Tunis-Manar University

Final year project

Machine learning for insurence fraud detection

Wael Chafai
Youssef Azzouz

supervised by
Dr. Faouzi Moussa

June 8, 2018

Contents

1 Abstract

Insurance fraud detection is a challenging problem, given the variety of fraud patterns and relatively small ratio of known frauds in typical samples. While building detection models, the savings from loss prevention needs to be balanced with cost of false alerts. Machine learning techniques allow for improving predictive accuracy, enabling loss control units to achieve higher coverage with low false positive rates. In this paper, multiple machine learning techniques for fraud detection are presented with their performance . The impact of feature engineering, feature selection and parameter tweaking are explored with the objective of achieving superior predictive performance.

2 Introdution

Insurance frauds cover the range of improper activities which an individual may commit in order to achieve a favorable outcome from the insurance company. This could range from staging the incident, misrepresenting the situation including the relevant actors and the cause of incident and finally the extent of damage caused.
Potential situations could include :

- Covering-up for a situation that wasnt covered under insurance (e.g. drunk driving, performing risky acts, illegal activities etc.)
- Misrepresenting the context of the incident: This could include transferring the blame to incidents where the insured party is to blame, failure to take agreed upon safety measures
- Infating the impact of the incident: Increasing the estimate of loss incurred either through addition of unrelated losses (faking losses) or attributing increased cost to the losses

The insurance industry has grappled with the challenge of insurance claim fraud from the very start. On one hand, there is the challenge of impact to customer satisfaction through delayed payouts or prolonged investigation during a period of stress. Additionally, there are costs of investigation and pressure from insurance industry regulators. On the other hand, improper payouts cause a hit to profitability and encourage similar delinquent behavior from other policy holders. According to FBI, the insurance industry in the USA consists of over 7000 companies that collectively received over $1 trillion annually in premiums. FBI also estimates the total cost of insurance fraud (non-health insurance) to be more than $40 billion annually[1] . It must be noted that insurance fraud is not a victimless crime the losses due to frauds, impact all the involved parties through increased premium costs, trust deficit during the claims process and impacts to process efficiency and innovation. Hence the insurance industry has an urgent need to develop capability that can help identify potential frauds with a high degree of accuracy, so that other claims can be cleared rapidly while identified cases can be scrutinized in detail.

[1]Source: https://www.fbi.gov/stats-services/publications/insurance-fraud

3 Why Machine Learning in Fraud Detection?

The traditional approach for fraud detection is based on developing heuristics around fraud indicators. Based on these heuristics, a decision on fraud would be made in one of two ways. In certain scenarios rules would be framed that would define if the case needs to be sent for investigation. In other cases, a checklist would be prepared with scores for the various indicators of fraud. An aggregation of these scores along with the value of the claim would determine if the case needs to be sent for investigation. The criteria for determining indicators and the thresholds will be tested statistically and periodically recalibrated. The challenge with the above approaches is that they rely very heavily on manual intervention which will lead to the following limitations

- Constrained to operate with a limited set of known parameters based on heuristic knowledge while being aware that some of the other attributes could also infiuence decisions
- Inability to understand context-specific relationships between parameters (geography, customer segment, insurance sales process) that might not refiect the typical picture. Consultations with that can help identify potential frauds with a high degree of accuracy, so that other claims can be cleared rapidly while identified cases can be scrutinized in detail. industry experts indicate that there is no typical model, and hence challenges to determine the model specific to context
- Recalibration of model is a manual exercise that has to be conducted periodically to refiect changing behavior and to ensure that the model adapts to feedback from investigations. The ability to conduct this calibration is challenging
- Incidence of fraud (as a percentage of the overall claims) is lowtypically less than 1% of the claims are classified. Additionally new modus operandi for fraud needs to be uncovered on a proactive basis

These are challenging from a traditional statistics perspective. Hence, insurers have started looking at leveraging machine learning capability. The intent is to present a variety of data to the algorithm without judgement around the relevance of the data elements. Based on identified frauds, the intent is for the machine to develop a model that can be tested on these known frauds through a variety of algorithmic techniques.

4 Foundations

The machine learning plays a significant role in fraud detection. This Chapteris an Introduction to the domain of machine learningbefore the author is moving tothe Contribution Chapter. Firstly, the target ofthemain topic is to clarify supervised and unsupervised learning methods. Secondly, it will disclose the cross-validationand bias vs.variance problems. TheChapter2.4will cover training, testing, validation and model selection. TheChapter2.5will introduce the issue of unbalanced data and a method to balance it.

4.1 Supervised and Unsupervised Learning

4.1.1 Supervised learning

Supervised learning is the most frequent type of machine learning problem. Data analysts classify supervised learning problems as "regression" and "classification" problems . The supervised machine learning is the task of inferring a function from labelled training data. Supervised machine learning method means that learning algorithm uses supervised and labelled data in which every example consists of the input object and the output value. Typically the input object is a vector,and the output value is a signal, for instance , binary {0 ; 1}. The researcher gives the dataset to the algorithm in which the correct answers algorithm provides as an output.

4.1.2 Unsupervised learning

We define unsupervised learning as problems where the researcher provides the data without the desired output. These algorithms are used to organise data clusters. Many companies, online shops and others have a lot of data in their databases, such as customer information for instance. We can look at this set of customer information and automatically detect the market segments. Unsupervised learning allows analysts to approach problems with little or no idea of what to expect as the resulting outcome.

5 Evaluation of Classification

Classification will help to understand better which model is more suitable to predict that this metric will count the number of mistakes made. The binary class labels in the training set can take on only two possible values that mostly refer to as positive or negative. The positive and negative instances that a classifier predicts correctly are called true positives TP and true negatives TN.The incorrectly classifie dinstances are called false positives FP and false negative FN. Based on that True Positive Rate, True Negative Rate, False Positive Rate, False Negative Rate, Precision and Accuracy concepts will occur.

$$TPR = \frac{TP}{TP + FN} \tag{1}$$

$$TNR = \frac{TN}{FP + TN} \tag{2}$$

$$FPR = \frac{FP}{FP + TN} \tag{3}$$

$$FNR = \frac{FN}{TP + FN} \tag{4}$$

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

6 Data mining

In this Chapter, We will investigate the dataset that is used in experiments. Datamining is the process of analysing data from different aspects and summarising it into valuable information -information that can be used to increase the revenue or to cut costs, or for both.

6.1 Dataset

The dataset dimension is 15420 rows and 34 columns , the FraudFoundP column is a label that shows if the claim is fraudulent or not .

Table 1: Dataset class statistic

Class	Rows	Percentage
Negative(normal)	14497	94.0%
Positive(fraudent)	923	6%

6.2 Features normalization and visualization

Now , let's have a look at our dataset columns .
AccidentArea : object
AddressChangeClaim : object
Age : int64
AgeOfPolicyHolder : object
AgeOfVehicle : object
AgentType : object
BasePolicy : object
DayOfWeek : object
DayOfWeekClaimed : object
DaysPolicyAccident : object
DaysPolicyClaim : object
Deductible : int64
DriverRating : int64
Fault : object
FraudFoundP : int64
Make : object
MaritalStatus : object
Month : object
MonthClaimed : object
NumberOfCars : object
NumberOfSuppliments : object
PastNumberOfClaims : object
PoliceReportFiled : object
PolicyNumber : int64
PolicyType : object
RepNumber : int64

Sex : object
VehicleCategory : object
VehiclePrice : object
WeekOfMonth : int64
WeekOfMonthClaimed : int64
WitnessPresent : object
Year : int64

Our dataset contains numirical and categorical features . So we are going to encode our categorical features before fitting our models . We do that using the CategoricalEncoder from scikitlearn python library .

Code:

```
le = preprocessing.LabelEncoder()
for i in range(33):
if data.iloc[:,i].dtypes == object:
le.fit(data.iloc[:,i].astype(str))
data.iloc[:,i] = le.transform((data.iloc[:,i]).astype(str))
else:
pass
```

Now we have features but what does they mean or actually how much do we need to know about these features The answer is that we do not need to know meaning of these features however in order to imagine in our mind we should know something like variance, standard deviation, number of sample (count) or max min values. These type of information helps to understand about what is going on data.

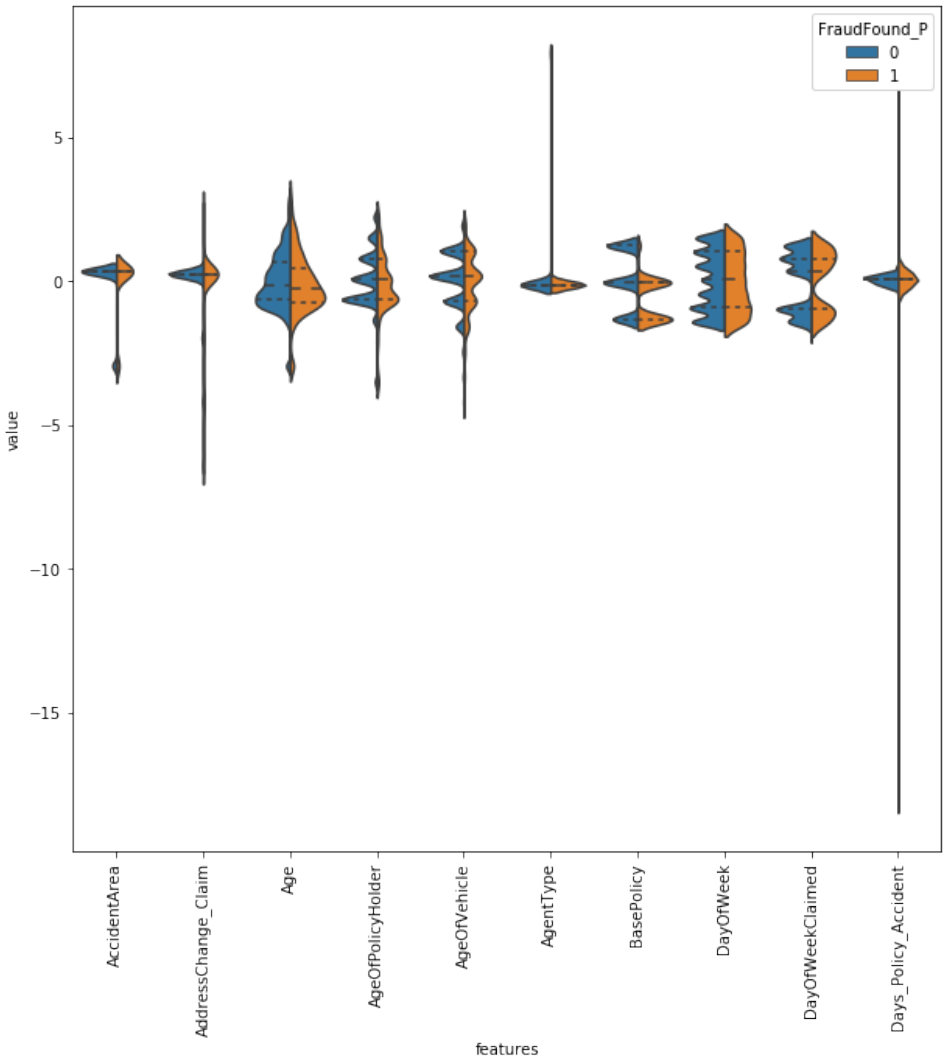
	AccidentArea	AddressChange_Claim	Age	AgeOfPolicyHolder	AgeOfVehicle	AgentType	BasePolicy	DayOfWeek	DayOfWeekClaimed
count	15420.000000	15420.000000	15420.000000	15420.000000	15420.000000	15420.000000	15420.000000	15420.000000	15420.000000
mean	0.896368	2.888521	39.855707	4.891894	4.799287	0.015629	1.036316	2.915759	4.202789
std	0.304792	0.451081	13.492377	1.395907	1.140206	0.124039	0.782355	2.055485	2.273944
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	3.000000	31.000000	4.000000	4.000000	0.000000	0.000000	1.000000	2.000000
50%	1.000000	3.000000	38.000000	5.000000	5.000000	0.000000	1.000000	3.000000	5.000000
75%	1.000000	3.000000	48.000000	6.000000	6.000000	0.000000	2.000000	5.000000	6.000000
max	1.000000	4.000000	80.000000	8.000000	7.000000	1.000000	2.000000	6.000000	7.000000

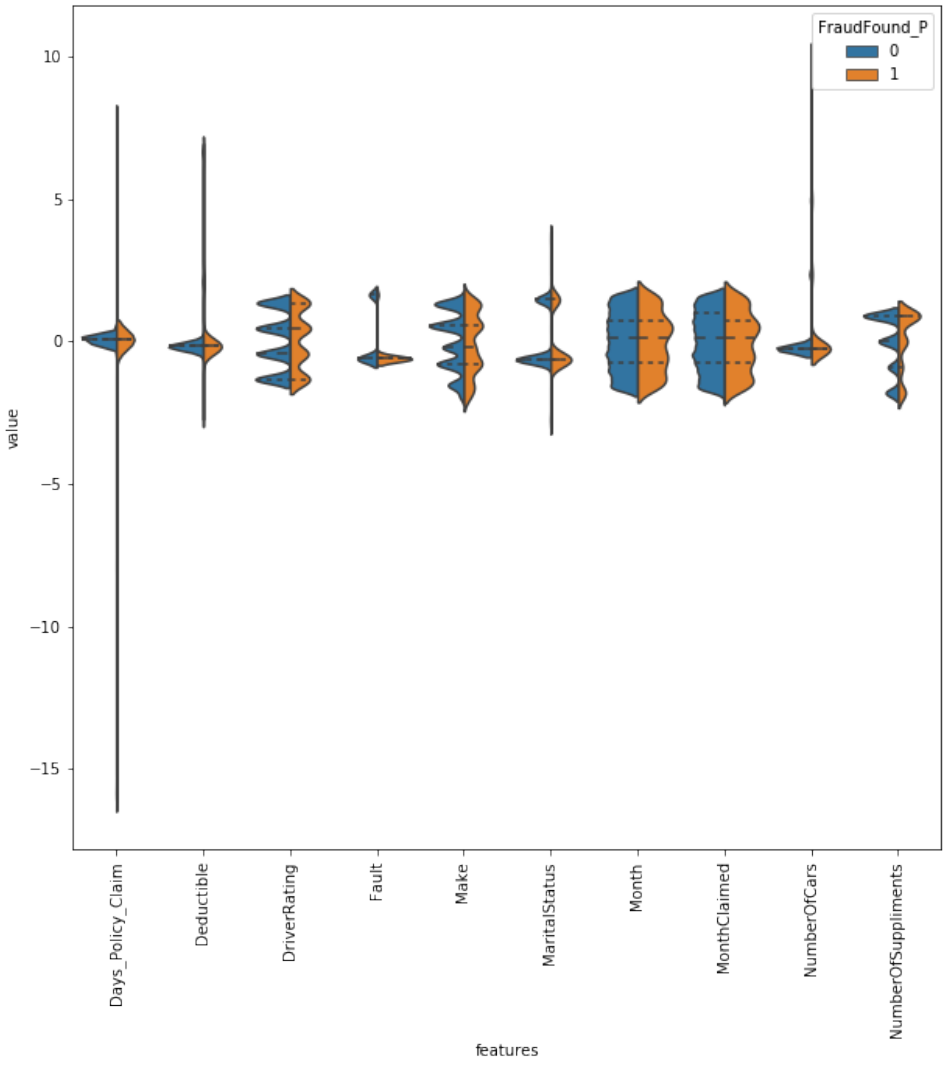
For example , the AccidentArea feature's max value is 1 and Age feature's max is 80 . Therfore we need standarization before visualization, feature selection and feature extraction .

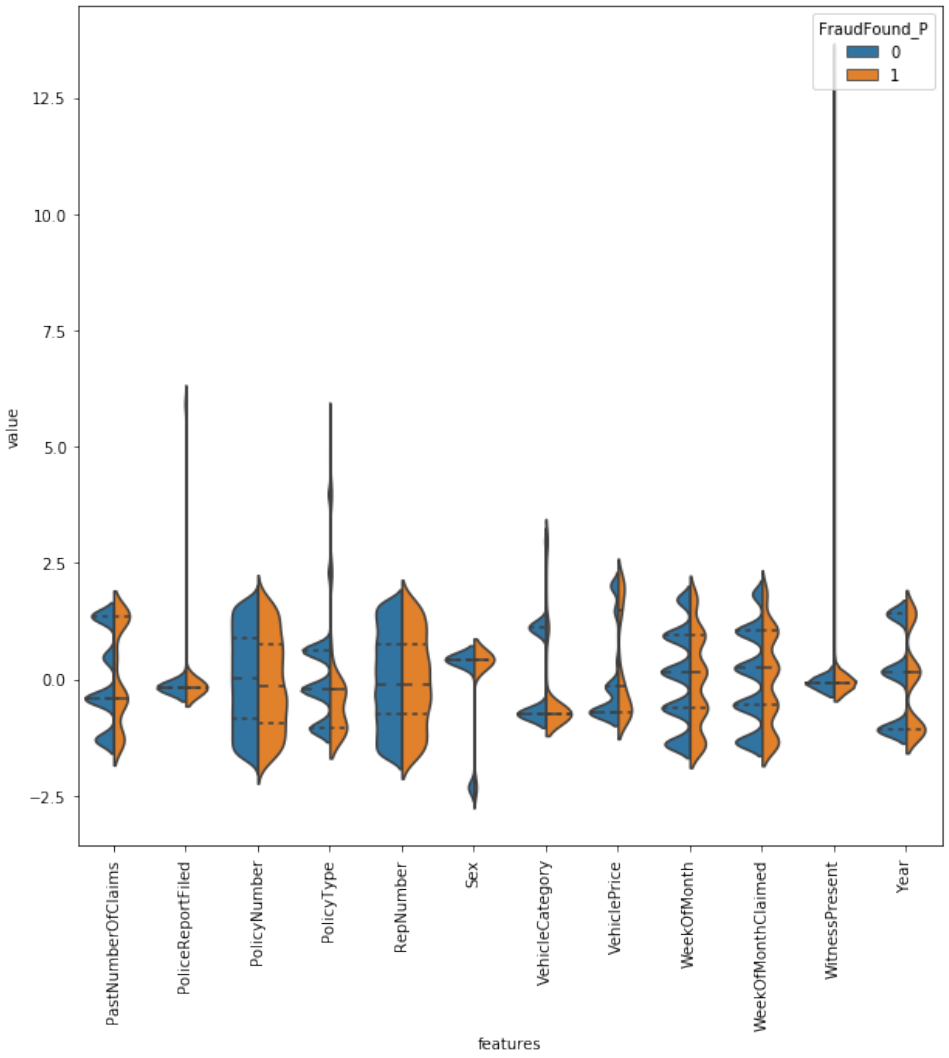
code :

```
data2 = (data - data.mean()) / (data.std()) #standarization
```

With the purpose of having an intion idea about how a feature is separating the fraudent claims of the normal ones , we did some vizualisation (the code for these vizualisations can be found in appendix A) .


FraudFound_P
0
1
5
0
−5
−10
−15
value
AccidentArea
AddressChange_Claim
Age
AgeOfPolicyHolder
AgeOfVehicle
AgentType
BasePolicy
DayOfWeek
DayOfWeekClaimed
Days_Policy_Accident
features


FraudFound_P
0
1
10
5
0
−5
−10
−15
value
Days_Policy_Claim
Deductible
DriverRating
Fault
Make
MaritalStatus
Month
MonthClaimed
NumberOfCars
NumberOfSuppliments
features



By analazing the images above we could for example interpret that , in PolicyType feature , median of the fraudent and normal looks like seperated so it can be good for the classification .

7 Experiments with learning algorithms

In this Chapter we will implement 5 different algorithms: Logistic Regression, Decision Tree , K-Nearest Neighbour , naive bayse and neural network and we will apply these algorithms on the same dataset. In the end , we will compare the results to make the conclusion. we use scikit python library for the experiments .

7.1 Logistic Regression

In this Chapter , a logistic regression model(also called the logit model)shall be set to predict whether a claim is fraud or not . Moreover, we will introduce the notion of classification , the cost function for logistic regression, sigmoid function and gradient.

7.1.1 Preliminaries

Logistic regression is a method for classifying data into discrete outcomes,for example , $\{0 ; 1\}$. In the logistic regression model, the log odds of the outcome is modelled as a linear combination of the predictor variables.The logistic regression hypothesis is defined as:

$$h_\theta(x) = g(\theta^T x) \tag{7}$$

Function g denotes the sigmoid function. The sigmoid function is defined as:

$$g(z) = \frac{1}{1 + e^{-z}} \tag{8}$$

Where z is :

$$z = \theta^T x \tag{9}$$

For a matrix, hypothesis function should perform the sigmoid function one very element .

The cost function for logistic regression looks like:

$$J(\theta) = \frac{1}{m}\Sigma_{i=1}^{m}[-y^{(i)} \log{(h_\theta(x^{(i)})} - (1 - y^{(i)}) \log{(1 - h_\theta(x^{(i)}))} \tag{10}$$

The gradient of the cost function is a vector of the same length as vector θ where the j^{th} element is defined as

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\Sigma_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} where j \geq 0 \tag{11}$$

Cost function and gradientthe investigator should call at the same time.This gradient looks identical to the linear regression gradient,but the formula differs because linear and logistic regressions have different definitions of $h_\theta(x)$

7.1.2 Experiments

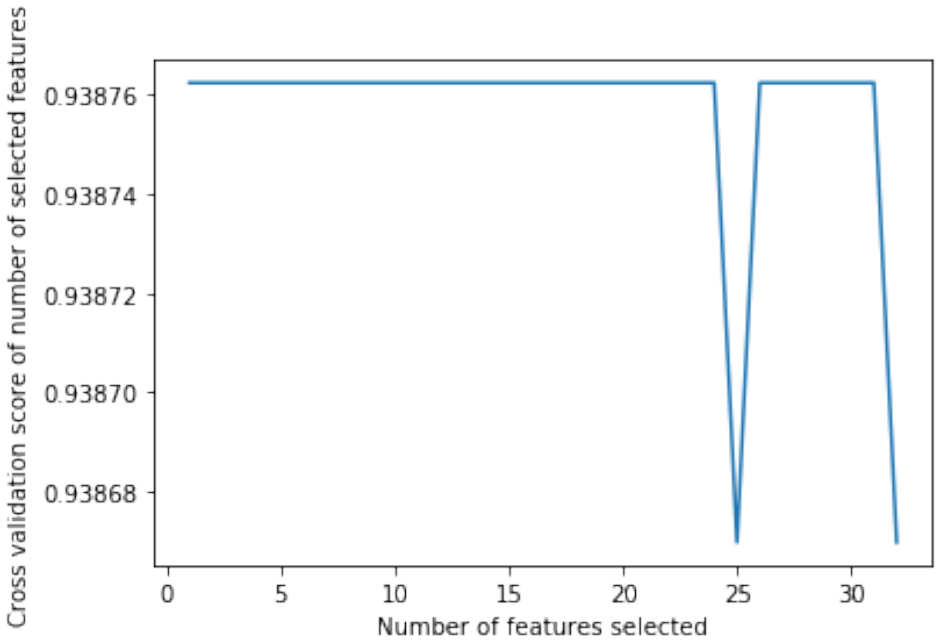
As it was mentioned on page 8 python programming language with the scikitlearn library is being used for the experiments , the code can be found in Appendix B .

First , we upload our data from the mongoDB and we make a projection because we dont need the id column using the import_data function (the code is availble in Appendix C)

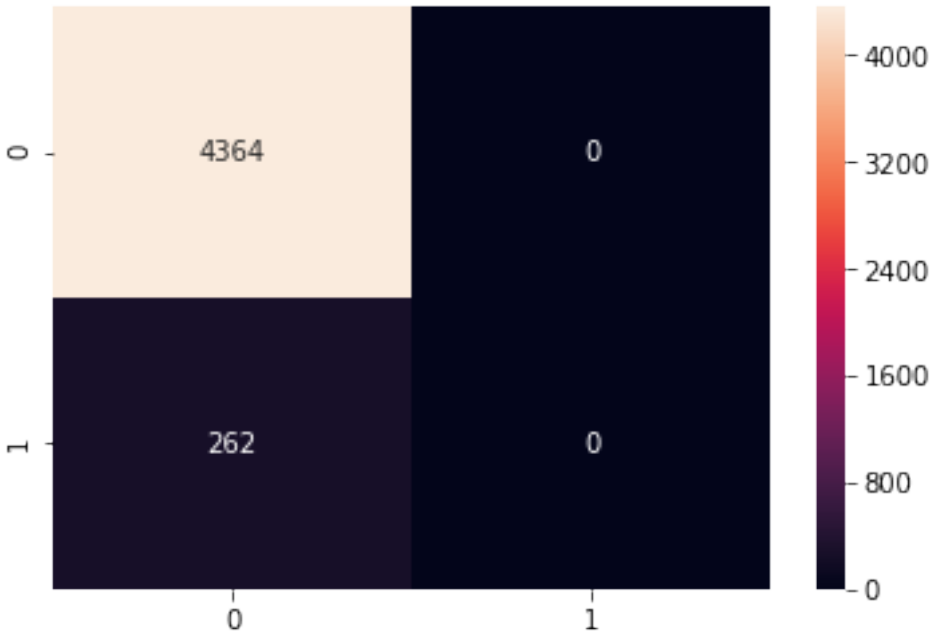
Then , we split our dataframe into a trainig set and a test set (30% for the test set) . Also we make a projection on our data to choose our target column wich is FraudFound_P , this operation is being done with the splitdataset function (code in Appendix D) After fitting our model , we call our predict function (Appendix E) to train and make prediction with the model , then the perfor function (Appendix F) to evaluate it .

The idea to find the optimum performance depending on the features selected to make the the prediction is to use RFE method . RFE is a method to choose the best set of features in data when used along with a linear model , such as Ridge or tree-based models such as RandomForests etc. It works in the following way. We initially start with all the features. For every step or iteration the worst x number of features (i.e the features which have the lowest feature importances in case of tree-based models and and the features which have the lowest absolute value in case of linear models) are eliminated using the "step" parameter till "n_features" are left. The RFECV object helps to tune or find this n_features parameter using cross-validation. For every step where "step" number of features are eliminated, it calculates the score on the validation data. The number of features left at the step which gives the maximum score on the validation data, is considered to be "the best n_features" of the data.

The result are as below :



Optimal number of features : 31
Best Accuracy : 94.3363597060095 %



7.2 Decision Tree

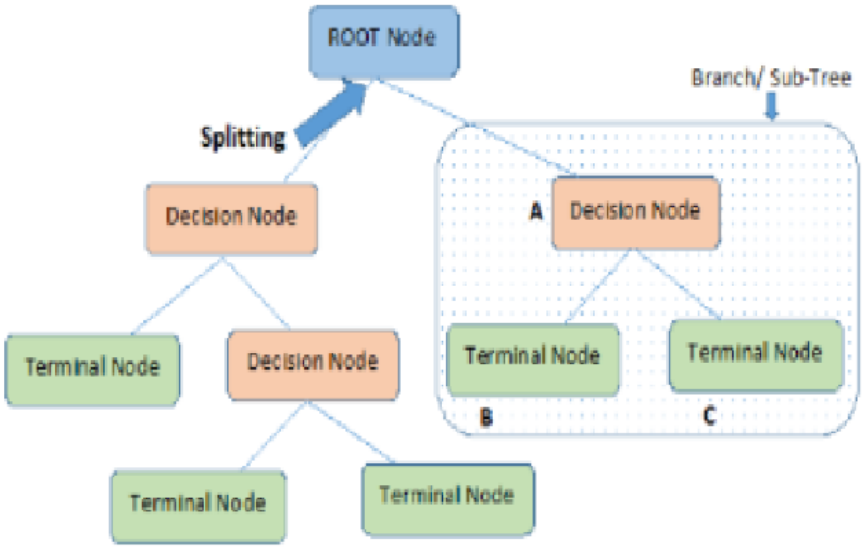
In this Chapter, we will create a decision tree model to predict whether a claim is fraudulent or not.W e will introduce the advantages and disadvantages of decision tree classification .

7.2.1 Preliminaries

The decision tree used in supervised learning is a method for both classification and regression problems. The decision tree is a graph that uses a branching method to show every possible output of a decision.The decision tree creates the model that predicts the output of a target variable by learning decision rules derived from the dataset.

A decisiontree consists of:

- Root node
- Splitting
- Decision node
- Branch/Sub-tree
- Terminal node



Some highlighted advantages of the decision tree :

- The investigator can visualise Decision trees , it is simple to understand.
- Can handle both numerical and categorical data.
- Can handle multi-outputproblems

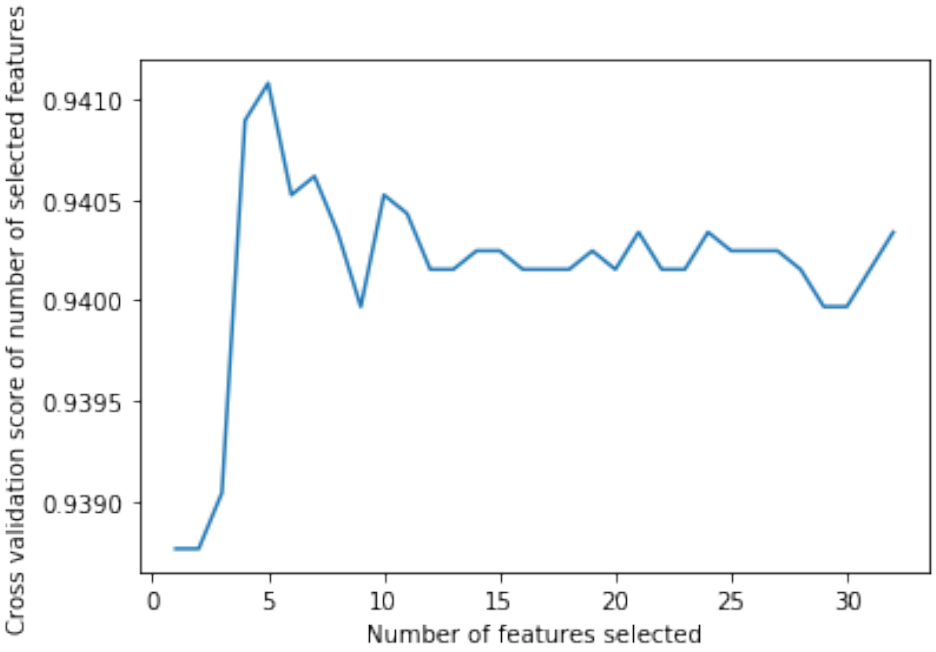
Some highlighted disadvantages ofthedecisiontree:

- Adecision tree learners can create biased treesif some classesdominate.The dataset-should be balanced.
- Over-complex trees(overfitting)..

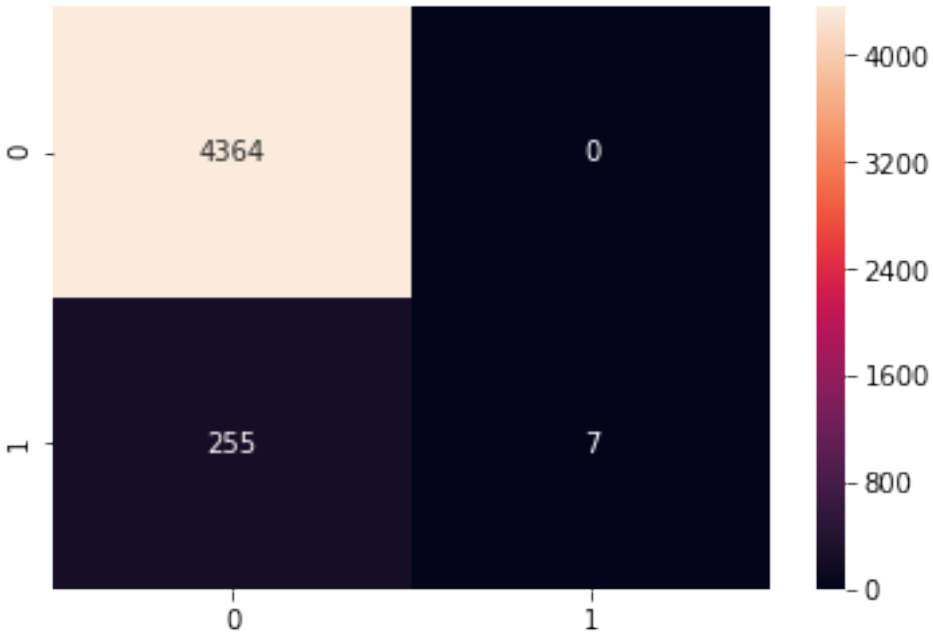
7.2.2 Experiments

As in the previous experiment in this examination, we will use Python programming language. The scikit ibrary will be utilised for the decision tree using RFECV. We will use the same dataset that is used for Linear Regression.

The results are as below :



Optimal number of features : 5
Best Accuracy : 94.48767833981842 %



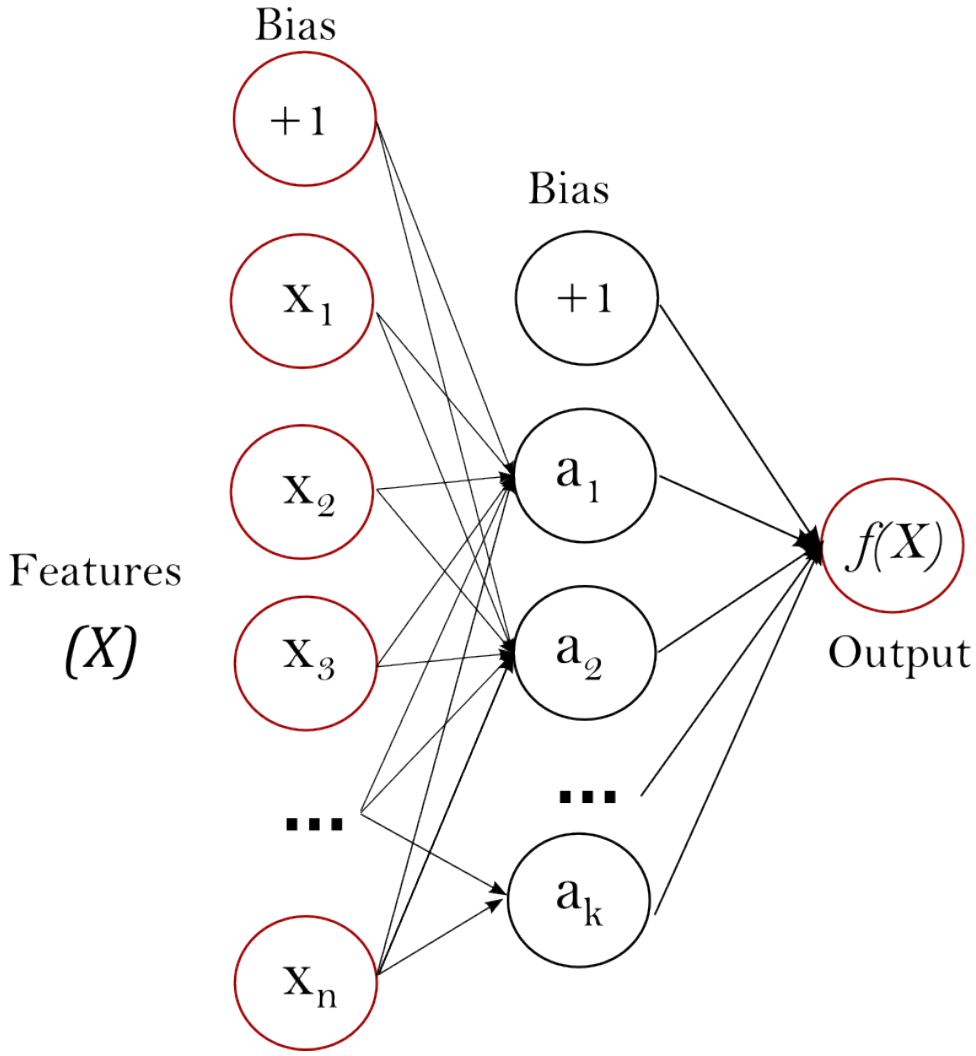
7.3 Neural Network

In this Chapter, we will create a decision tree model to predict whether a claim is fraudulent or not .

7.3.1 Preliminaries

Neural Network is a supervised learning model that learns a function $f(.) : R^m -> R^0$ by training on a dataset, where m is the number of dimensions for input and o is the

number of dimensions for output. Given a set of features $X = x_1, x_2, ..., x_m$ and a target y ,it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers.

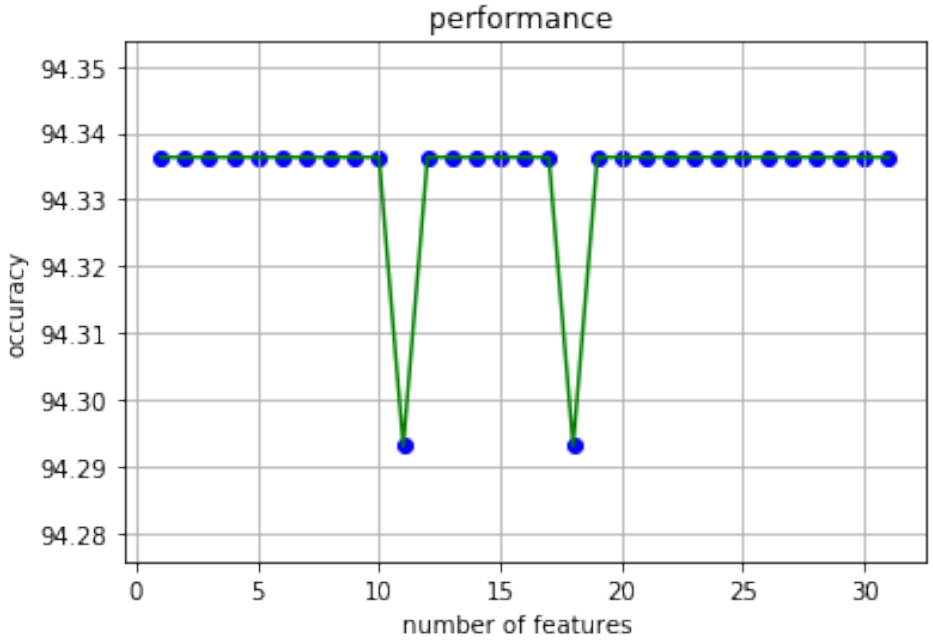


The leftmost layer, known as the input layer, consists of a set of neurons $\{x_i|x_1, x_2, ..., x_m\}$ representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation$w_1x_1 + w_2x_2 + ... + w_mx_m$, followed by a non-linear activation function $g(\cdot) : R \to R$ - like the hyperbolic tan function. The output layer receives the values from the last hidden layer and transforms them into output values.

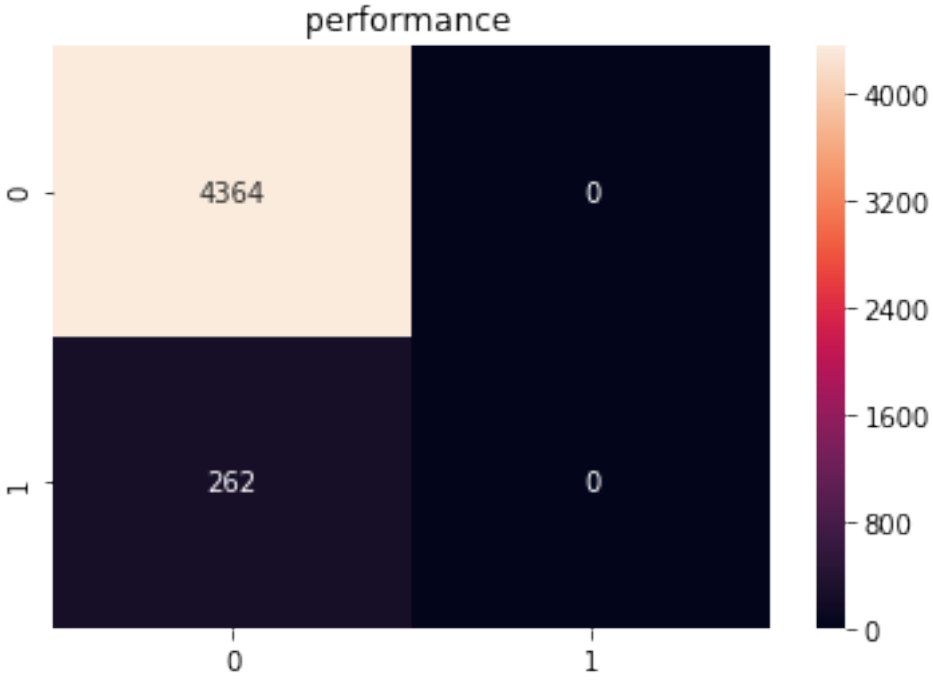
7.3.2 Experiments

For the expiriments , we are using scikitlearn library , we have 3 hidden layers with size of 13 each , and since RFECV don't support our model , the idea is to fit the model by one feature on evry step to classify them by their accuracy . Then , we repeat fitting our model with the best feautre , then the 2 best features , and so on until we find the best features for our model . (The code for Neural Network is given in Appendix G)

The results are as below :



Optimal number of features : 5
Best Accuracy : 94.3363597060095 %



7.4 Gaussian Naive Bayse

In this Chapter, we will create Gaussian Naive Bayse model to predict whether a claim is fraudulent or not and give the mathematical model of the algoritm .

7.4.1 Preliminaries

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes theorem with the naive assumption of independence between every pair of features. Given

a class variable y and a dependent feature vector x_1 through x_n, Bayes theorem states the following relationship:

$$P(y|x_1, ..., x_n) = \frac{P(y)p(x_1, ..., x_n|y)}{P(x_1, ...x_n)}$$

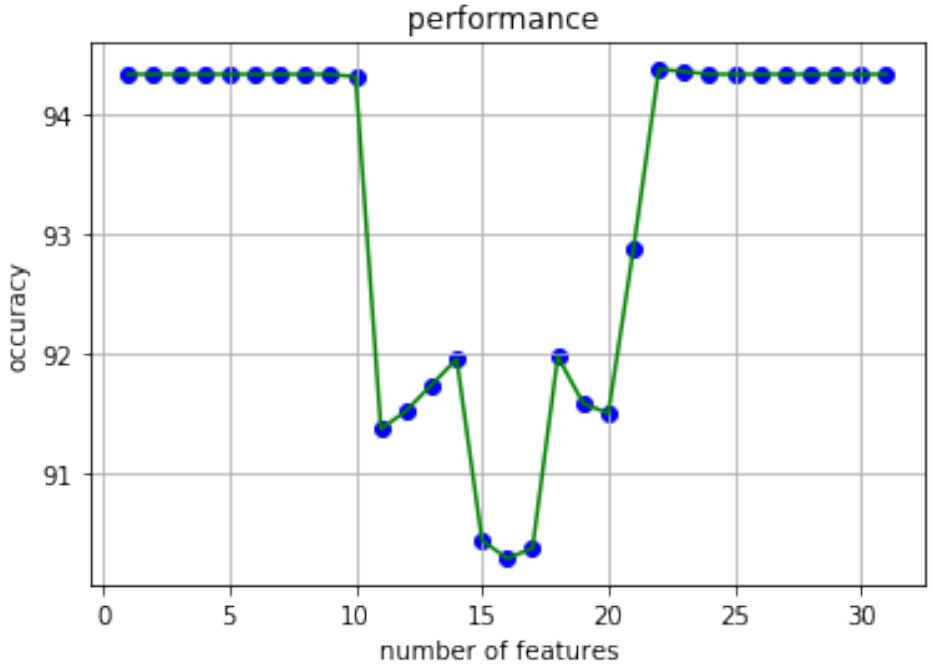
GaussianNB implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{x_i - \mu^2}{2\sigma_y^2}\right)$$

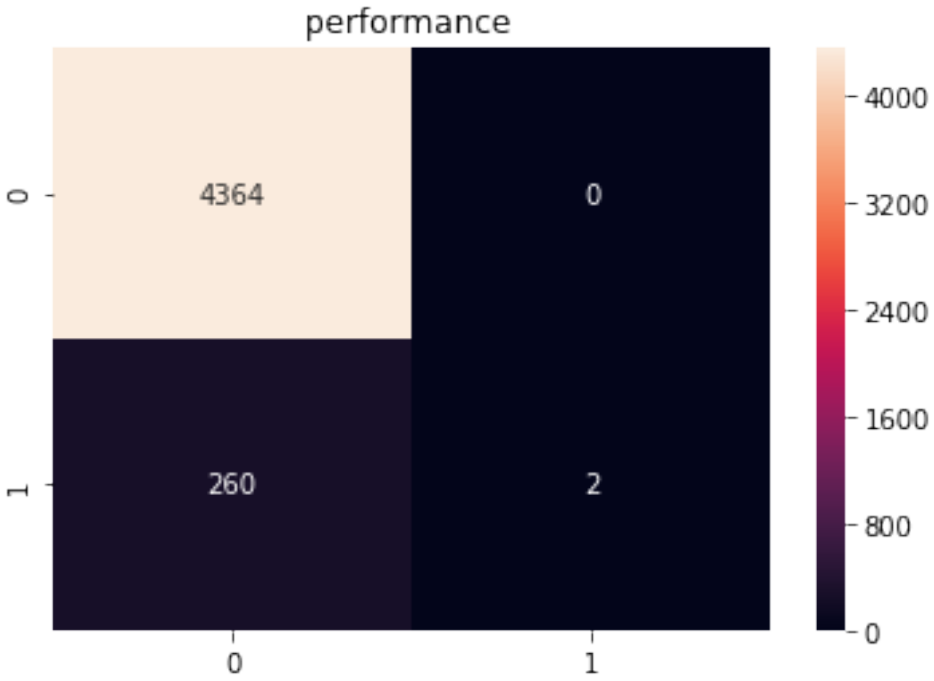
7.4.2 Experiments

To find the best features for the best result , we followed the same steps used followed before for the Neurol Network .

The results are as below :



Optimal number of features : 22
Best Accuracy : 94.37959360138348 %



7.5 K-Nearest Neighbour

In this Chapter, we will create a K-Nearest Neighbour model to predict whether a claim is fraudulent or not .

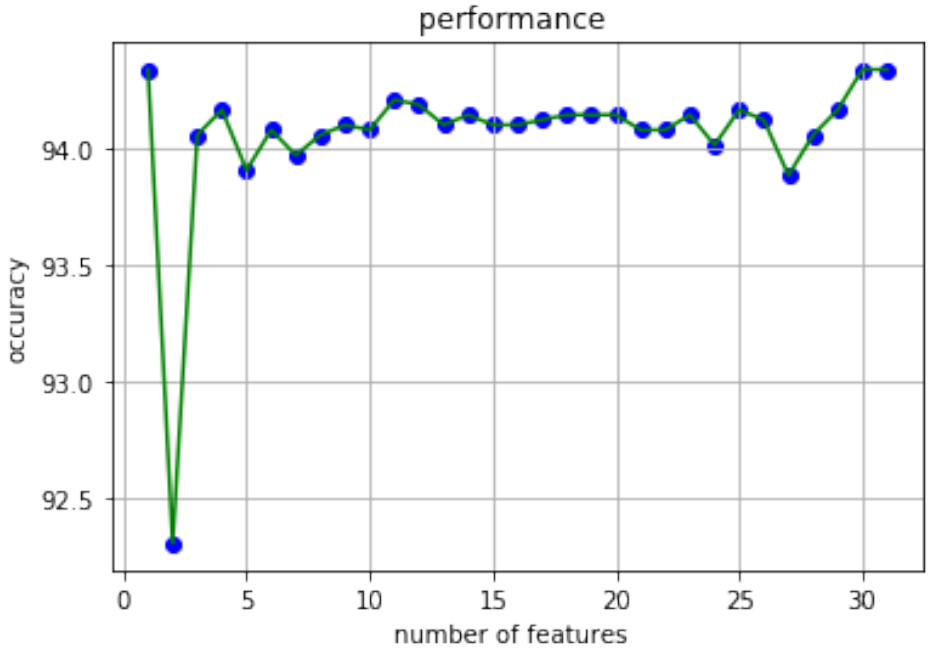
7.5.1 Preliminaries

Neighbors-based classification is a type of instance-based learning or non-generalizing learning: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

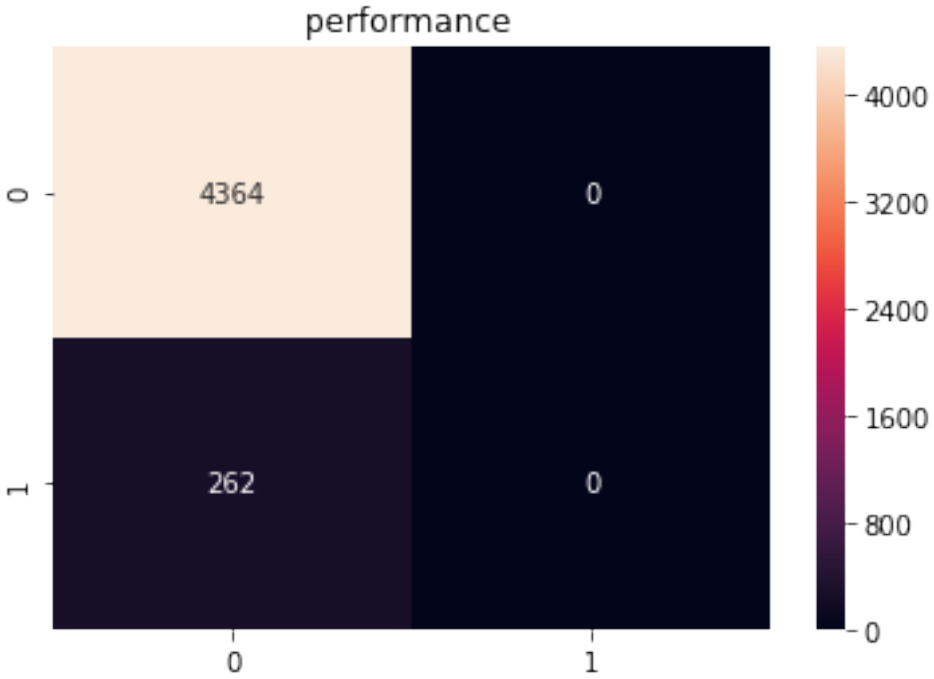
7.5.2 Experiments

We proceed the same method used for Neural Network and Gausssian Naive bayse .

The results are as below :



Optimal number of features : 1
Best Accuracy : 94.3363597060095 %



8 Conclusion

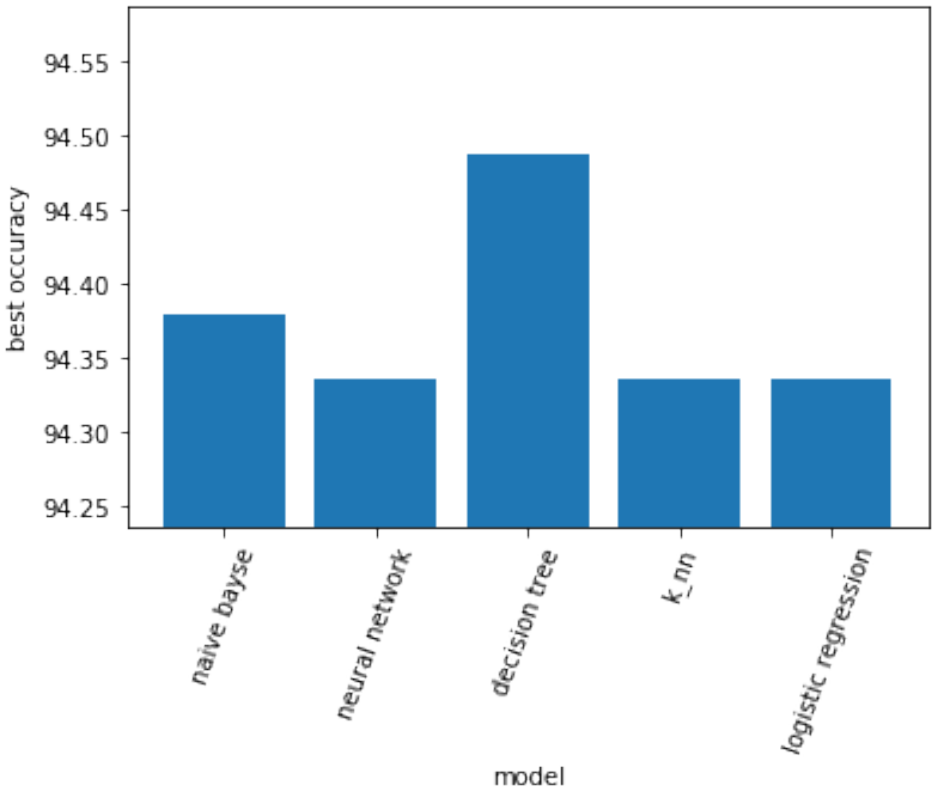


Table 2: Models statistics

Model	Precision	Recall	Accuracy
Logistic regression	0	0	94.33%
Decision tree	1	0.02	94.48%
Neural network	0	0	94.33%
Gaussian naive bayse	1	0.007	94.37%
K-nearest neighbour	0	0	94.33%

The decision tree model seems to be the winner model , it has the best accuracy , best precision and the best recall . Also , it reached his optimum result with only 5 features wich tell as that we are going to need only the best 5 features for our model for future classifications . Finally , decision tree model was the fastest model during the traing phase .

Appendices

A

Features classification visualization code

```
first ten features
    datadia = y
    data = X
    data2 = (data - data.mean()) / (data.std())
    data = pd.concat([y,data2.iloc[:,0:10]],axis=1)
    data = pd.melt(data,idvars="FraudFoundP",
    varname="features",
    valuename='value')
    plt.figure(figsize=(10,10))
    sns.violinplot(x="features", y="value", hue="FraudFoundP",
    data=data,split=True, inner="quart")
    plt.xticks(rotation=90)
```

B

Logistic regression Code

```
from data import import_data
    from splitdataset import splitdataset
    from model_performance import perfor
    from prediction import predict
    from sklearn.linear_model import LogisticRegression
    data = import_data()
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
    clf_object = LogisticRegression()
    clf_object.fit(X_train, y_train)
    # Performing training
    # Predicton on test
    y_pred,rfecv=predict(X_test , clf_object , X_train , y_train)
    perfor(y_test , y_pred , rfecv , X_train)
```

C

import_data Code

```
from sklearn import preprocessing
    from pymongo import MongoClient
    from pandas.io.json import json_normalize
    # function to import the dataset
    def import_data():
    client=MongoClient('wael-PC',27017)
    db = client.fraud_detection
```

```
claims=db.pfa
data_projection =”$project”: ”_id”:0
cursor = claims.aggregate([data_projection] )
data1=list(cursor)
data=json_normalize(data1)
data.head()
le = preprocessing.LabelEncoder()
for i in range(33):
if data.iloc[:,i].dtypes == object:
le.fit(data.iloc[:,i].astype(str))
data.iloc[:,i] = le.transform((data.iloc[:,i]).astype(str))
else:
pass
return(data)
```

D splitdataset Code

```
from sklearn.cross_validation import train_test_split
    def splitdataset(data):
    # Seperating the target variable
    X = data.loc[: ,data.columns != ”FraudFound_P”]
    Y = data.loc[: ,data.columns == ”FraudFound_P”]
    X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size = 0.3, random_state = 100)
    return X, Y, X_train, X_test, y_train, y_test
```

E Predict Code

```
from sklearn.feature_selection import RFECV
    def predict(X_test, clf_object , x_train , y_train):
    rfecv = RFECV(estimator=clf_object, step=1, cv=5,scoring='accuracy')
    rfecv = rfecv.fit(x_train, y_train)
    y_pred = rfecv.predict(X_test)
    return y_pred , rfecv
```

F perfor Code

```
from sklearn.metrics import accuracy_score
    import seaborn as sns # data visualization library
    import matplotlib.pyplot as plt
    from sklearn.metrics import confusion_matrix
    def perfor(y_test, y_pred , rfecv , X_train):
```

```
ac = accuracy_score(y_test,y_pred)
print('Accuracy is: ',ac*100)
cm = confusion_matrix(y_test,y_pred)
sns.heatmap(cm,annot=True,fmt="d")
plt.figure()
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score of number of selected features")
plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_)
plt.show()
print('Optimal number of features :', rfecv.n_features_)
print('Best features :', X_train.columns[rfecv.support_])
```

G

Neural Network code

```
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import cross_val_score
from data import import_data
from splitdataset import splitdataset
import operator
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix
data = import_data()
X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
# sorting the futures in a list by their impact in the classification
def sort():
f_list=dict()
fc_list=dict()
a=list(X)
clf_object = MLPClassifier(hidden_layer_sizes=(13,13,13,13),max_iter=50)
n_feats = X.shape[1]
for i in range(n_feats):
xi = X.iloc[:, i].reshape(-1,1)
scores = cross_val_score(clf_object, xi, Y)
f_list[i]= scores.mean()
fc_list[i]= a[i]
f_list = sorted(f_list.items() , key=operator.itemgetter(1) ) #sorted by occuracy
return f_list,fc_list
#function to generate lists of features
def lists_generator(v,c):
L=list()
for i in range (32) :
li=list()
for j in range (i+1) :
li.append(c[v[j][0]])
```

```
    L.append(li)
    L.reverse()
    return L
    def main() :
    v,c=sort()
    Ls=lists_generator(v,c)
    ocuur_list=list()
    cm_list=list()
    model = MLPClassifier(hidden_layer_sizes=(13,13,13,13),max_iter=50)
    for i in range(31) :
    model.fit(X_train.drop(Ls[i+1] , axis=1) , y_train) y_pred= model.predict(X_test.drop(Ls[i+1]
, axis=1))
    ac=accuracy_score(y_test,y_pred)*100
    ocuur_list.append(ac)
    cm = confusion_matrix(y_test,y_pred)
    cm_list.append(cm)
    n_of_feat=list(range(1,32,1))
    plt.scatter(n_of_feat,ocuur_list,marker="o",color="blue")
    plt.title("performance ")
    plt.xlabel("number of features ")
    plt.ylabel("occuracy")
    plt.plot(n_of_feat,ocuur_list,color="green",ls="-")
    plt.grid()
    print("best performance : " , max(ocuur_list) )
    print("number of features : " , ocuur_list.index(max(ocuur_list))+1 )
    cmm = cm_list[ocuur_list.index(max(ocuur_list))]
```