Abstract

Project 1

Credit Card Fraud Detection

Banks have the data related to their customer can provide Credit Card for a person upon the request of a customer. But before they issue any credit card the bank has to check the history transactions of a customer based on the credit history bank has to decide whether the credit card can be issued to an individual or not. The Dataset consists of the Entries of a customer and the class whether the entry is fraud or not. So, it is a binary classification problem.

Project 2

Used Car Price Prediction

Cars are used for easy mode of transportation, for a middle class person to buy a new car he can't afford that much amount, in that case he can choose an used car with better condition and can buy that car. For a used car with certain specifications we have to predict the price of the car. The Dataset consists of attributes and its Price is given for a used car. So, it is a Regression problem.

Tools / Skills Used

1. Python Programming
2. Jupyter Notebook
3. Pandas
4. Numpy
5. Matplotlib
6. Seaborn
7. Exploratory Data Analysis
8. Machine Learning
9. Deep Learning
10. Neural Networks

Problem Statement 1 - Introduction to the project

Credit Card Fraud Detection:

Banks have the data related to their customer can provide Credit Card for a person upon the request of a customer. But before they issue any credit card the bank has to check the history transactions of a customer based on the credit history bank has to decide whether the credit card can be issued to an individual or not. The Dataset consists of the Entries of a customer and the class whether the entry is fraud or not. So, it is a binary classification problem. The problem statement is to train a model that can predict whether the given entry is fraud or not.

Implementation

Workflow:

Modelling:

1. **Random Forest:** Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

2. **Decision Tree:** A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

3. **Neural networks**: Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input.

Code Snippets:

```
In [72]: import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import seaborn as sns
         from sklearn.metrics import confusion_matrix
         from sklearn.metrics import classification_report
```

-->imported libraries

```
In [2]: data=pd.read_csv('creditcard.csv')
```

--> loaded data

Using Random Forest

```
In [55]: # Fitting Random Forest Classification to the Training set
         from sklearn.ensemble import RandomForestClassifier
         classifierrc = RandomForestClassifier(n_estimators = 10, criterion = 'entropy', random_state = 0)
         classifierrc.fit(X_train, y_train)
```

```
Out[55]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                                criterion='entropy', max_depth=None, max_features='auto',
                                max_leaf_nodes=None, max_samples=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=10,
                                n_jobs=None, oob_score=False, random_state=0, verbose=0,
                                warm_start=False)
```

```
In [56]: #making predictions with test set
         y_pred=classifierrc.predict(X_test)
         pred_real=pd.DataFrame(columns=['y_pred','y_test'])
         pred_real['y_pred']=y_pred
         pred_real['y_test']=y_test
```

```
In [57]: pred_real.head()
```

Out[57]:

	y_pred	y_test
0	1	1
1	1	1
2	1	1
3	0	0
4	1	1

```
In [27]: # Making the Confusion Matrix
         confusion_matrix(y_test, y_pred)
```

```
Out[27]: array([[56854,     7],
                [   24,    77]], dtype=int64)
```

```
In [28]: classifierrc.score(X_test,y_test)
```

```
Out[28]: 0.9994557775359011
```

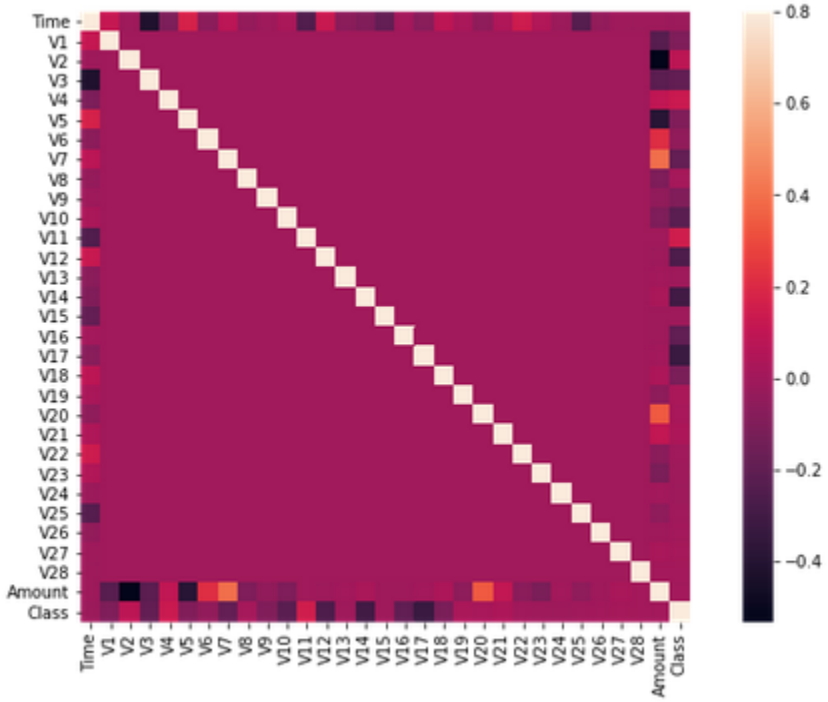
```
In [29]: print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56861
           1       0.92      0.76      0.83       101

    accuracy                           1.00     56962
   macro avg       0.96      0.88      0.92     56962
weighted avg       1.00      1.00      1.00     56962
```

Visualization Snippets:



Conclusion/ Results

Random forest classifier is the best fit to this dataset, gives 99% accuracy. The Decision tree classifier and Artificial Neural Networks also performed very well with good accuracy. As we have some oversampling we had used “smote” to maintain the balance between both the classes on the data and used random forest again also performed well.

Future Scope

In future, the models can be upgraded with some better techniques in terms of getting higher and better accuracy, f1-score, precision, recall etc.

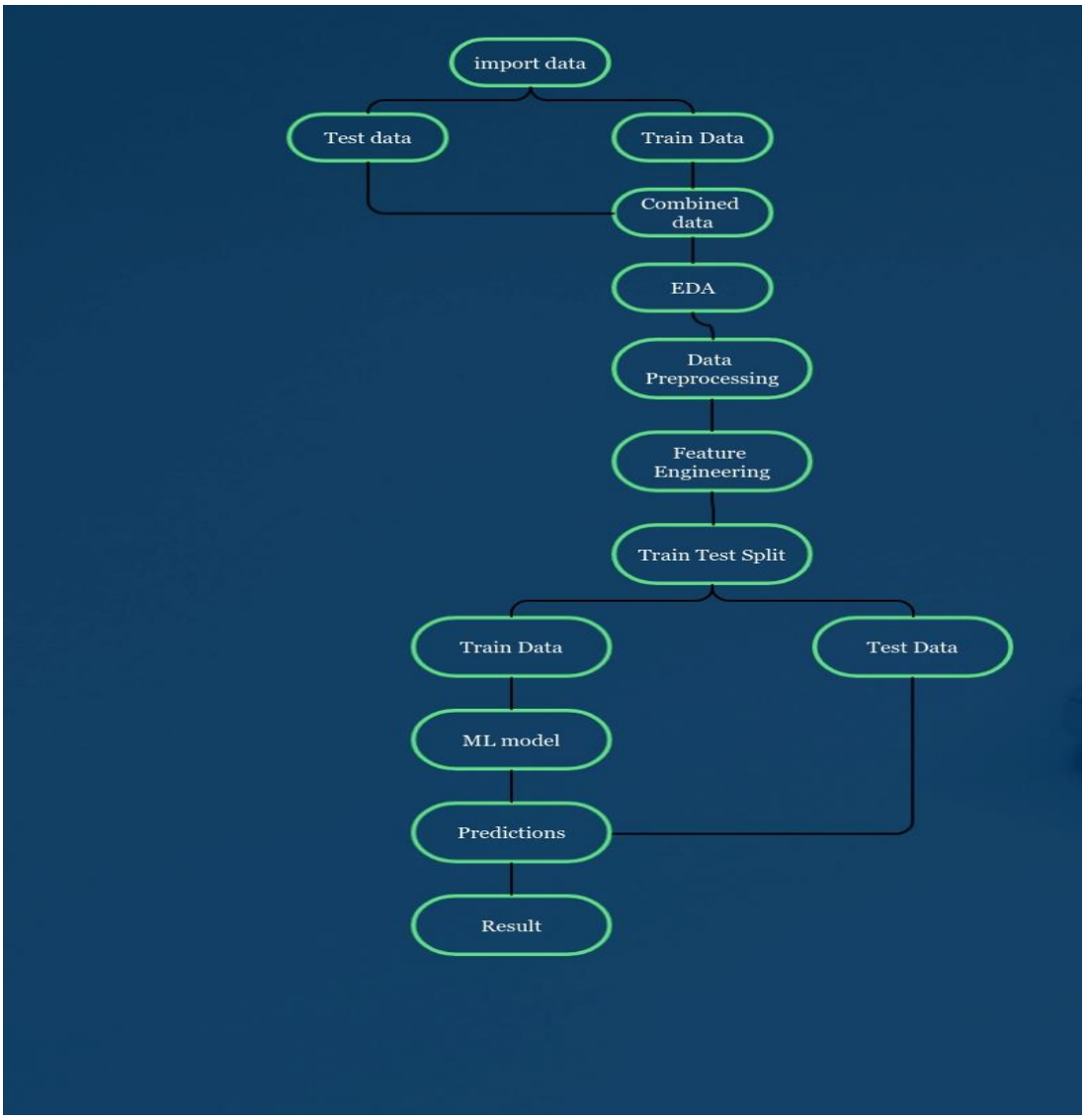
Problem Statement 2 - Introduction to the project

Used Car Price Prediction:

Cars are used for easy mode of transportation, for a middle class person to buy a new car he can't afford that much amount, in that case he can choose an used car with better condition and can buy that car. For a used car with certain specifications we have to predict the price of the car. The Dataset consists of attributes and its Price is given for a used car. So, it is a Regression problem.

Implementation

Workflow:



Modelling:

Random Forest: Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees

Linear regression:

Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. A linear regression line has an equation of the form Y = a + bX, where X is the explanatory variable and Y is the dependent variable.

Code Snippets:

Using Multiple Linear Regresssion

```
In [681]: # fitting train data to the regression
          from sklearn.model_selection import train_test_split
          from sklearn.linear_model import LinearRegression


          X_train, X_test, y_train, y_test = train_test_split(train_X, train_y, test_size=0.3, random_state=42)
          regressor= LinearRegression()
          regressor.fit(X_train, y_train)

Out[681]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

---> model is ready and trained with training set

```
In [682]: # making predictions
          y_pred = regressor.predict(X_test)
          pred_real=pd.DataFrame(columns=['y_pred','y_test'])
          pred_real['y_pred']=np.array(y_pred).reshape(-1)
          pred_real['y_test']=np.array(y_test).reshape(-1)
          pred_real.head()
```

Out[682]:

	y_pred	y_test
0	4.867982	5.75
1	7.027573	5.75
2	9.091410	8.00
3	-2.470615	5.90
4	4.604518	3.81

```
In [683]: regressor.score(X_test,y_test)

Out[683]: 0.727104121676971
```

Using Random Forest

```
In [685]: # Fitting Random Forest Regression to the dataset
          from sklearn.ensemble import RandomForestRegressor
          regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
          regressor.fit(X_train, y_train)

Out[685]: RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                                max_depth=None, max_features='auto', max_leaf_nodes=None,
                                max_samples=None, min_impurity_decrease=0.0,
                                min_impurity_split=None, min_samples_leaf=1,
                                min_samples_split=2, min_weight_fraction_leaf=0.0,
                                n_estimators=10, n_jobs=None, oob_score=False,
                                random_state=0, verbose=0, warm_start=False)
```

---> model is ready and trained with training set

```
In [686]: # making predictions
          y_pred = regressor.predict(X_test)
          pred_real=pd.DataFrame(columns=['y_pred','y_test'])
          pred_real['y_pred']=y_pred
          pred_real['y_test']=np.array(y_test).reshape(-1)
          pred_real.head()
```

Out[686]:

	y_pred	y_test
0	5.519	5.75
1	6.720	5.75
2	7.584	8.00
3	4.945	5.90
4	4.462	3.81

```
In [687]: regressor.score(X_test,y_test)

Out[687]: 0.8355048894084132
```

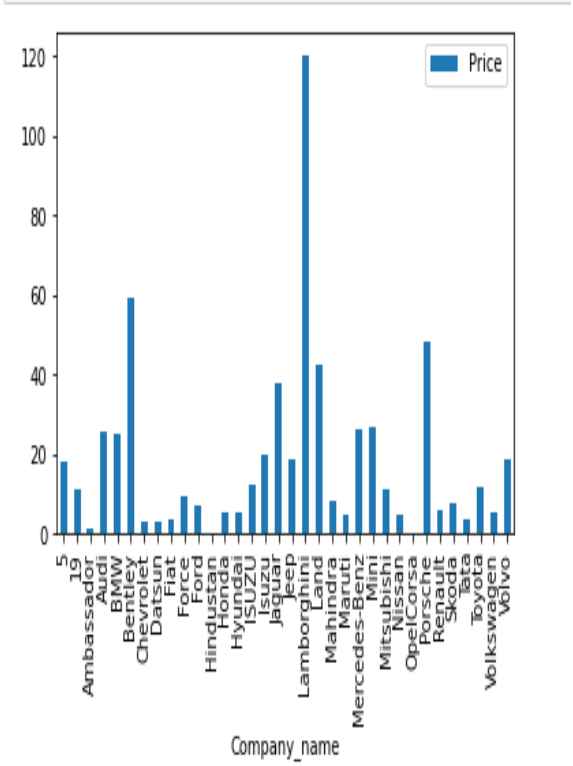
```
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error
print("Multiple Linear regression gives R2 score",r2_score(y_pred,y_test))
print('Multiple Linear regression gives MSE is:',mean_squared_error(y_test, y_pred))
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print('Multiple Linear regression gives RMSE is:',rmse)
print("-----------------------------------------")
```

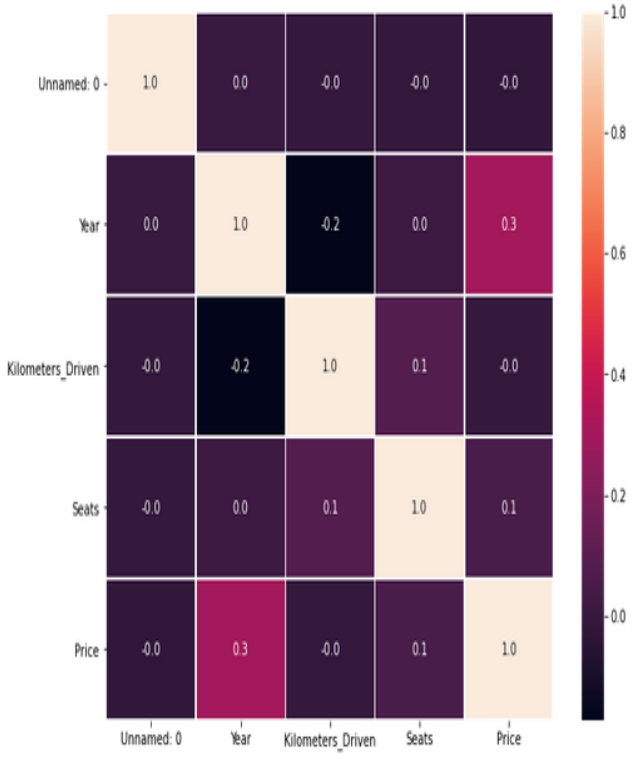
```
Multiple Linear regression gives R2 score 0.6311578376237479
Multiple Linear regression gives MSE is: 36.173476812650847
Multiple Linear regression gives RMSE is: 6.014439027250045
-----------------------------------------
```

```
In [688]: print("Random Forest Regression gives R2 score",r2_score(y_pred,y_test))
          print('Random Forest Regression gives MSE is:',mean_squared_error(y_test, y_pred))
          rmse = np.sqrt(mean_squared_error(y_test, y_pred))
          print('Random Forest Regression gives RMSE is:',rmse)
          print("-----------------------------------------")
```

```
Random Forest Regression gives R2 score 0.8056268010882469
Random Forest Regression gives MSE is: 21.804506925210085
Random Forest Regression gives RMSE is: 4.669526269144
-----------------------------------------
```

Visualization Snippets:





Conclusion/ Results

Random forest regressor is the best fit to this dataset, gives rmse as 4.66952. The Multiple Linear Regression also performed very well with good rmse.

Future Scope

In future, better Regression techniques can be used and the models can be upgraded which can give more accurate results.