Problem Statement

- To predict the best fit model for predicting fraudulent activities in credit card dataset.

Data Description

- The dataset consists of 24 columns:
- Independent Variable: Limit balance Sex, Education,Marriage, Age, Pay and Bill
- Dependent Variable: Default Payment (y)
- There are no Missing Variable

```
dt.head()
```

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	y
0	20000	2	2	1	24	2	2	-1	-1	-2	...	0	0	0	0	689	0	0	0	0	1
1	120000	2	2	2	26	-1	2	0	0	0	...	3272	3455	3261	0	1000	1000	1000	0	2000	1
2	90000	2	2	2	34	0	0	0	0	0	...	14331	14948	15549	1518	1500	1000	1000	1000	5000	0
3	50000	2	2	1	37	0	0	0	0	0	...	28314	28959	29547	2000	2019	1200	1100	1069	1000	0
4	50000	1	2	1	57	-1	0	-1	0	0	...	20940	19146	19131	2000	36681	10000	9000	689	679	0

5 rows × 24 columns

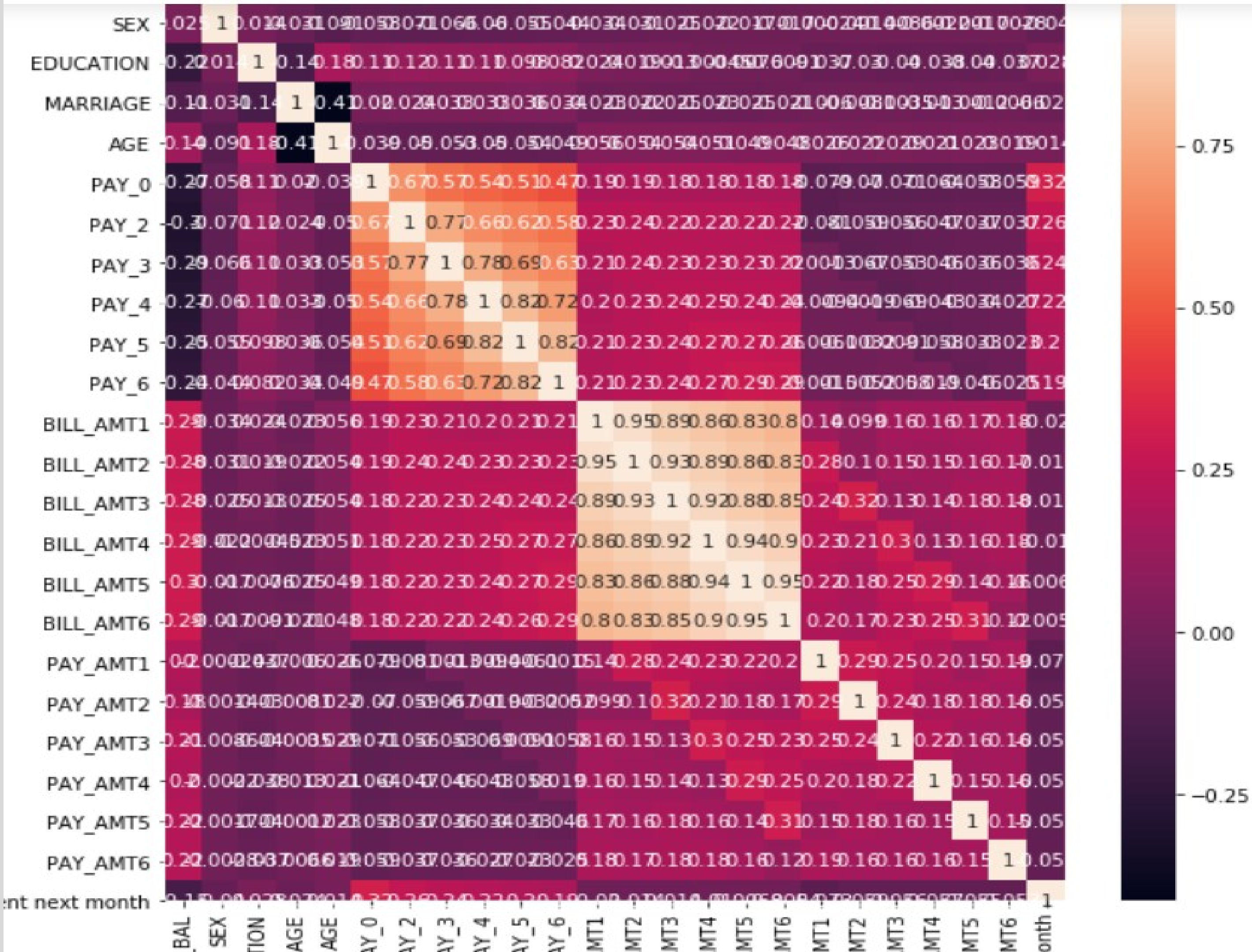
No	Attribute Name	Description
1	ID	User ID
2	Limit_Bal	Amount of the given credit (NT dollar)
3	Sex	Gender(1=male,2=female)
4	Education	Education (1=graduate school,2=University,3=others)
5	Marriage	Marital status(1=married,2=unmarried)
6	Age	Age(year)
7	Pay_0	the repayment status in September, 2005
8	Pay_2	the repayment status in August, 2005
9	Pay_3	the repayment status in July, 2005
10	Pay_4	the repayment status in June, 2005
11	Pay_5	the repayment status in May, 2005
12	Pay_6	the repayment status in April, 2005
13	Bill_Amt1	amount of bill statement in September, 2005
14	Bill_Amt2	amount of bill statement in August, 2005
15	Bill_Amt3	amount of bill statement in July, 2005
16	Bill_Amt4	amount of bill statement in June, 2005
17	Bill_Amt5	amount of bill statement in May, 2005
18	Bill_Amt6	amount of bill statement in April, 2005
19	Pay_Amt1	amount paid in April, 2005
20	Pay_Amt2	amount paid in May, 2005
21	Pay_Amt3	amount paid in June, 2005
22	Pay_Amt4	amount paid in July, 2005
23	Pay_Amt5	amount paid in August, 2005
24	Pay_Amt6	amount paid in September, 2005
25	Default	Amount to be paid next month

CORRELATION

Heat map is plotted to check how
well the variables are correlated
with each other.

We can see that Education and Repayment status represents strong negative
correlation with Amount of credit
Given (LIMIT_BAL)

Each Variables of Billing statement: BILL_AMT1, BILL_AMT6 are positively correlated among each other.



Scaling - Standard Scaler

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	...	BILL_AMT4	BILL_AMT5	BILL_AMT6
0	-1.136720	0.810161	0.185828	-1.057295	-1.246020	1.794564	1.782348	-0.696663	-0.666599	-1.530046	...	-0.672497	-0.663059	-0.652724
1	-0.365981	0.810161	0.185828	0.858557	-1.029047	-0.874991	1.782348	0.138865	0.188746	0.234917	...	-0.621636	-0.606229	-0.597966
2	-0.597202	0.810161	0.185828	0.858557	-0.161156	0.014861	0.111736	0.138865	0.188746	0.234917	...	-0.449730	-0.417188	-0.391630
3	-0.905498	0.810161	0.185828	-1.057295	0.164303	0.014861	0.111736	0.138865	0.188746	0.234917	...	-0.232373	-0.186729	-0.156579
4	-0.905498	-1.234323	0.185828	-1.057295	2.334029	-0.874991	0.111736	-0.696663	0.188746	0.234917	...	-0.346997	-0.348137	-0.331482

5 rows × 24 columns

The Standard Scalar transforms the data such that each value will have a mean 0 and Standard Deviation 1.

PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	default payment next month
-0.341942	-0.227086	-0.296801	-0.308063	-0.314136	-0.293382	1
-0.341942	-0.213588	-0.240005	-0.244230	-0.314136	-0.180878	1
-0.250292	-0.191887	-0.240005	-0.244230	-0.248683	-0.012122	0
-0.221191	-0.169361	-0.228645	-0.237846	-0.244166	-0.237130	0
-0.221191	1.335034	0.271165	0.266434	-0.269039	-0.255187	0

Outlier(After Scaling)

- Outliers are extreme values that deviate from other observations on data
- Z score - Since 99.73 data points lie within +/- 3 Standard Deviation. *if Z is greater than 3* then that value is considered as Outliers

```
#Removing outlier - z_score
from scipy import stats
z = np.abs(stats.zscore(d1))
print(z)
```

```
[[1.13672015 0.81016074 0.18582826 ... 0.31413612 0.29338206 1.87637834]
 [0.3659805  0.81016074 0.18582826 ... 0.31413612 0.18087821 1.87637834]
 [0.59720239 0.81016074 0.18582826 ... 0.24868274 0.01212243 0.53294156]
 ...
 [1.05964618 1.23432296 0.18582826 ... 0.18322937 0.11900109 1.87637834]
 [0.67427636 1.23432296 1.45111372 ... 3.15253642 0.19190359 1.87637834]
 [0.90549825 1.23432296 0.18582826 ... 0.24868274 0.23713013 1.87637834]]
```

```
#Removing outliers
threshold = 3
d1.shape
```

```
(30000, 24)
```

```
#creating new dataframe after remmoving outliers
d2 = d1[(z<3).all(axis=1)]
d2.shape
```

```
(26429, 24)
```

Outlier Detection(Without Scaling)

```
from scipy import stats
z = np.abs(stats.zscore(df[['X1', 'X5', 'X12', 'X13', 'X14', 'X15', 'X16', 'X17', 'X18', 'X19', 'X20', 'X21', 'X22', 'X23' ]]))
print(z)
```

```
[[1.13584859 1.24613545 0.64373456 ... 0.30844347 0.31452581 0.29374281]
 [0.36525716 1.02923602 0.66044213 ... 0.24467821 0.31452581 0.1813593 ]
 [0.59643459 0.16163827 0.30000105 ... 0.24467821 0.24914153 0.01278404]
 ...
 [1.05878945 0.16371088 0.64845774 ... 0.04062937 0.18375725 0.11954838]
 [0.67349373 0.59750975 0.71916972 ... 0.18563158 3.14848722 0.19237289]
 [0.90467116 1.13975835 0.0463337  ... 0.24467821 0.24914153 0.23755106]]
```

```
threshold = 3
print(np.where(z > 3))
```

```
(array([    6,     6,     6, ..., 29923, 29928, 29928], dtype=int64), array([ 2,  3,  4, ...,  9,  8, 12], dtype=int64))
```

```
df.shape
```

```
(29930, 24)
```

```
df_new = df[(z < 3).all(axis=1)]
```

```
df_new.shape
```

```
(27007, 24)
```

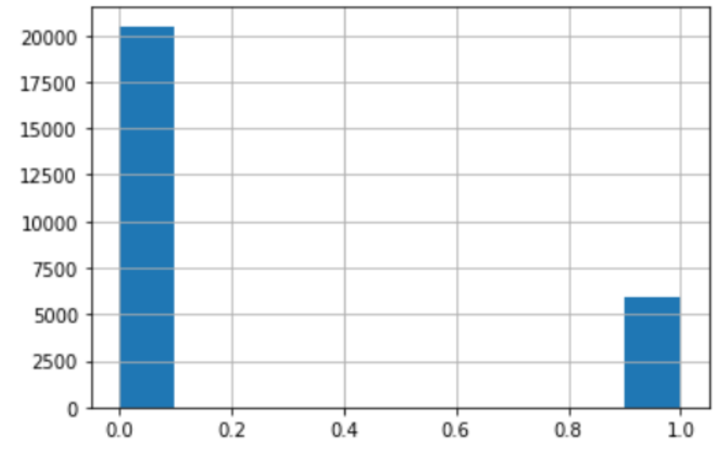
Smote - Imbalanced dataset

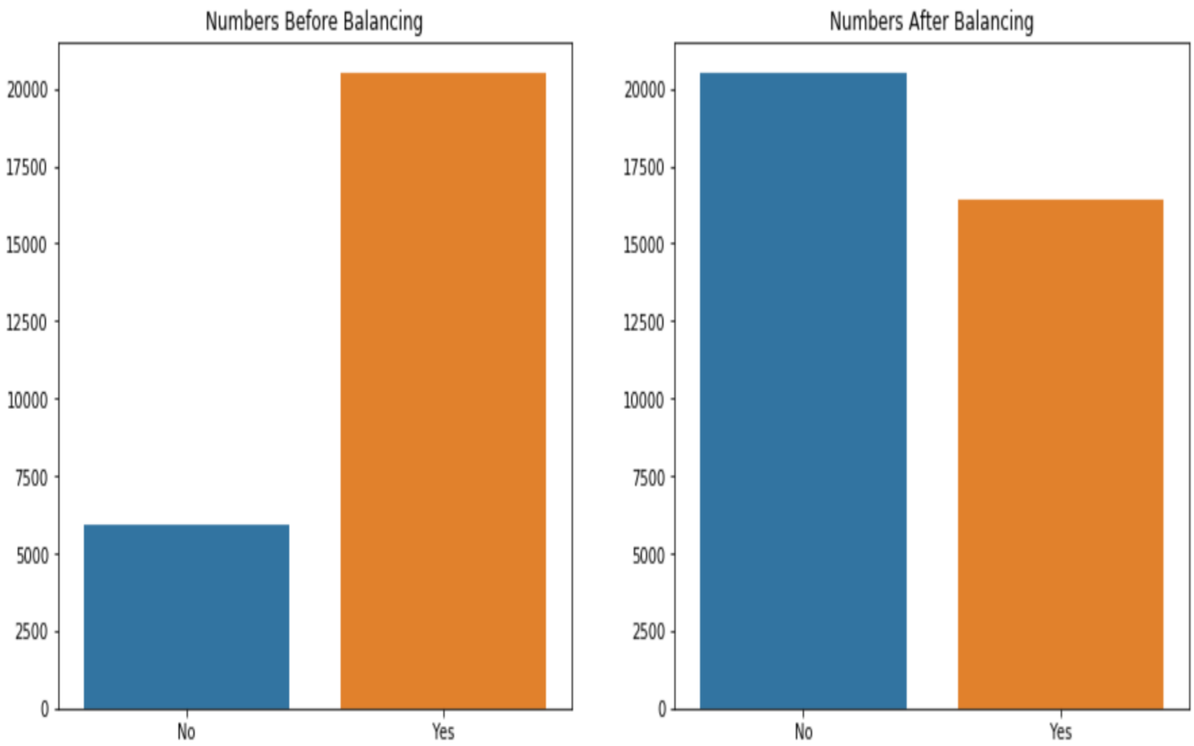
(Synthetic Minority Oversampling Technique)

```
{1: 5935, 0: 20494}
{0: 20493, 1: 16395}
```

```
#To check whether data is balanced or not
y1.hist()  #seen imbalanced data
```

```
<AxesSubplot:>
```





Variable exclusion - LOGIT I

- To check for insignificant variables and remove them

This is the **final output** after removing the necessary insignificant data and thus getting all the variables that are significant.

Since the *LLR **p-value is lesser than 0.05 and all variables are significant***, the model is deployable and these variables could be used in implementation of further techniques.

Logit Regression Results

Dep. Variable:	default payment next month	**No. Observations:**	36888
Model:	Logit	**Df Residuals:**	36872
Method:	MLE	**Df Model:**	15
Date:	Tue, 13 Oct 2020	**Pseudo R-squ.:**	0.08817
Time:	23:18:11	**Log-Likelihood:**	-23106.
converged:	True	**LL-Null:**	-25341.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
LIMIT_BAL	-0.0743	0.015	-4.879	0.000	-0.104	-0.044
SEX	-0.0632	0.011	-5.613	0.000	-0.085	-0.041
EDUCATION	-0.0288	0.014	-2.121	0.034	-0.055	-0.002
MARRIAGE	-0.0747	0.012	-6.019	0.000	-0.099	-0.050
AGE	0.0595	0.013	4.556	0.000	0.034	0.085
PAY_0	0.5741	0.015	38.184	0.000	0.545	0.604
PAY_2	0.1737	0.019	9.138	0.000	0.136	0.211
PAY_3	0.0476	0.021	2.261	0.024	0.006	0.089
PAY_4	0.0420	0.020	2.091	0.037	0.003	0.081
PAY_6	-0.0378	0.017	-2.232	0.026	-0.071	-0.005
BILL_AMT1	-0.6364	0.050	-12.611	0.000	-0.735	-0.537
BILL_AMT3	0.2276	0.065	3.487	0.000	0.100	0.356
BILL_AMT6	0.1702	0.040	4.286	0.000	0.092	0.248
PAY_AMT1	-0.1001	0.039	-2.557	0.011	-0.177	-0.023
PAY_AMT2	-0.2517	0.051	-4.977	0.000	-0.351	-0.153
PAY_AMT6	0.1975	0.039	5.030	0.000	0.121	0.274

Variable exclusion - Principal Component Analysis II

```
#PCA for feature selection and reduce dimensionality
#Here instead of mentioning n_components, we will directly mention the variance we would like PCA to capture
from sklearn.decomposition import PCA
pca = PCA(0.85)
pca.fit(x_train)
PCA(n_components=0.85, random_state=None, whiten = False)
```

```
PCA(n_components=0.85)
```

```
#Checking for number of components that would provide with 85% variance
pca.n_components_
```

```
8
```

```
#Just to cross check whether 8 components would be enough to explain major variance in the data
pca = PCA(n_components=8)
fit = pca.fit(x)
print("Explained Variance: %s" % fit.explained_variance_ratio_)
print(fit.components_)
```

```
Explained Variance: [0.36959408 0.15302819 0.10437215 0.0724056  0.05658796 0.04980261
 0.03840171 0.03293198]
```

- Applied PCA, removed the variables that were insignificant according to the given here. (Having least variance among each other)
- After transforming the dataset, different techniques were applied using the rest of the significant variables.

Techniques - Logistic Regression (After LOGIT)

- Applied on the significant variables derived above.

- **Model Assumptions:**

- Target variable is binary
- Predictive features are interval (continuous) or categorical
- Features are independent of one another
- Sample size is adequate

- **Accuracy : 0.694**
- **F1 - Score : 0.624**

```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
cm_lr = confusion_matrix(y_test, y_pred_test)
acc_lr = accuracy_score(y_test, y_pred_test)
pre_lr = precision_score(y_test, y_pred_test)
recall_lr = recall_score(y_test, y_pred_test)
f1_lr = f1_score(y_test, y_pred_test)

print('Confusion Matrix : \n',cm_lr)
print('Accuracy : ',acc_lr)
print('Precision score', pre_lr)
print('Recall score',recall_lr)
print('F1 score',f1_lr)
```

```
Confusion Matrix :
 [[4051 1086]
 [1738 2347]]
Accuracy :  0.6937757536326177
Precision score 0.683658607631809
Recall score 0.5745410036719706
F1 score 0.624368183027401
```

Techniques - Logistic Regression (After PCA)

- **Accuracy : 0.6025**
- **F1 - Score : 0.3121**

```python
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
cm_lr = confusion_matrix(y_test, y_pred_test)
acc_lr = accuracy_score(y_test, y_pred_test)
pre_lr = precision_score(y_test, y_pred_test)
recall_lr = recall_score(y_test, y_pred_test)
f1_lr = f1_score(y_test, y_pred_test)

print('Confusion Matrix : \n',cm_lr)
print('Accuracy : ',acc_lr)
print('Precision score', pre_lr)
print('Recall score',recall_lr)
print('F1 score',f1_lr)
```

```
Confusion Matrix :
 [[4801  445]
 [3279  845]]
Accuracy :  0.6025613660618997
Precision score 0.6550387596899225
Recall score 0.20489815712900097
F1 score 0.31215367565570745
```

Techniques - Decision Tree (After LOGIT)

- Similar approach in case of decision tree.

- **Accuracy : 0.746**
- **F1 - score : 0.718**

- Though better than that in the case of logistic regression, but will try to impute the improvised version of the same - Random Forest to check if the accuracy is increasing or not

```
Confusion Matrix :
 [[3890 1247]
 [1098 2987]]
Accuracy :  0.745716764259379
Precision score :  0.705479452054794 6
Recall score :  0.731211750305997 5
F1 score :  0.718115158071883 6
```

Techniques - Decision Tree (After PCA)

- Similar approach in case of decision tree.

- **Accuracy : 0.7239**
- **F1 - score : 0.6911**

```
Confusion Matrix :
 [[3889 1357]
 [1230 2894]]
Accuracy :  0.723906083244397
Precision score :  0.6807809927075982
Recall score :  0.7017458777885548
F1 score :  0.6911044776119403
```

Techniques - Random Forest Classifier(After LOGIT)

- Being improvised version of Decision Tree, it resulted in deriving **higher** accuracy and F1 score.

- **Accuracy : 0.829**
- **F1 Score : 0.797**

```
Confusion Matrix :
 [[4554  583]
 [ 994 3091]]
Accuracy :  0.8289958794187812
Precision score :  0.8413173652694611
Recall score :  0.7566707466340269
F1 score :  0.7967521587833484
```

Techniques - Random Forest Classifier(After PCA)

- Being improvised version of Decision Tree, it resulted in deriving **higher** accuracy and F1 score.

- **Accuracy : 0.822**
- **F1 Score : 0.7862**

```
Confusion Matrix :
 [[4654  592]
 [1069 3055]]
Accuracy :  0.8227321237993597
Precision score :  0.8376748012064711
Recall score :  0.7407856450048497
F1 score :  0.786256595032814З
```

Techniques - Support Vector Machine (SVM)

AFTER LOGIT

- Another Powerful Machine Learning Algorithm which is used for both Classification and Regression.

- **Accuracy : 0.726**
- **F1 Score : 0.647**

```
Confusion Matrix :
 [[4384  753]
 [1771 2314]]
Accuracy :  0.7263066579917589
Precision score :  0.7544832083469188
Recall score :  0.5664626682986537
F1 score :  0.6470917225950783
```

Techniques - Support Vector Machine (SVM)

AFTER PCA

- Since the target variable is **binary in nature**, we tried to implement SVM as it is a binary classifier.

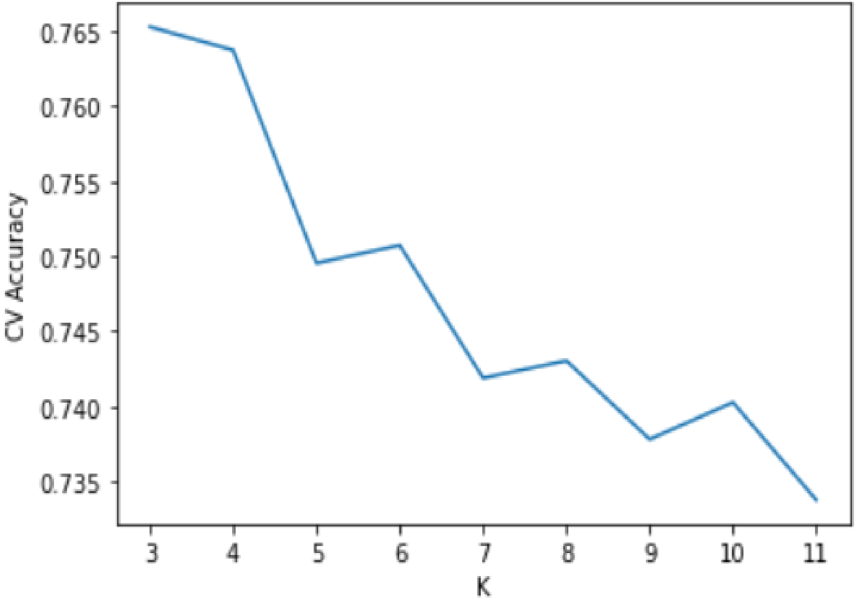
- **Accuracy : 0.63299**
- **F1 Score : 0.5545**

```
Confusion Matrix :
 [[3795 1451]
 [1985 2139]]
Accuracy :  0.6332977588046959
Precision score :  0.595821727019498б
Recall score :  0.5186711930164889
F1 score :  0.5545760954109411
```

Techniques - K-Nearest Neighbours (KNN): After LOGIT

```
Best K is : 3 | Cross validation Accuracy : 0.7653067306790259
```

```python
#Plotting CV accuracy curve for clarity
plt.plot(range(3,12),accList)  #Scores list
plt.xlabel('K')
plt.ylabel('CV Accuracy')
plt.show()
```

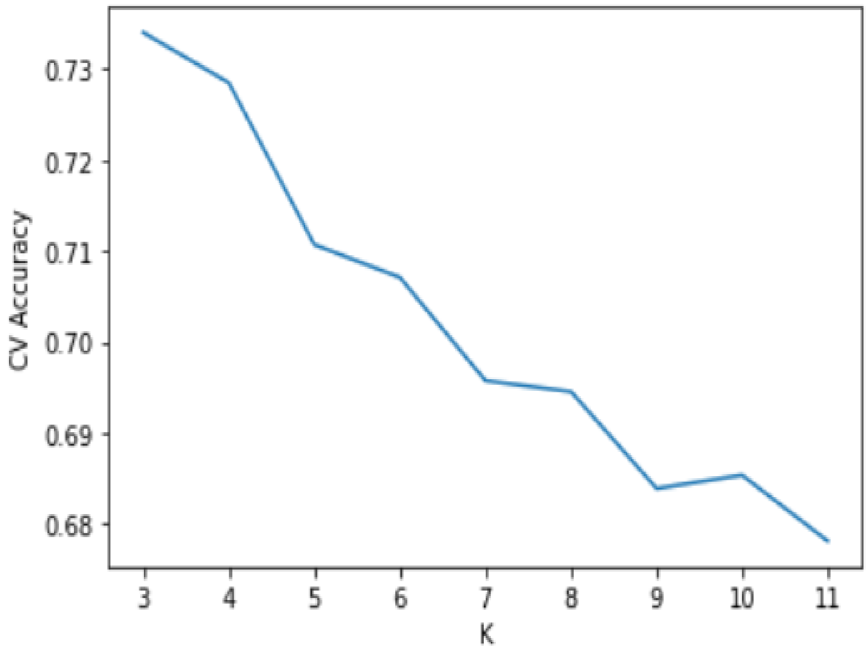



```
Confusion Matrix :
 [[3598 1539]
 [ 594 3491]]
Accuracy :  0.7687052700065062
Precision score :  0.6940357852882704
Recall score :  0.8545899632802938
F1 score :  0.7659901261656611
```

- Here, we have given a **range from 3-12** as **k,** in order **to estimate the best k** among the above which can derive better accuracy when compared.
- In this model, **k =3** turned out to be the best K, having the **highest CV accuracy** than others.
- The CV Accuracy plot clearly shows that as the number of K increases the CV accuracy falls.

Techniques - K-Nearest Neighbours (KNN): After PCA

```
#Plotting CV accuracy curve for clarity
plt.plot(range(3,12),accList)  #Scores list
plt.xlabel('K')
plt.ylabel('CV Accuracy')
plt.show()
```




```
Confusion Matrix :
 [[3527 1719]
 [ 738 3386]]
Accuracy :  0.7377801494130203
Precision score :  0.663271302644662
Recall score :  0.8210475266731329
F1 score :  0.733773973344891
```

- Here, we have given a **range from 3-12** as **k,** in order **to estimate the best k** among the above which can derive better accuracy when compared.
- In this model, **k =3** turned out to be the best K, having the **highest CV accuracy** than others.
- The CV Accuracy plot clearly shows that as the number of K increases the CV accuracy falls.

Conclusion - Best Model

LOGIT

Techniques	Accuracy	F1_Score
Logistic Regression	0.694	0.624
Decision Tree	0.742	0.716
Random Forest	0.831	0.8
SVM	0.725	0.645
KNN	0.781	0.777

PCA

Techniques	Accuracy	F1_Score
Logistic Regression	0.603	0.312
Decision Tree	0.724	0.691
Random Forest	0.822	0.786
SVM	0.633	0.555
KNN	0.738	0.733

Q&A

1. **Which other scalar techniques did you use, why did you only consider standard scalar?**

Ans. We used Min-Max scalar and Robust scalar apart from standard scalar. While applying Min-Max scalar we got an error that the smote data was not able to fit in the model. Hence, we used Standard scaler for standardizing smote data.
We also used Robust scaler after removing outliers using Isolation Forest.

2. Explain the correlation matrix

Ans. From the heat map we could make out that is Limit_Bal, which means the amount limit the credit card gets to spend on credit is highly negatively correlated with the default. Which means that higher the credit limit lower the chance of the customer to default. This is because your credit card limit increases on your ability to payback and the account balance you have so if you are a safer customer, then only you have higher credit limit and the chance of defaulting is low. Limit balance is the main factor affecting the default of the customer.

Business Applications of Fraud Detection Models in the Finance World

1. Predictive Modelling

Use Historical Behavioural information of known fraud to identify suspicious behaviour similar to previous fraud patterns.

2. Anamoly Detection

It helps to identify data points and observations that deviate from a dataset’s normal behaviour.

3. Natural Language Processing

NLP with Machine Learning is a win win combination used to detect fraud. It can be used for :

(a) Transaction Narration Analysis
(b) Company News and Sentiment Analysis
(c) Email Monitoring
(d) Financial Statement Analysis

THANK YOU