

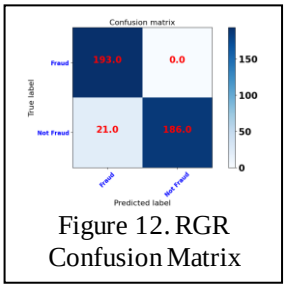


Figure 12. RGR Confusion Matrix

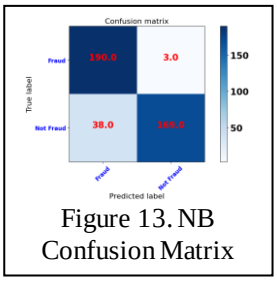


Figure 13. NB Confusion Matrix

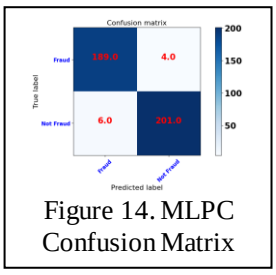


Figure 14. MLPC Confusion Matrix

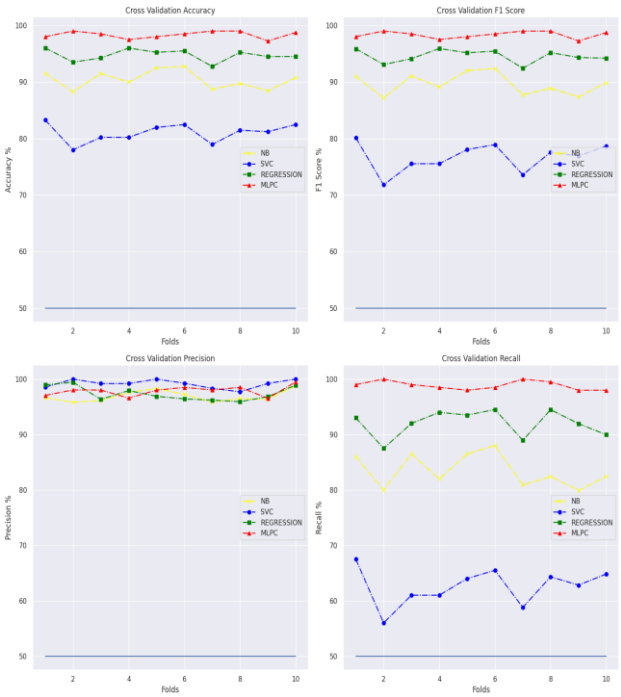


Figure 15. Performance Measurement for Cross-Validation

B. *N-Fold (N=10)*

For cross-validation, we have two separate line graphs for each model, one with original assigned parameters and one with the best parameters assigned using GridSearchCV function of sklearn library. This function fits the model with different parameters and finds the best results.

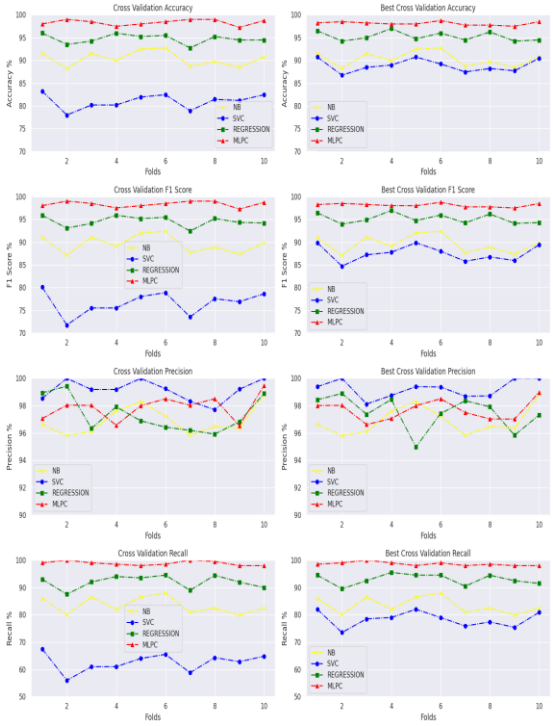


Figure 16. Comparing the Performances of Original Parameters vs Best Parameters

As can be seen, the selection of parameters effects svc the most, while making others slightly better. These figures (Figure 15 and Figure 16) confirm our observations about every model except svc, where it is possible for it to be acceptable given the best parametres.

ACKNOWLEDGMENT

"The dataset has been collected and analysed during a research collaboration of Worldline and the Machine Learning Group (http://mlg.ulb.ac.be/) of ULB (Université Libre de Bruxelles) on big data mining and fraud detection. More details on current and past projects on related topics are available on https://www.researchgate.net/project/Fraud-detection-5 and the page of the DefeatFraud project " [6][7][8]

REFERENCES

[1] Phua, Clifton, et al. "A comprehensive survey of data mining-based fraud detection research." arXiv preprint arXiv:1009.6119 (2010).

[2] Ngai, Eric WT, et al. "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature." Decision support systems 50.3 (2011): 559-569.I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[3] Al-Hashedi, Khaled Gubran, and Pritheega Magalingam. "Financial fraud detection applying data mining techniques: A comprehensive review from 2009 to 2019." Computer Science Review 40 (2021): 100402.

[4] Chan, Philip K., et al. "Distributed data mining in credit card fraud detection." IEEE Intelligent Systems and Their Applications 14.6 (1999): 67-74.

[5] Ogwueleka, Francisca Nonyelum. "Data mining application in credit card fraud detection system." Journal of Engineering Science and Technology 6.3 (2011): 311-322.

[6] Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Undersampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015

[7] Dal Pozzolo, Andrea; Caelen, Olivier; Le Borgne, Yann-Ael; Waterschoot, Serge; Bontempi, Gianluca. Learned lessons in credit card fraud detection from a practitioner perspective, Expert systems with applications,41,10,4915-4928,2014, Pergamon

[8] Dal Pozzolo, Andrea; Boracchi, Giacomo; Caelen, Olivier; Alippi, Cesare; Bontempi, Gianluca. Credit card fraud detection: a realistic.