National Institute of Technology,

Surathkal, Karnataka

M.Tech: Computational and Data Science

Assignment 2: Insurance Fraud Detection

by Team 10
Subhadeep Santra - 202CD028
Tejpal Singh Rao - 202CD030
Abhishek Rao - 202CD001

ACKNOWLEDGEMENT

It is with immense pleasure and satisfaction that we present our first attempt in practical experience in the form of project work. This project is a result of dedicated effort. It gives us immense pleasure to prepare this report.

We would like to express heartfelt thanks to Head of The Department, MACS Mr. Shyam S. Kamath for the encouragement he has given us throughout the coursework. We are highly indebted to our Machine Learning professor Mr. Jidesh P for his guidance constant supervision as well as for providing necessary information regarding the project and also for his support in completing the project.

Finally, we would like to thank our parents who have always been supportive and guided us to work on the right path of life.

Subhadeep Santra - 202CD028

Tejpal Singh Rao - 202CD030

Abhishek Rao - 202CD001

Project Objective:

Auto-insurance Fraud Detection:

To identify whether the claims made by the customer are fraudulent or not. Identify the key attributes that determine the likelihood of false insurance claims.

Introduction:

Fraud is causing billions of $$ in loss especially for BFSI. Fraudulent claims a widespread disease and an expensive affair to the Insurance system. Moreover, the incidence of insurance fraud keeps growing every year. In this project, we develop a predictive model based on ML algorithms to investigate fraudulent claims.
We will use the historical insurance claim data including normal and fraudulent ones, to investigate the normal/fraud behavior features based on ML techniques, and using these features will check if a claim is fraud or not. A comparative study will be performed to decide which ML classifier is the best for this analysis to train the behavior features of fraudulent claims. This includes some preliminary knowledge of the insurance system and its fraudulent claim behaviors, analysis of the characteristics of insurance data.

The data that we have here is from Automobile Insurance. The answer between YES/NO, is a Binary Classification task. Therefore, the reporting here will deal with classification algorithms to detect fraud transactions in Python. We shall compare different available classifiers including RandomForest, Support Vector Classifier and Extreme Gradient Boosting.

Methodology:

- Downloading Dataset from Kaggle
- Data Preprocessing
- Exploratory Data Analysis
- Using Different Machine Learning Algorithms
- Comparison of the different Machine Learning Models
- Choosing the best One

Data Pre-processing and Feature engineering:

After importing necessary libraries, we read the dataset into python. The data-set is with 1000 rows and 40 columns.

```
1
2 data=pd.read_csv('insurance_claims.csv')
3 data.head()
```

	months_as_customer	age	policy_number	policy_bind_date	policy_state	policy_csl	policy_deductable	policy_annual_premium	umbrella_limit	insured_zip
0	328	48	521585	2014-10-17	OH	250/500	1000	1406.91	0	466132
1	228	42	342868	2006-06-27	IN	250/500	2000	1197.22	5000000	468176
2	134	29	687698	2000-09-06	OH	100/300	2000	1413.14	5000000	430632
3	256	41	227811	1990-05-25	IL	250/500	2000	1415.74	6000000	608117
4	228	44	367455	2014-06-06	IL	500/1000	1000	1583.91	6000000	610706

```
1 data.columns
```

```
Index(['months_as_customer', 'age', 'policy_number', 'policy_bind_date',
       'policy_state', 'policy_csl', 'policy_deductable',
       'policy_annual_premium', 'umbrella_limit', 'insured_zip', 'insured_sex',
       'insured_education_level', 'insured_occupation', 'insured_hobbies',
       'insured_relationship', 'capital-gains', 'capital-loss',
       'incident_date', 'incident_type', 'collision_type', 'incident_severity',
       'authorities_contacted', 'incident_state', 'incident_city',
       'incident_location', 'incident_hour_of_the_day',
       'number_of_vehicles_involved', 'property_damage', 'bodily_injuries',
       'witnesses', 'police_report_available', 'total_claim_amount',
       'injury_claim', 'property_claim', 'vehicle_claim', 'auto_make',
       'auto_model', 'auto_year', 'fraud_reported', '_c39'],
      dtype='object')
```

Data processing is an important activity before the data is fed into the ML model.

Some variables such as ‘policy_bind_date’, ‘incident_date’, ‘incident_location’ and ‘insured_zip’ contain a very high number of levels, which we remove for the goal of this exercise.
We make a list of columns not necessary for our prediction (given below) and we drop them:
policy_number, policy_bind_date, policy_state, insured_zip, incident_location, incident_date, incident_state, incident_city, insured_hobbies, auto_make, auto_model, auto_year, _c39

The missing values in the dataset are replaced with NaN for them to be imputed down the line.

We segregate the categorical columns first and explore them:

```
1 cat_df = data.select_dtypes(include=['object']).copy()
```

```
1 cat_df.columns
```

```
Index(['policy_csl', 'insured_sex', 'insured_education_level',
       'insured_occupation', 'insured_relationship', 'incident_type',
       'collision_type', 'incident_severity', 'authorities_contacted',
       'property_damage', 'police_report_available', 'fraud_reported'],
      dtype='object')
```

As the columns which have missing values are only categorical, we use the categorical imputer from scikit.learn. We performed a custom mapping for encoding (label-encoding) on these columns: **policy_csl, insured_education_level, incident_severity, insured_sex, property_damage, police_report_available** and finally our target column **fraud reported.**
After that we drop the above columns and perform **one hot encoding** on the rest of the categorical columns.
Then we combine the numerical and categorical data-frames to get the final dataset for model building. In the end the predictor variables and target variables are separated in x and y. A train-test split helped us to separate the dataset into **train_x, test_x, train_y,** and **test_y.**

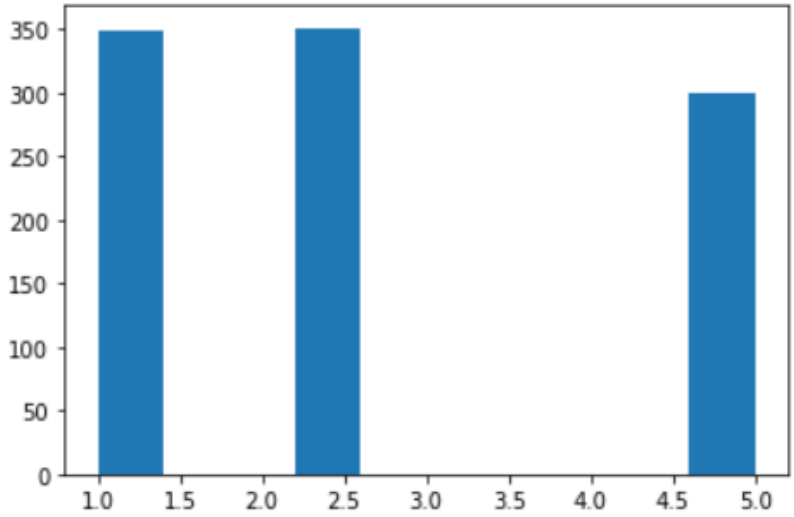
Data standardization:

Before we start modelling our dataset, we scale the data of the numerical columns. The numerical columns are extracted and scaled using StandardScaler from scikit.learn. After that we drop the numerical columns and concatenate the scaled data with our original dataset. Now we're ready for model-training. Before that, we explore the data further to understand its distribution and correlation which would help us in choosing the suitable algorithms.

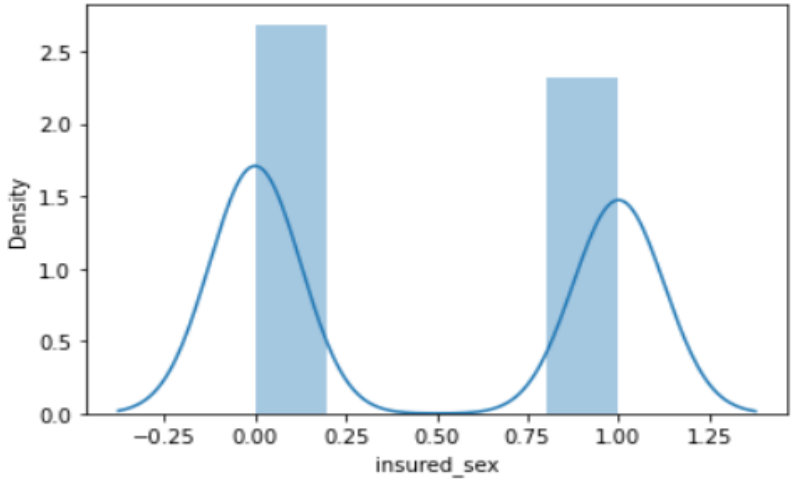
Exploratory Data Analysis:

Exploratory Data Analysis is an approach of analyzing data sets to summarize their main characteristics, often using statistical graphs and other data visualisation methods.EDA is an approach to data analysis that postpones the usual assumptions about what kind of model the data follow with the more direct approach of allowing the data itself to reveal its underlying structure and model.
Most EDA techniques are graphical in nature with a few quantitative techniques. The reason for the heavy reliance on graphics is that by its very nature the main role of EDA is to open-mindedly explore, and graphics gives the analysts unparalleled power to do so, enticing the data to reveal its structural secrets, and being always ready to gain some new, often unsuspected, insight into the data.

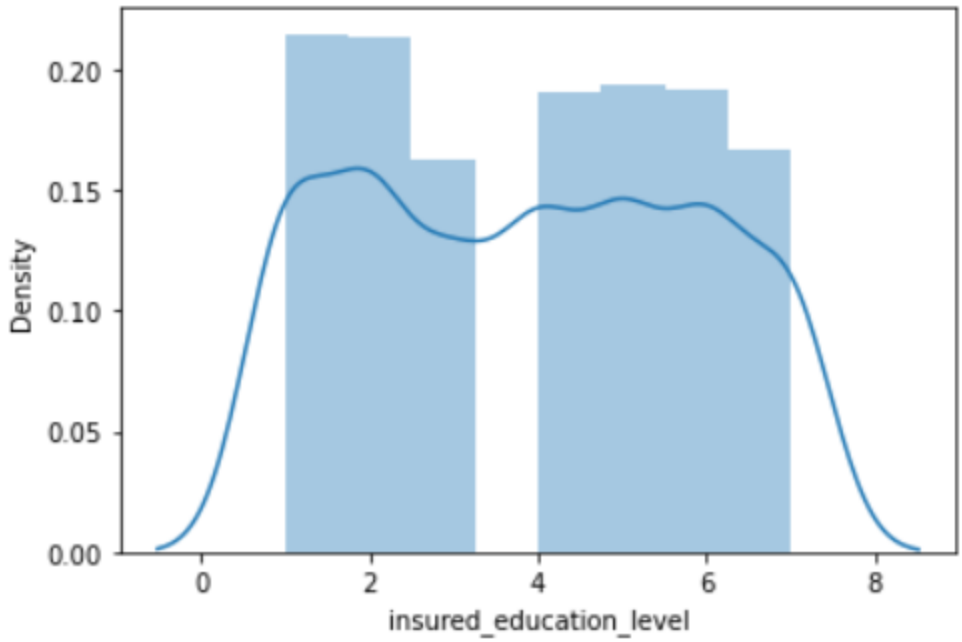
From this distribution plot of policy_csl, we see that for almost all categories of CSL the data is uniformly distributed.



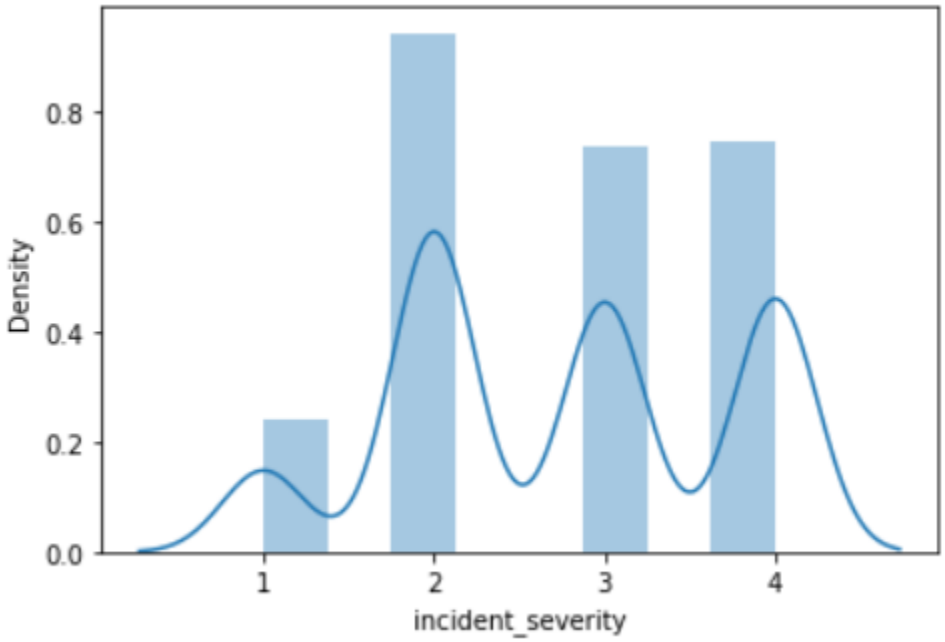
The next distribution plot of the sex of claimants show that women are slightly more likely to claim than men as shown in this plotting



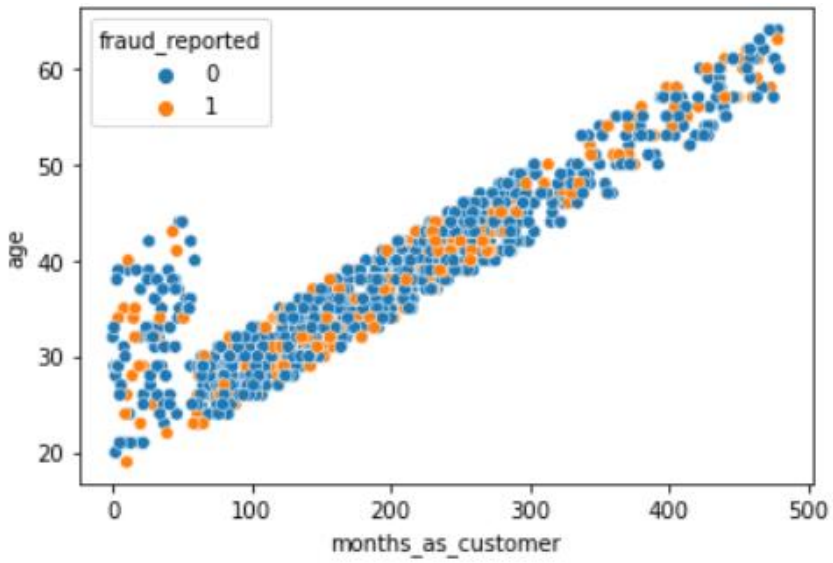
We can see that for almost all categories of the education level of the person insured the data is uniformly distributed



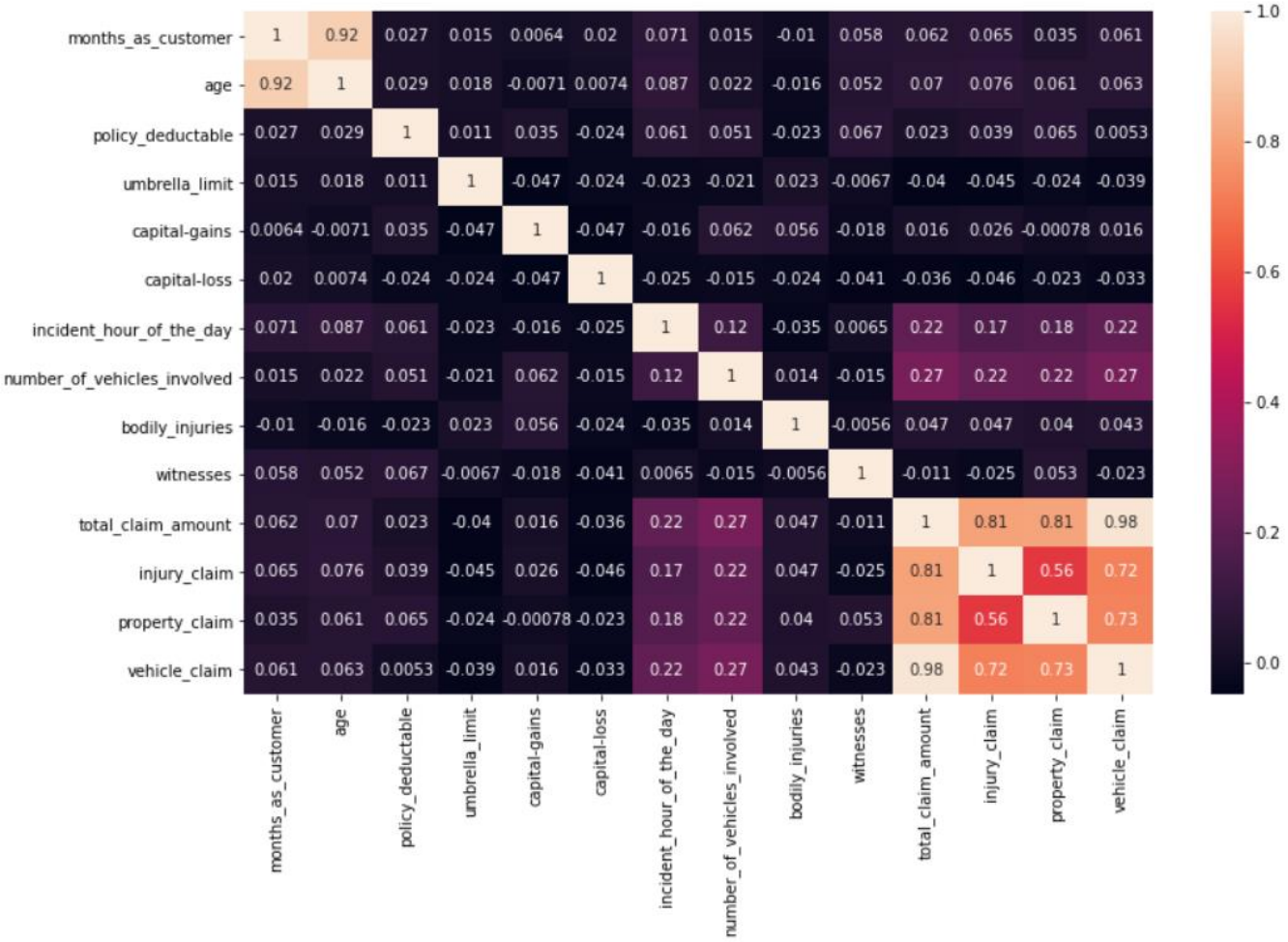
We can see that there are least claims for trivial incidents, most claims for minor incidents(incident_severity = 2), and for major and Total loss incidents the claims are almost equal.



From the graph it can be concluded that most of the fraud cases are done by the customers new to the company and that too by comparatively younger ones.



From the plot below, we can see that there is high correlation between age and the number of months. A high correlation between total claim amount, injury claim,vehicle claim, and property claim exists as total claim is the sum of all others. So, we drop the age and total claim amount column.



Model Training:

A machine learning training model is a process in which a machine learning (ML) algorithm is fed with sufficient training data to learn from.

For computing the problem we use different different models:

1) Logistic Regression
2) Random Forest Classifier
3) Support Vector Classifier
4) XGB Classifier

By using this we will find i)Accuracy Score ii)Confusion Matrix iii)Classification Report.

i) Accuracy Score:

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition:

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

ii) Confusion Matrix:

Confusion Matrix as the name suggests gives us a matrix as output and describes the complete performance of the model, showing the number of TN, FP, FN, TP cases.

iii) Classification Report:-

The classification report displays the precision, recall, F1, and support scores for the model.

F1 Score:- F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances).

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

Precision : It is the number of correct positive results divided by the number of positive results predicted by the classifier.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Recall : It is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

1) Logistic Regression:-

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

```
1 print(classification_report(test_y, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.78      0.20      0.32       182
           1       0.28      0.85      0.43        68

    accuracy                           0.38       250
   macro avg       0.53      0.53      0.37       250
weighted avg       0.65      0.38      0.35       250
```

We can see in the Classification Report we got very less accuracy (accuracy_score=0.376) in Logistic Regression. So Logistic regression was unable to produce any great result. So, we have checked the other classifiers to compare.

2) Random Forest Classifier:-

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

```
1 print(classification_report(test_y, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.73      0.99      0.84       182
           1       0.67      0.03      0.06        68

    accuracy                           0.73       250
   macro avg       0.70      0.51      0.45       250
weighted avg       0.71      0.73      0.63       250
```

We can see in the Classification Report that accuracy(accuracy_score=0.732) is now improved but we have to compare accuracy scores with every model so that we will get the most accurate result.

3) Support Vector Classifier:-

Support Vector Machine (SVM) is a relatively simple Supervised Machine Learning Algorithm used for classification and/or regression. It is more preferred for classification but is sometimes very useful for regression as well. Basically, SVC finds a hyper-plane that creates a boundary between the types of data.

```
sv_classifier=SVC(C=0.1, random_state=0)
y_pred = sv_classifier.fit(train_x, train_y).predict(test_x)
sc=accuracy_score(test_y,y_pred)
sc
```

```
0.728
```

We Can see in this code that our accuracy score is 0.728.

Hyperparameter Tuning:-

Now we pass predefined values for hyperparameters to the GridSearchCV function. We do this by defining a dictionary in which we mention a particular hyperparameter along with the values it can take.

GridSearchCV tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method. Hence after using this function we get accuracy/loss for every combination of hyperparameters and we can choose the one with the best performance.

This function helps to loop through predefined hyperparameters and fit our estimator (model) on our training set. So, in the end, we can select the best parameters from the listed hyperparameters.

```
from sklearn.model_selection import GridSearchCV
param_grid = {"kernel": ['rbf','sigmoid'],
              "C":[0.1,0.5,1.0],
              "random_state":[0,100,200,300]}
grid = GridSearchCV(estimator=sv_classifier, param_grid=param_grid, cv=5,  verbose=3)
grid.fit(train_x, train_y)
```

```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
[CV 1/5] END ..............C=0.1, kernel=rbf, random_state=0; total time=   0.0s
[CV 2/5] END ..............C=0.1, kernel=rbf, random_state=0; total time=   0.0s
[CV 3/5] END ..............C=0.1, kernel=rbf, random_state=0; total time=   0.0s
[CV 4/5] END ..............C=0.1, kernel=rbf, random_state=0; total time=   0.0s
[CV 5/5] END ..............C=0.1, kernel=rbf, random_state=0; total time=   0.0s
[CV 1/5] END ............C=0.1, kernel=rbf, random_state=100; total time=   0.0s
[CV 2/5] END ............C=0.1, kernel=rbf, random_state=100; total time=   0.0s
[CV 3/5] END ............C=0.1, kernel=rbf, random_state=100; total time=   0.0s
[CV 4/5] END ............C=0.1, kernel=rbf, random_state=100; total time=   0.0s
[CV 5/5] END ............C=0.1, kernel=rbf, random_state=100; total time=   0.0s
[CV 1/5] END ............C=0.1, kernel=rbf, random_state=200; total time=   0.0s
[CV 2/5] END ............C=0.1, kernel=rbf, random_state=200; total time=   0.0s
[CV 3/5] END ............C=0.1, kernel=rbf, random_state=200; total time=   0.0s
[CV 4/5] END ............C=0.1, kernel=rbf, random_state=200; total time=   0.0s
```

As shown in the figure we take hyperparameters as param_grid.

```
{"kernel": ['rbf','sigmoid'],
"C":[0.1,0.5,1.0],
"random_state":[0,100,200,300]}
```

Here C, random_state and kernels are some of the hyperparameters of an SVM model. Note that the rest of the hyperparameters will be set to their default values.

Now after using Hyperparameter Tuning we again check our accuracy :-

```
1 sv_classifier=SVC(C=0.1, random_state=0)
2 y_pred = sv_classifier.fit(train_x, train_y).predict(test_x)
3 sc=accuracy_score(test_y,y_pred)
4 sc
```

```
0.728
```

We can see as per our dataset the accuracy remains the same but generally it will increase.Now we will compare Accuracy score with other models.

4) XGB Classifier:-

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.)
So we can also use this classifier to predict our program :-

```
1 from xgboost import XGBClassifier
2 xgb=XGBClassifier()
3 y_pred = xgb.fit(train_x, train_y).predict(test_x)
4 ac2=accuracy_score(test_y,y_pred)
5 ac2
```

```
Out[130]: 0.748
```

We Can see in this code that our accuracy score is 0.748.Now we pass predefined values for hyperparameters to the GridSearchCV function or perform Hyperparameter Tuning.

```
1 param_grid = {"n_estimators": [10, 50, 100, 130], "criterion": ['gini', 'entropy'],
2                               "max_depth": range(2, 10, 1)}
3
4 #Creating an object of the Grid Search class
5 grid = GridSearchCV(estimator=xgb, param_grid=param_grid, cv=5,  verbose=3,n_jobs=-1)
6
7 #finding the best parameters
8 grid.fit(train_x, train_y)
```

As shown in the figure we can say we take hyperparameters as param_grid.

```
{"n_estimators": [10, 50, 100, 130], "criterion": ['gini', 'entropy'],
                 "max_depth": range(2, 10, 1)}
```

Here criterion, n_estimators and max_depth are some of the hyperparameters of an XGBoost model. Note that the rest of the hyperparameters will be set to their default values.

Now after using Hyperparameter Tuning we again check our accuracy :-

```
1 print(classification_report(test_y, y_pred))
```

	precision	recall	f1-score	support
0	0.83	0.83	0.83	182
1	0.54	0.54	0.54	68
accuracy			0.75	250
macro avg	0.69	0.69	0.69	250
weighted avg	0.75	0.75	0.75	250

We can see in the Classification Report our Accuracy has slightly increased and Now our **accuracy is 0.75.**

After comparing all the models we can conclude that the Accuracy Score is maximum in the case of XGB Classifier. So we should use XGB Classifier to compute insurance fraud detection.

Conclusion:-

Through this project we have tried to develop ML algorithm to detect fraud. This report deals with classification algorithm random forest model, support vector machines, Logistic regression, xg boost to detect fraud transaction in python. The project has used historical transaction data including normal transactions and fraud ones to obtain normal/fraud behaviour features based on machine learning technique, and utilized these features to check if a insurance claim is fraud or not. A comparative study has been conducted to decide which classifier is best for this project to train the behaviour features of normal and abnormal transactions.

References:-

1) https://www.kaggle.com/
2) **Numpy Documentation**
3) **Pandas Documentation**
4) **Python Documentation**
5) https://www.youtube.com/watch?v=gmvvaobm7eQ&list=PLeo1K3hjS3uvCeTYTeyfe0-rN5r8zn9rw