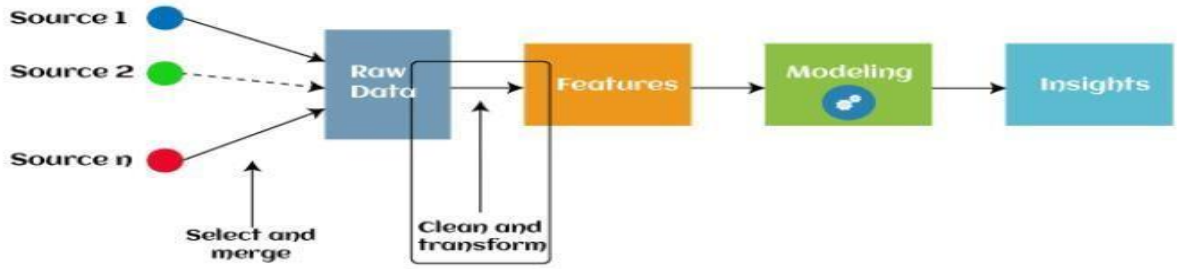
CREDIT CARD FRAUD DETECTION

DEVELOPMENT PART-2

FEATURE ENGINEERING

Feature engineering is the pre-processing step of machine learning, which extracts features from raw data. It helps to represent an underlying problem to predictive models in a better way, which as a result, improve the accuracy of the model for unseen data. The predictive model contains predictor variables and an outcome variable, and while the feature engineering process selects the most useful predictor variables for the model.



► **Feature Engineering Techniques**

- ◉ Imputation
- ◉Handling Outliers
- ◉Log transform
- ◉Binning

1.Imputation

Feature engineering deals with inappropriate data, missing values, human interruption, general errors, insufficient data sources, etc. Missing values within the dataset highly affect the performance of the algorithm, and to deal with them “Imputation” technique is used. Imputation is responsible for handling irregularities within the dataset.

```
import pandas as pd
df=pd.read_csv('/content/creditcard.csv',usecols=['Time','Amount','Class'])
df.head()
```

	Time	Amount	Class
0	0	149.62	0.0
1	0	2.69	0.0
2	1	378.66	0.0
3	1	123.50	0.0
4	2	69.99	0.0

```
df.isnull().sum()
```

```
Time      0
Amount    1
Class     1
dtype: int64
```

```
median=df.Amount.median()
median
```

```
12.99
```

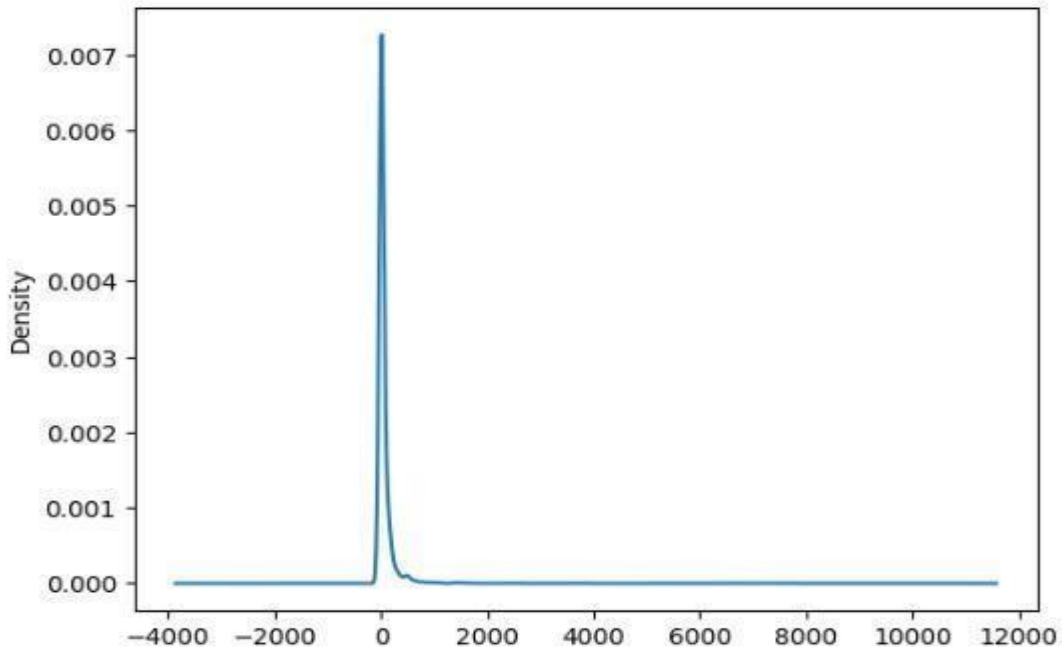
```
def impute_nan(df,variable,median):
  df[variable+"_median"]=df[variable].fillna(median)
  df[variable+"_random"]=df[variable]
  random_sample=df[variable].dropna().sample(df[variable].isnull().sum().random_sample==0)
  random_sample.indeex=[df[variable].isnull()].index
  df.loc[df[variable].isnull(),variable+'_random']=random_sample
```

```
median=df.Amount.median
median
```

```
<bound method
NDFrame._add_numeric_operations.
<locals>.median of 0        149.62
1            2.69
2          378.66
3          123.50
4           69.99
           ...
3968        13.99
3969        27.43
3970       730.32
3971         6.87
3972          NaN
Name: Amount, Length: 3973, dtype:
float64>
```

```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
fig=plt.figure()
ax=fig.add_subplot(111)
df['Amount'].plot(kind='kde',ax=ax)
df.Amount_random=plot(kind='kde',ax=ax,color='green')
lines,labels=ax.get_legend_handles_labels()
ax.legend(lines,labels,loc='best')
```



2.Handling Outliers

Standard deviation can be used to identify the outliers. For example, each value within a space has a definite to an average distance, but if a value is greater distant than a certain value, it can be considered as an outlier. Z-score can also be used to detect outliers.

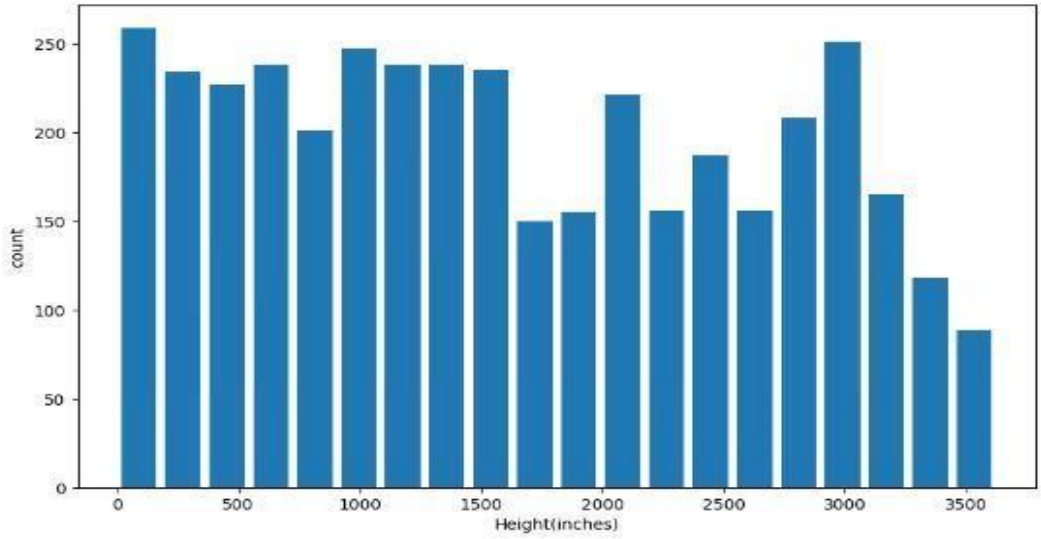
```
from ast import increment_lineno
import pandas
import matplotlib
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
matplotlib.rcParams['figure.figsize']=(10,6)
```

```
df=pd.read_csv("/content/creditcard.csv")
df.sample(5)
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...
2659	2197	-2.335963	1.457168	0.978213	1.489269	-0.840717	2.219489	-1.352552	2.215092	0.572527	...
3740	3227	-0.356490	0.778361	-0.035769	-1.212109	2.447476	3.372709	0.129061	0.935836	-0.522625	...
1684	1308	-1.322313	0.863591	1.844234	1.089375	0.771005	0.152191	1.491514	-0.917308	0.765538	...
1979	1522	-0.892709	0.134810	0.952378	-2.111891	0.372398	-0.831994	1.184740	-0.237214	0.428697	...
3592	3071	-0.302521	0.447474	-0.495757	-3.214799	2.705041	2.762440	0.593847	0.556451	0.697720	...

5 rows × 31 columns

```
plt.hist(df.Time,bins=20,rwidth=0.8)
plt.xlabel('Height(inches)')
plt.ylabel('count')
plt.show()
```



3.Log transform

Logarithm transformation or log transform is one of the commonly used mathematical techniques in machine learning. Log transform helps in handling the skewed data, and it makes the distribution more approximate to normal after transformation. It also reduces the effects of outliers on the data, as because of the normalization of magnitude differences, a model becomes much robust.

```
import numpy as np
data['log_median_income']=np.log((data.Amount))
data.log_median_income.head()
```

```
/usr/local/lib/python3.10/dist-packages/pandas/core/arraylike.py:402
  result = getattr(ufunc, method)(*inputs, **kwargs)
0    5.008099
1    0.989541
2    5.936639
3    4.816241
4    4.248352
Name: log_median_income, dtype: float64
```

4.Binning

In machine learning, overfitting is one of the main issues that degrade the performance of the model and which occurs due to a greater number of parameters and noisy data. However, one of the popular techniques of feature engineering, “binning”, can be used to normalize the noisy data. This process involves segmenting different features into bins.

```
from numpy.lib import imag
import pandas as pd
from matplotlib import pyplot as plt
df=pd.read_csv('/content/creditcard.csv')
data.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V22
0	0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	0.277838
1	0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.638672
2	1	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.771679
3	1	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	0.005274
4	2	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	0.798278

5 rows × 32 columns

```
bins=np.linspace(data.Amount.min(),data.Amount.max(),4)
bins
```

```
array([   0.   , 2570.81, 5141.62, 7712.43])
```

MODEL TRAINING

► **Testing data:** The testing set is a smaller portion of the data, usually around 20-30% of the dataset. It is kept separate and is not used during the model training phase. Instead, it is used to evaluate the model's performance by making predictions or performing analyses and comparing them to the actual, known outcomes.

► **Training data:**This subset contains a majority of the data, typically around 70-80% of the dataset. It is used to train machine learning models or perform data analysis tasks. The model learns patterns, relationships, and trends within the data from this set.

```
X = data.drop(['Class'], axis = 1)
Y = data["Class"]
print(X.shape)
print(Y.shape)


xData = X.values
yData = Y.values
```

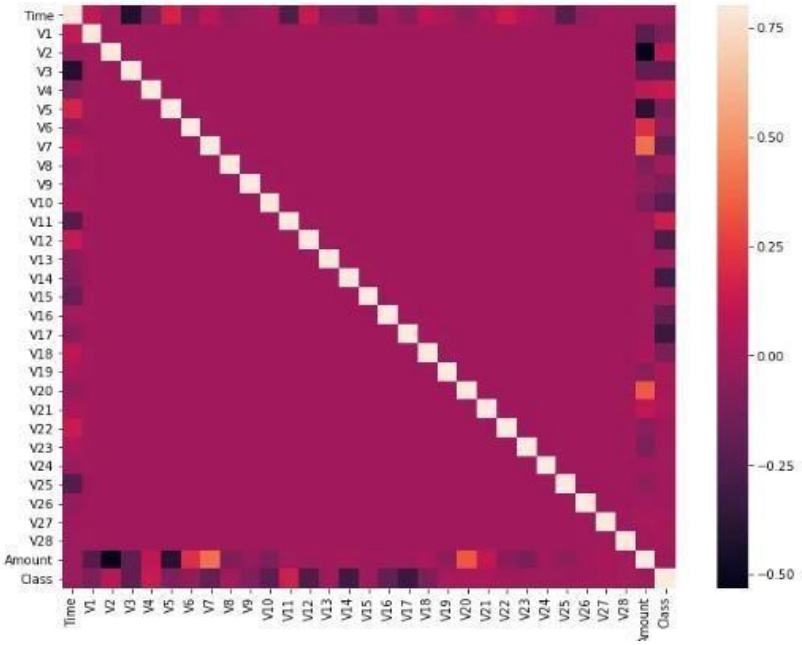
```
from sklearn.model_selection import train_test_split

xTrain, xTest, yTrain, yTest = train_test_split(
        xData, yData, test_size = 0.2, random_state = 42)
```

Evaluation

► Correlation matrix

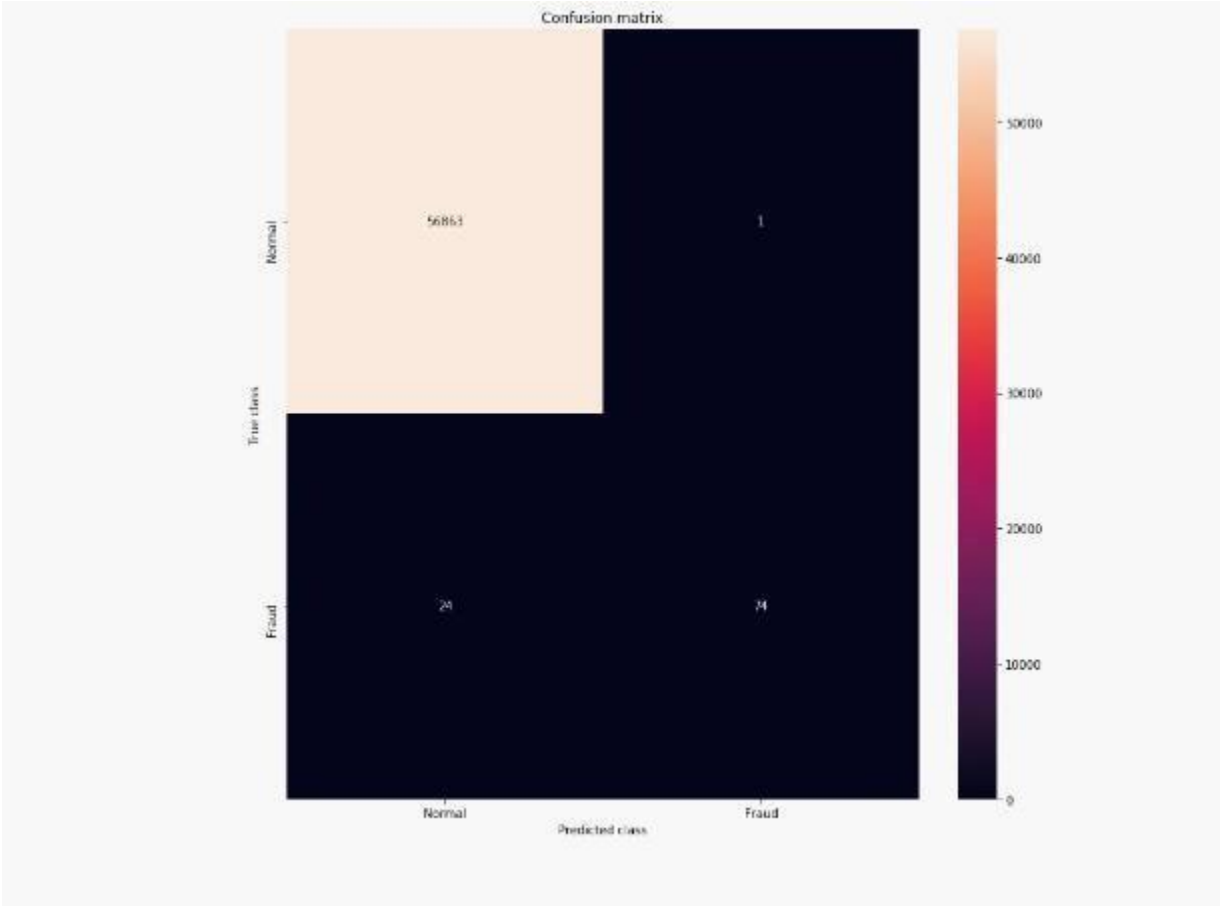
The correlation matrix graphically gives us an idea of how features correlate with each other and can help us predict what are the features that are most relevant for the prediction.

```
corrmat = data.corr()
fig = plt.figure(figsize = (12, 9))
sns.heatmap(corrmat, vmax = .8, square = True)
plt.show()
```



► Visualizing the Confusion Matrix

```
LABELS = ['Normal', 'Fraud']
conf_matrix = confusion_matrix(yTest, yPred)
plt.figure(figsize =(12, 12))
sns.heatmap(conf_matrix, xticklabels = LABELS,
            yticklabels = LABELS, annot = True, fmt ="d");
plt.title("Confusion matrix")
plt.ylabel('True class')
plt.xlabel('Predicted class')
plt.show()
```



So this is how we can evaluate a machine learning model and perform the feature engineering for credit card fraud detection.