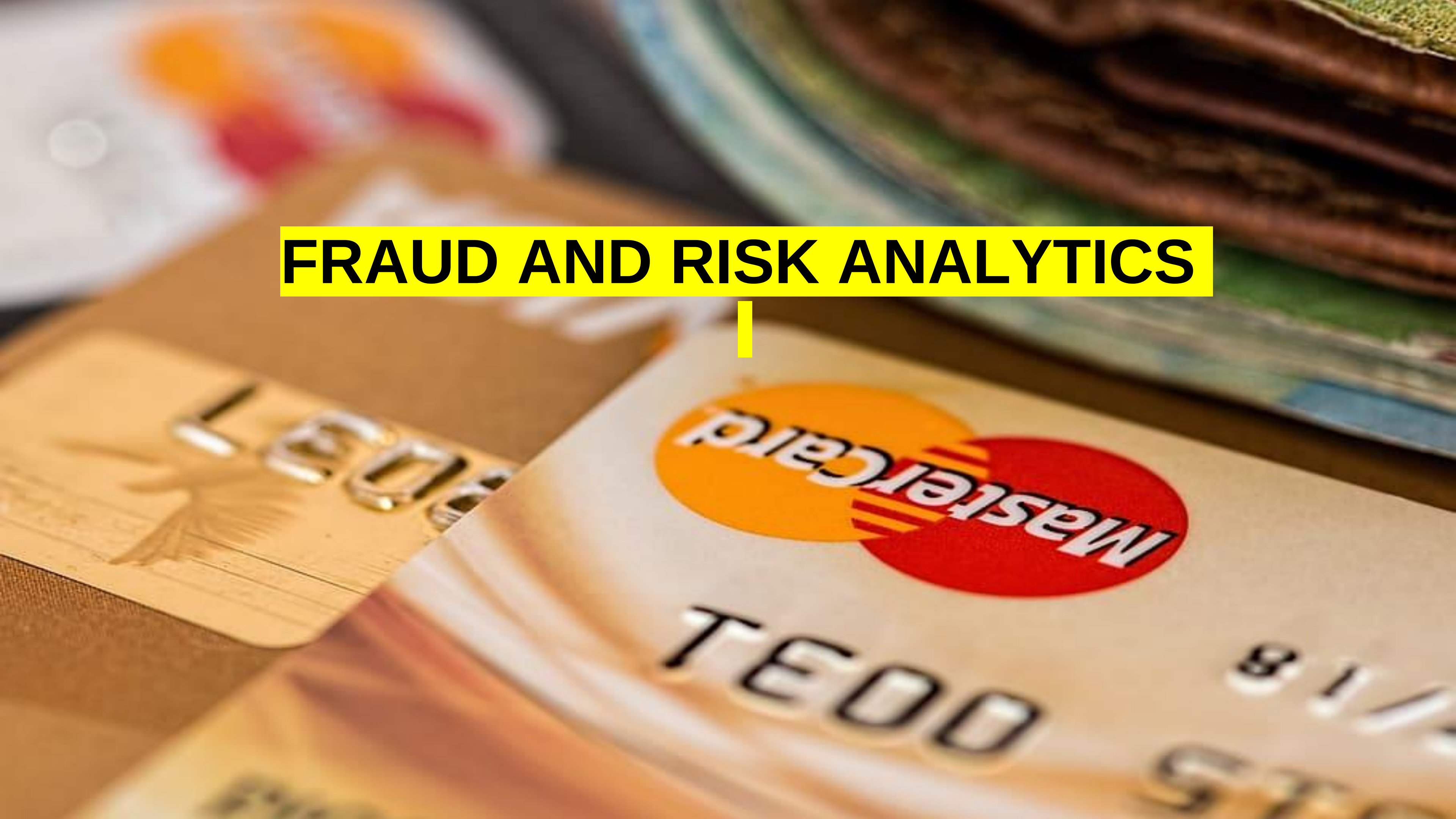
FRAUD AND RISK ANALYTICS



PROBLEM STATEMENT

To predict whether customers will default the payment next month.

DATA DESCRIPTION

```
print(data)

       LIMIT_BAL  SEX  EDUCATION  MARRIAGE  AGE  PAY_0  PAY_2  PAY_3  PAY_4  \
0          20000    2          2         1   24      2      2     -1     -1
1         120000    2          2         2   26     -1      2      0      0
2          90000    2          2         2   34      0      0      0      0
3          50000    2          2         1   37      0      0      0      0
4          50000    1          2         1   57     -1      0     -1      0
...          ...  ...        ...       ...  ...    ...    ...    ...    ...
29995     220000    1          3         1   39      0      0      0      0
29996     150000    1          3         2   43     -1     -1     -1     -1
29997      30000    1          2         2   37      4      3      2     -1
29998      80000    1          3         1   41      1     -1      0      0
29999      50000    1          2         1   46      0      0      0      0

       PAY_5  ...  BILL_AMT4  BILL_AMT5  BILL_AMT6  PAY_AMT1  PAY_AMT2  \
0         -2  ...          0          0          0         0       689
1          0  ...       3272       3455       3261         0      1000
2          0  ...      14331      14948      15549      1518      1500
3          0  ...      28314      28959      29547      2000      2019
4          0  ...      20940      19146      19131      2000     36681
...      ...  ...        ...        ...        ...       ...       ...
29995      0  ...      88004      31237      15980      8500     20000
29996      0  ...       8979       5190          0      1837      3526
29997      0  ...      20878      20582      19357         0         0
29998      0  ...      52774      11855      48944     85900      3409
29999      0  ...      36535      32428      15313      2078      1800

       PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6  default payment next month
0             0         0         0         0                           1
1          1000      1000         0      2000                           1
2          1000      1000      1000      5000                           0
3          1200      1100      1069      1000                           0
4         10000      9000       689       679                           0
...         ...       ...       ...       ...                         ...
29995      5003      3047      5000      1000                           0
29996      8998       129         0         0                           0
29997     22000      4200      2000      3100                           1
29998      1178      1926     52964      1804                           1
29999      1430      1000      1000      1000                           1

[30000 rows x 24 columns]
```

- This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005.
- This data set was extracted from the UCI machine learning Repository
- The data set contains 23 independent variables and 1 dependent variable.
- It has 30000 data points of various customers.
- The variables are limit balance, education, marriage, age, pay, bill amount , pay amount LIMIT_BAL,SEX, EDUCATION, MARRIAGE, AGE, PAY, BILL_AMT, PAY_AMT,default payment next month.

- Numerical Variables: 'Age,'LIMIT_BAL', 'PAY_0','PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2', 'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1', 'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6'
- Categorical variables: : ‘SEX', 'EDUCATION', 'MARRIAGE' , ‘DEFAULT PAYMENT NEXT MONTH’

VARIABLE DESCRIPTION

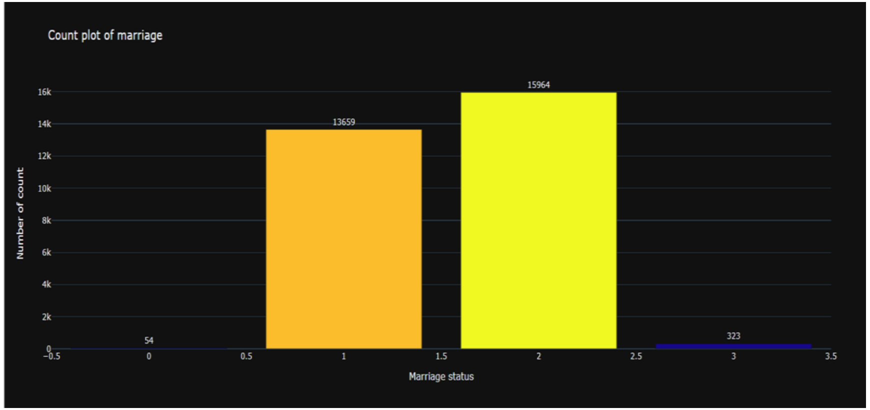
- ID: ID of each client
- LIMIT_BAL: Amount of given credit in NT dollars (includes individual and family/supplementary credit
- SEX: Gender (1=male, 2=female)
- EDUCATION: (1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown)
- MARRIAGE: Marital status (1=married, 2=single, 3=others)
- AGE: Age in years
- PAY_0: Repayment status in September, 2005 (-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, ... 8=payment delay for eight months, 9=payment delay for nine months and above)
- PAY_2: Repayment status in August, 2005 (scale same as above)
- PAY_3: Repayment status in July, 2005 (scale same as above)
- PAY_4: Repayment status in June, 2005 (scale same as above)
- PAY_5: Repayment status in May, 2005 (scale same as above)
- PAY_6: Repayment status in April, 2005 (scale same as above)
- BILL_AMT1: Amount of bill statement in September, 2005 (NT dollar)
- BILL_AMT2: Amount of bill statement in August, 2005 (NT dollar)
- BILL_AMT3: Amount of bill statement in July, 2005 (NT dollar)
- BILL_AMT4: Amount of bill statement in June, 2005 (NT dollar)
- BILL_AMT5: Amount of bill statement in May, 2005 (NT dollar)
- BILL_AMT6: Amount of bill statement in April, 2005 (NT dollar)
- PAY_AMT1: Amount of previous payment in September, 2005 (NT dollar)
- PAY_AMT2: Amount of previous payment in August, 2005 (NT dollar)
- PAY_AMT3: Amount of previous payment in July, 2005 (NT dollar)
- PAY_AMT4: Amount of previous payment in June, 2005 (NT dollar)
- PAY_AMT5: Amount of previous payment in May, 2005 (NT dollar)
- PAY_AMT6: Amount of previous payment in April, 2005 (NT dollar)
- default.payment.next.month: Default payment (1=yes, 0=no)

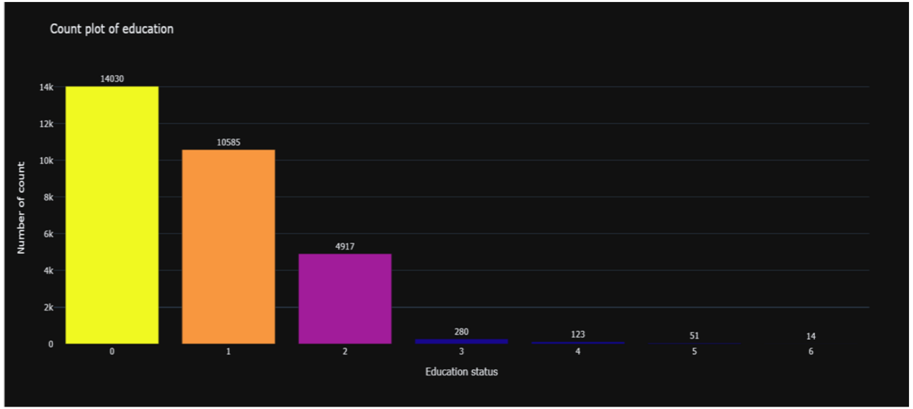
DATA PREPROCESSING

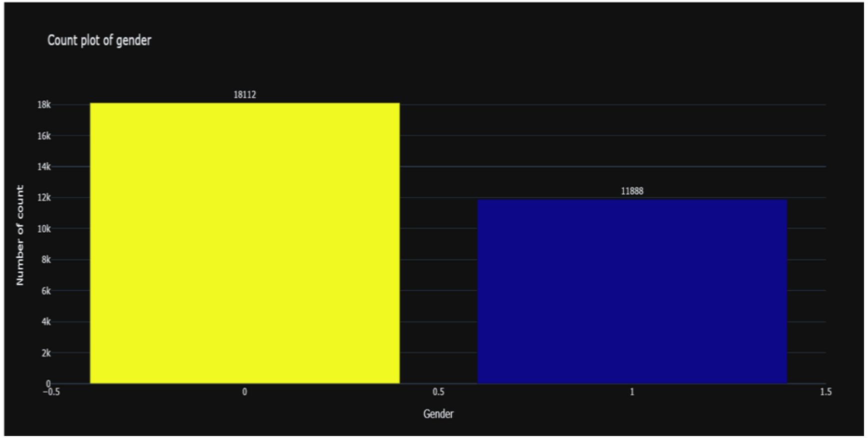
```
Out[5]: LIMIT_BAL                     0
        SEX                           0
        EDUCATION                     0
        MARRIAGE                      0
        AGE                           0
        PAY_0                         0
        PAY_2                         0
        PAY_3                         0
        PAY_4                         0
        PAY_5                         0
        PAY_6                         0
        BILL_AMT1                     0
        BILL_AMT2                     0
        BILL_AMT3                     0
        BILL_AMT4                     0
        BILL_AMT5                     0
        BILL_AMT6                     0
        PAY_AMT1                      0
        PAY_AMT2                      0
        PAY_AMT3                      0
        PAY_AMT4                      0
        PAY_AMT5                      0
        PAY_AMT6                      0
        default payment next month    0
        dtype: int64
```

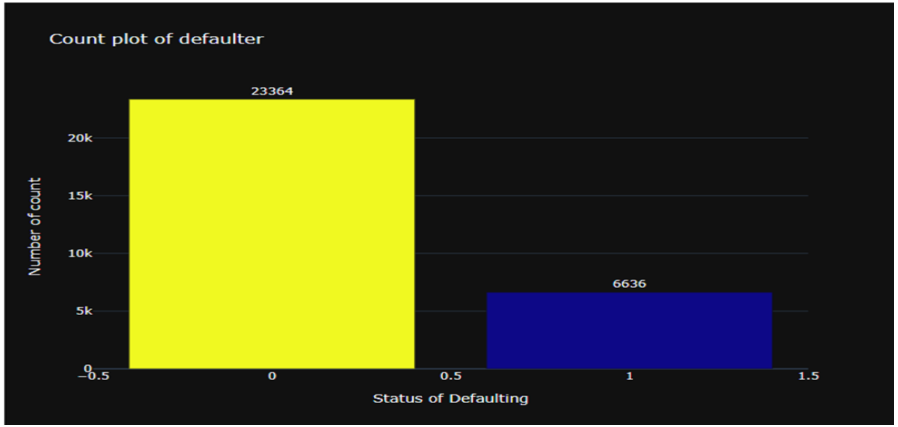
As we can see that there is no missing data, hence no preprocessing wrt Missing value analysis needs to be done

DATA VISUALIZATION









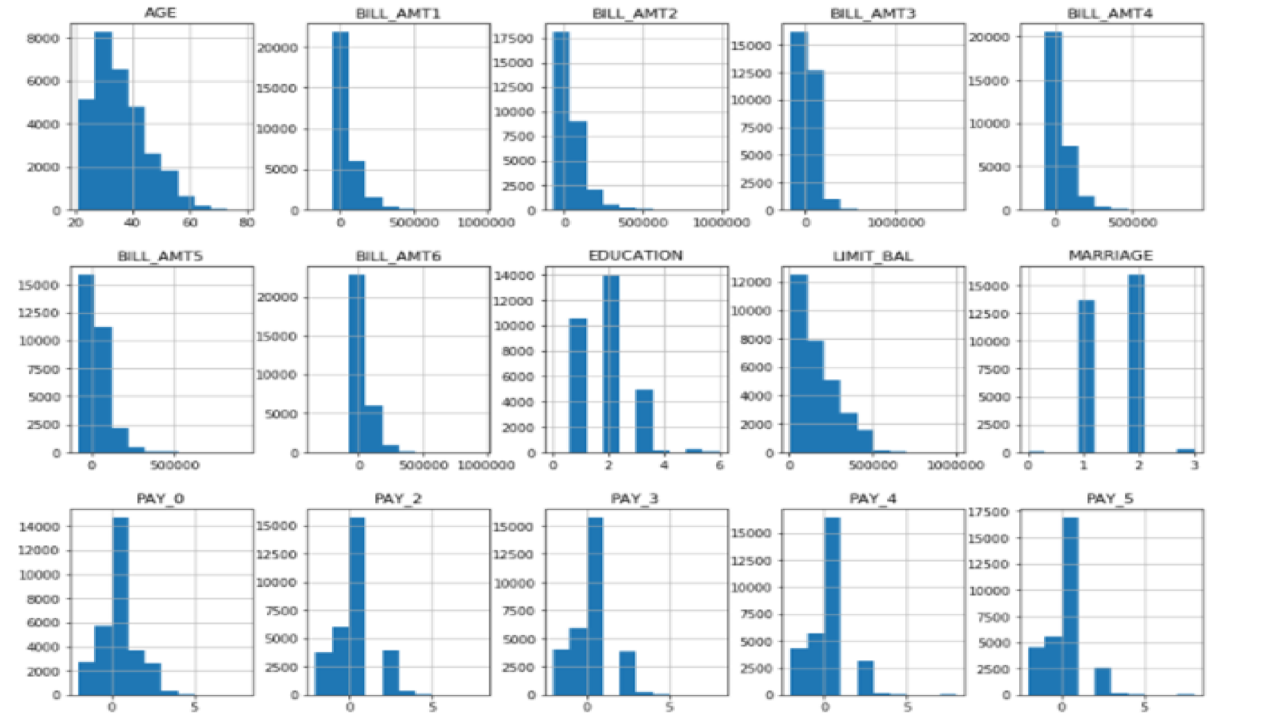
The graphs show the count plot of the Variables SEX, EDUCATION,MARRIAGE & DEFAULT PAYMENT NEXT MONTH

DESCRIPTIVE STATISTICS

	AGE	LIMIT_BAL	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	3.000000e+04	30000.000000	30000.000000
mean	35.485500	167484.322667	-0.016700	-0.133767	-0.166200	-0.220667	-0.266200	-0.291100	51223.330900	49179.075167	4.701315e+04	43262.948967	40311.400967
std	9.217904	129747.661567	1.123802	1.197186	1.196868	1.169139	1.133187	1.149988	73635.860576	71173.768783	6.934939e+04	64332.856134	60797.155770
min	21.000000	10000.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-165580.000000	-69777.000000	-1.572640e+05	-170000.000000	-81334.000000
25%	28.000000	50000.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	3558.750000	2984.750000	2.666250e+03	2326.750000	1763.000000
50%	34.000000	140000.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	22381.500000	21200.000000	2.008850e+04	19052.000000	18104.500000
75%	41.000000	240000.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	67091.000000	64006.250000	6.016475e+04	54506.000000	50190.500000
max	79.000000	1000000.000000	8.000000	8.000000	8.000000	8.000000	8.000000	8.000000	964511.000000	983931.000000	1.664089e+06	891586.000000	927171.000000

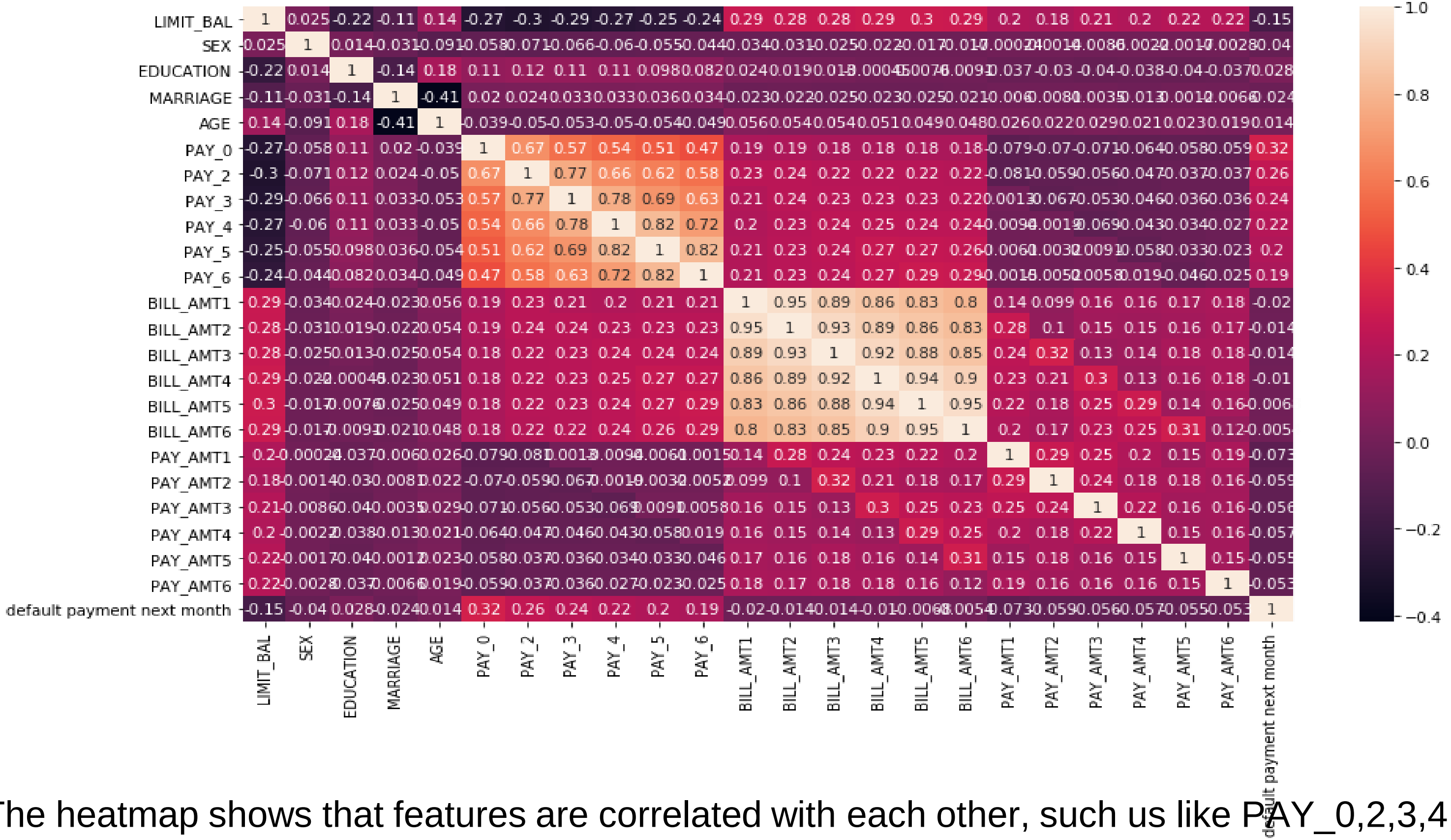
The above figure shows the descriptive statistics of the Numerical variables

HISTOGRAMS OF VARIABLES



Histogram of Bill Amounts & Payment is Highly skewed.
Age is skewed on the right hand side

HEAT-MAP & CORRELATION



The heatmap shows that features are correlated with each other, such us like PAY_0,2,3,4,5,6 and BILL_AMT1,2,3,4,5,6. In those cases, the correlation is positive.

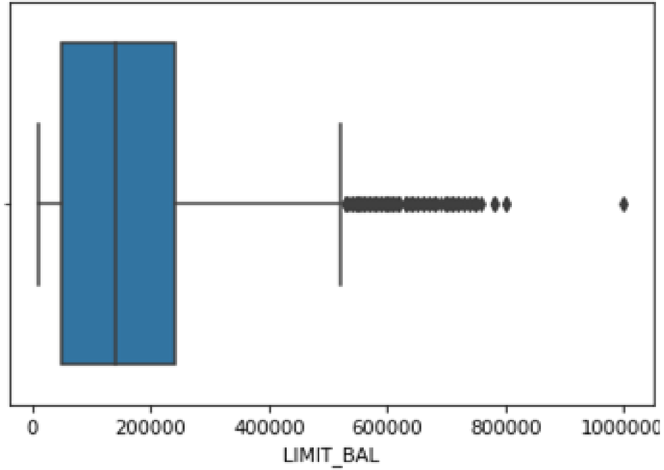
Check for Data Balance

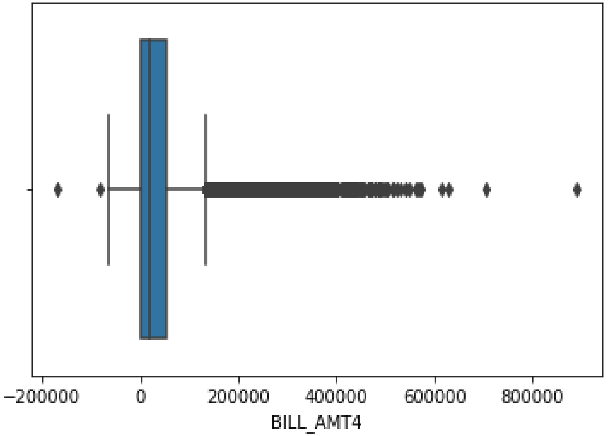
From the below result we can see that data is **imbalanced**.

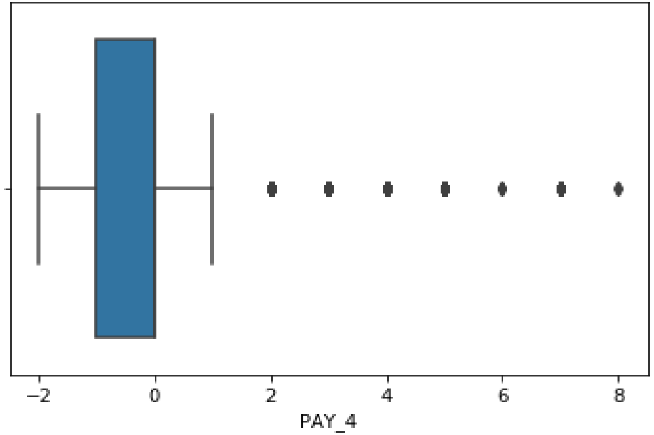
```
# The classes are heavily skewed we need to solve this issue later.
print('No default', round(data['default payment next month'].value_counts()[0]/len(data) * 100,2), '% of the dataset')
print('default', round(data['default payment next month'].value_counts()[1]/len(data) * 100,2), '% of the dataset')
```

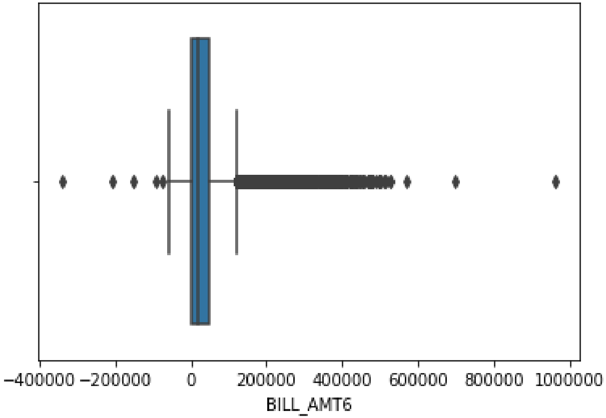
```
No default 77.88 % of the dataset
default 22.12 % of the dataset
```

OUTLIER Detection and Removal









Using Z scores to remove outliers

```
from scipy import stats
z = np.abs(stats.zscore(data[['LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0', 'PAY_2','PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2','BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_
print(z)
```

```
[[1.13672015 0.81016074 0.18582826 ... 0.30806256 0.31413612 0.29338206]
 [0.3659805  0.81016074 0.18582826 ... 0.24422965 0.31413612 0.18087821]
 [0.59720239 0.81016074 0.18582826 ... 0.24422965 0.24868274 0.01212243]
 ...
 [1.05964618 1.23432296 0.18582826 ... 0.03996431 0.18322937 0.11900109]
 [0.67427636 1.23432296 1.45111372 ... 0.18512036 3.15253642 0.19190359]
 [0.90549825 1.23432296 0.18582826 ... 0.24422965 0.24868274 0.23713013]]
```

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

```
(array([    6,     6,     6, ..., 29997, 29998, 29998], dtype=int64), array([11, 12, 13, ...,  5, 17, 21], dtype=int64))
```

```
data.shape
```

```
(30000, 24)
```

Feature Engineering

Feature engineering can be done using two methods:

1) Logistic Regression
2) PCA (Principle component Analysis)

As there are many features in the data, it impacts the accuracy thus insignificant features needs to be removed.

Feature Engineering- Logistic Regression

Logit Regression Results

Dep. Variable:	default payment next month	No. Observations:	26429
Model:	Logit	Df Residuals:	26406
Method:	MLE	Df Model:	22
Date:	Mon, 12 Oct 2020	Pseudo R-squ.:	0.1248
Time:	13:27:52	Log-Likelihood:	-12320.
converged:	True	LL-Null:	-14077.
Covariance Type:	nonrobust	LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
LIMIT_BAL	-5.038e-07	1.7e-07	-2.972	0.003	-8.36e-07	-1.72e-07
SEX	-0.1708	0.028	-5.997	0.000	-0.227	-0.115
EDUCATION	-0.1149	0.023	-4.962	0.000	-0.160	-0.070
MARRIAGE	-0.2416	0.024	-9.977	0.000	-0.289	-0.194
AGE	0.0016	0.001	1.083	0.279	-0.001	0.004
PAY_0	0.5749	0.019	30.688	0.000	0.538	0.612
PAY_2	0.1238	0.022	5.714	0.000	0.081	0.166
PAY_3	0.0596	0.025	2.423	0.015	0.011	0.108
PAY_4	0.0626	0.027	2.329	0.020	0.010	0.115
PAY_5	0.0217	0.029	0.749	0.454	-0.035	0.078
PAY_6	0.0003	0.024	0.014	0.989	-0.047	0.047
BILL_AMT1	-9.71e-06	1.65e-06	-5.872	0.000	-1.3e-05	-6.47e-06
BILL_AMT2	2.205e-06	2.29e-06	0.963	0.336	-2.28e-06	6.69e-06
BILL_AMT3	2.426e-06	2.1e-06	1.154	0.248	-1.69e-06	6.55e-06
BILL_AMT4	2.271e-06	2.08e-06	1.093	0.274	-1.8e-06	6.34e-06
BILL_AMT5	-4.452e-06	2.7e-06	-1.652	0.099	-9.74e-06	8.31e-07
BILL_AMT6	5.403e-06	2.23e-06	2.426	0.015	1.04e-06	9.77e-06
PAY_AMT1	-2.424e-05	4.71e-06	-5.148	0.000	-3.35e-05	-1.5e-05
PAY_AMT2	-2.932e-05	4.57e-06	-6.415	0.000	-3.83e-05	-2.04e-05
PAY_AMT3	-1.7e-05	4.41e-06	-3.853	0.000	-2.56e-05	-8.35e-06
PAY_AMT4	-9.1e-06	4.37e-06	-2.083	0.037	-1.77e-05	-5.39e-07
PAY_AMT5	-2.128e-05	4.84e-06	-4.393	0.000	-3.08e-05	-1.18e-05
PAY_AMT6	-9.29e-06	3.85e-06	-2.413	0.016	-1.68e-05	-1.74e-06

The logistic regression result shows that variables like AGE, PAY_5 ,PAY_6, BILL_AMT2, BILL_AMT3, BILL_AMT4 and BILL_AMT5 have **p value** greater than 0.05. Hence they are insignificant and are dropped.

```
#Removing values having p value greater than 0.05
X.drop(['AGE',"PAY_5","PAY_6","BILL_AMT2","BILL_AMT3","BILL_AMT4","BILL_AMT5"],1,inplace=True)
res = sm.Logit(y,X).fit()
res.summary()
```

Optimization terminated successfully.
Current function value: 0.466389
Iterations 7

Logit Regression Results

Dep. Variable:	default payment next month	**No. Observations:**	26429
Model:	Logit	**Df Residuals:**	26413
Method:	MLE	**Df Model:**	15
Date:	Mon, 12 Oct 2020	**Pseudo R-squ.:**	0.1244
Time:	13:27:56	**Log-Likelihood:**	-12326.
converged:	True	**LL-Null:**	-14077.
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
LIMIT_BAL	-4.838e-07	1.59e-07	-3.034	0.002	-7.96e-07	-1.71e-07
SEX	-0.1630	0.027	-5.970	0.000	-0.217	-0.110
EDUCATION	-0.1004	0.019	-5.222	0.000	-0.138	-0.063
MARRIAGE	-0.2390	0.024	-9.962	0.000	-0.286	-0.192
PAY_0	0.5779	0.019	30.911	0.000	0.541	0.615
PAY_2	0.1231	0.021	5.749	0.000	0.081	0.165
PAY_3	0.0640	0.024	2.620	0.009	0.016	0.112
PAY_4	0.0782	0.022	3.534	0.000	0.035	0.122
BILL_AMT1	-6.421e-06	6.67e-07	-9.624	0.000	-7.73e-06	-5.11e-06
BILL_AMT6	4.608e-06	8.32e-07	5.538	0.000	2.98e-06	6.24e-06
PAY_AMT1	-2.134e-05	4.4e-06	-4.853	0.000	-3e-05	-1.27e-05
PAY_AMT2	-2.726e-05	4.27e-06	-6.387	0.000	-3.56e-05	-1.89e-05
PAY_AMT3	-1.614e-05	4.04e-06	-3.998	0.000	-2.41e-05	-8.23e-06
PAY_AMT4	-1.292e-05	4.02e-06	-3.210	0.001	-2.08e-05	-5.03e-06
PAY_AMT5	-2.042e-05	4.48e-06	-4.560	0.000	-2.92e-05	-1.16e-05
PAY_AMT6	-9.578e-06	3.84e-06	-2.492	0.013	-1.71e-05	-2.05e-06

Removing variables with P value greater than 0.05

Feature Engineering- PCA

```
from sklearn.decomposition import PCA
```

```
x = df.drop(['default payment next month'],1)
y = df['default payment next month']
```

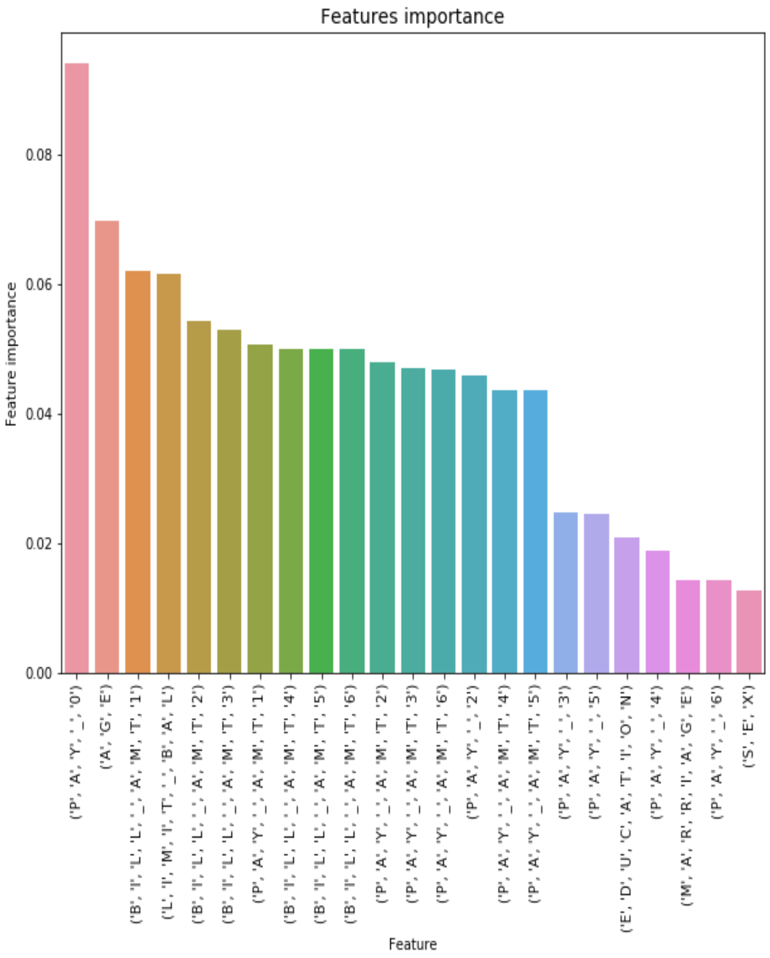
```
from sklearn.model_selection import train_test_split
X= preprocessing.StandardScaler().fit(X).transform(X)
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=1)
```

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 50)
classifier.fit(x_train, y_train)
```

```
RandomForestClassifier(n_estimators=50)
```

```
X_train_PCA = pca.fit_transform(x_train)
X_train_PCA = pd.DataFrame(data=X_train_PCA, index=x_train.index)
```

```
X_train_PCA_inverse = pca.inverse_transform(X_train_PCA)
X_train_PCA_inverse = pd.DataFrame(data=X_train_PCA_inverse, \
                                   index=x_train.index)
```



Removed components using PCA

```python
import statsmodels.api as sm
#Removing values less than 25%
x.drop(["SEX","PAY_4","PAY_6","EDUCATION","MARRIAGE"],1,inplace=True)
print(x)
```

Handling Imbalance Data set (SMOTE)

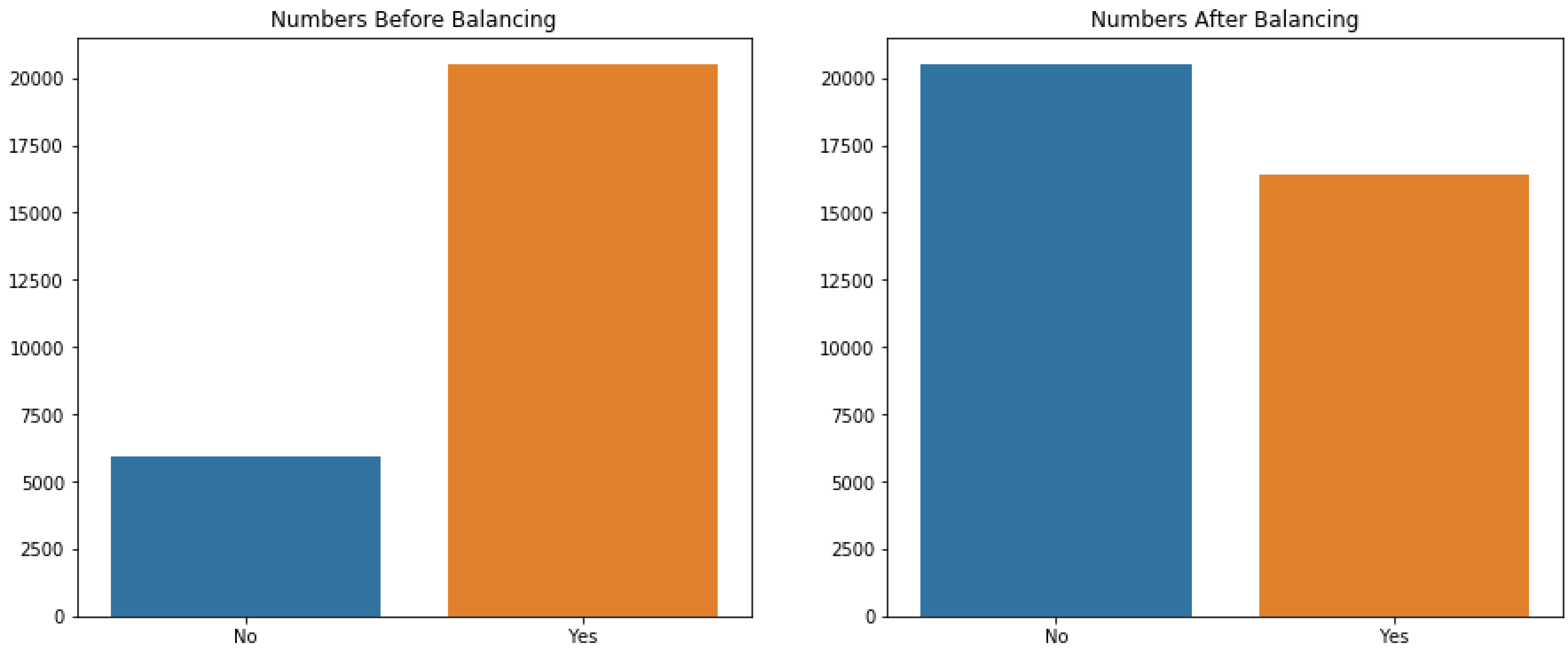
```
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
from collections import Counter
```

```
#The numbers before SMOTE
num_before = dict(Counter(y))

#Performing SMOTE

#Define pipeline
over = SMOTE(sampling_strategy=0.8)
under = RandomUnderSampler(sampling_strategy=0.8)
steps = [('o', over), ('u', under)]
pipeline = Pipeline(steps=steps)

#Transforming the dataset
X_smote, y_smote = pipeline.fit_resample(X,y)


#Tthe numbers after SMOTE
num_after = dict(Counter(y_smote))
```

```
print(num_before, num_after)
```

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

- As we have observed above that our data set was imbalanced. In an imbalanced dataset generally the class is not uniformly distributed.
- There are two sampling namely - Undersampling and Oversampling.
- Here we use **SMOTE** (Synthetic Minority Oversampling Technique) for Over sampling



Before balancing the minority and majority class had huge gap. We reduced the gap between the two as seen above

New Data frame after using SMOTE

```
[ ]  X1 = pd.DataFrame(X_smote)
     y1= pd.DataFrame(y_smote)
```

```
[ ]  new_data = pd.concat([X1, y1], axis=1)
     new_data.columns = ['LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE','PAY_0', 'PAY_2',
         'PAY_3', 'PAY_4','BILL_AMT1','BILL_AMT6','PAY_AMT1',
         'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6',"default payment next month"]
     new_data.head()
```

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	PAY_0	PAY_2	PAY_3	PAY_4	BILL_AMT1	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6	def paym r mo
0	170000	1	1	2	0	0	0	0	129848	0	5115	2500	0	0	0	0	
1	20000	1	2	1	0	0	0	0	16621	18893	1200	10000	1301	662	700	1000	
2	70000	2	2	3	0	0	0	0	70820	67818	2500	2507	2428	2594	2602	2500	
3	210000	2	2	1	0	0	0	0	128553	64108	3762	3808	4177	2594	2442	2319	
4	170000	2	2	2	0	0	0	0	71752	25140	2687	2671	3100	3286	1100	1000	

```
[ ] #FINDING THE NUMBER OF OUTLIERS
    default = new_data[new_data['default payment next month']==1]
    no_default = new_data[new_data['default payment next month']==0]
    outlier_fraction = len(default)/float(len(no_default))
```

```
[ ] print(outlier_fraction)
    print("Fraud Cases : {}".format(len(default)))
    print("Valid Cases : {}".format(len(no_default)))
```

```
0.8000292782901478
Fraud Cases : 16395
Valid Cases : 20493
```

Anomaly Detection using LOF and Isolation Forest

```
[ ]  # APPLYING BOTH MODELS
     classifiers = {
        "Isolation Forest":IsolationForest(n_estimators=100, max_samples=len(X),
                                        contamination=outlier_fraction,random_state= state, verbose=0),
        "Local Outlier Factor":LocalOutlierFactor(n_neighbors=20, algorithm='auto',
                                            leaf_size=30, metric='minkowski',
                                            p=2, metric_params=None)

     }
```

```
Isolation Forest: 18314
Accuracy Score :
0.5035241813055736
Classification Report :
              precision    recall  f1-score   support

           0       0.65      0.23      0.34     20493
           1       0.47      0.84      0.60     16395

    accuracy                           0.50     36888
   macro avg       0.56      0.54      0.47     36888
weighted avg       0.57      0.50      0.46     36888

Local Outlier Factor: 19063
Accuracy Score :
0.4832194751680763 4
Classification Report :
              precision    recall  f1-score   support

           0       0.52      0.75      0.62     20493
           1       0.32      0.14      0.20     16395

    accuracy                           0.48     36888
   macro avg       0.42      0.45      0.41     36888
weighted avg       0.43      0.48      0.43     36888
```

After applying Isolation forest and LOF we observe that:

Isolation Forest provides an accuracy of **0.503**
While LOF provides accuracy of **0.483**

Model Building

We use the below models and compare the results:

- Logistic Regression
- Random Forest
- KNN Algorithm
- SVM

❖ Before applying the model we split the data into train and test. We use 80:20 ratio for splitting.

```
[ ]  from sklearn.model_selection import train_test_split
     x_train, x_test, y_train, y_test = train_test_split(X_new,y_new,test_size=0.2,random_state=0)
```

Logistic Regression

Using Logistic Regression

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(random_state = 0)
lr.fit(x_train, y_train)
```

```
LogisticRegression(random_state=0)
```

```
print(cm)
print(acc)
print(pre)
print(recall)
print(f1)
```

```
[[3222  863]
 [1177 2116]]
0.7235023041474654
0.7103054716347768
0.642575159429092
0.6747448979591837
```

Using PCA

```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression(random_state = 0)
lr.fit(x_train, y_train)
```

```
LogisticRegression(random_state=0)
```

```
print(cm)
print(acc)
print(pre)
print(recall)
print(f1)
```

```
[[3162  923]
 [1453 1840]]
0.6779615071835186
0.6659428157799493
0.5587610081992105
0.6076618229854689
```

Random Forest

CODE

```
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 50)
classifier.fit(x_train, y_train)
```

```
RandomForestClassifier(n_estimators=50)
```

```
y_predRF= classifier.predict(x_test)
```

Random Forest

Using Logistic Regression

```
Confusion Matrix:
[[3552  533]
 [ 819 2474]]
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.87      0.84      4085
           1       0.82      0.75      0.79      3293

    accuracy                           0.82      7378
   macro avg       0.82      0.81      0.81      7378
weighted avg       0.82      0.82      0.82      7378

Accuracy: 0.8167525074545947
```

```
print(cm)
print(acc)
print(pre)
print(recall)
print(f1)
```

```
[[3552  533]
 [ 819 2474]]
0.8167525074545947
0.822746923844363l
0.7512906164591557
0.7853968253968253
```

Using PCA

```
Confusion Matrix:
[[3625  460]
 [ 826 2467]]
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.89      0.85      4085
           1       0.84      0.75      0.79      3293

    accuracy                           0.83      7378
   macro avg       0.83      0.82      0.82      7378
weighted avg       0.83      0.83      0.82      7378

Accuracy: 0.8256980211439414
```

```
print(cm)
print(acc)
print(pre)
print(recall)
print(f1)
```

```
[[3625  460]
 [ 826 2467]]
0.8256980211439414
0.8428425008541168
0.7491648952323ll
0.7932475884244373
```

KNN Algorithm

CODE

```
knnclassifier = KNeighborsClassifier(n_neighbors=5)
knnclassifier.fit(x_train,y_train)
```

```
KNeighborsClassifier()
```

```
y_predKNN= knnclassifier.predict(x_test)
```

KNN Algorithm

Using Logistic Regression

	precision	recall	f1-score	support
0	0.75	0.82	0.79	4085
1	0.75	0.67	0.71	3293
accuracy			0.75	7378
macro avg	0.75	0.74	0.75	7378
weighted avg	0.75	0.75	0.75	7378

```
]: 0.7527785307671455
```

```
print(cm)
print(acc)
print(pre)
print(recall)
print(f1)
```

```
[[3122  963]
 [ 879 2414]]
0.7503388452155055
0.7148356529464022
0.7330701488004859
0.7853968253968253
```

Using PCA

```
Confusion Matrix:
[[3081 1004]
 [ 899 2394]]
Classification Report:
```

	precision	recall	f1-score	support
0	0.77	0.75	0.76	4085
1	0.70	0.73	0.72	3293
accuracy			0.74	7378
macro avg	0.74	0.74	0.74	7378
weighted avg	0.74	0.74	0.74	7378

```
Accuracy:  0.74207102195717
```

```
print(cm)
print(acc)
print(pre)
print(recall)
print(f1)
```

```
[[3081 1004]
 [ 899 2394]]
0.74207102195717
0.7045320776927605
0.7269966595809293
0.7932475884244373
```

SVM (Support Vector Machine)

CODE

```
from sklearn import svm
#SVC (Support Vector Classifier) is to fit to the data you pr
clf = svm.SVC()
clf.fit(x_train, y_train)
```

```
SVC()
```

```
predictions = clf.predict(x_test)
print("Size of training set: ", x_test.shape)
print(predictions.shape)
```

```
Size of training set:  (7378, 18)
(7378,)
```

```
from sklearn.metrics import classification_report,confusion_m
print(confusion_matrix(y_test,predictions))
```

```
[[3404  681]
 [1293 2000]]
```

SVM (Support Vector Machine)

Using Logistic Regression

	precision	recall	f1-score	support
0	0.75	0.82	0.79	4085
1	0.75	0.67	0.71	3293
accuracy			0.75	7378
macro avg	0.75	0.74	0.75	7378
weighted avg	0.75	0.75	0.75	7378

: 0.7527785307671455

```
print(cm)
print(acc)
print(pre)
print(recall)
print(f1)
```

```
[[3350  735]
 [1089 2204]]
0.7527785307671455
0.7499149370534195
0.6692985119951412
0.7073170731707318
```

Using PCA

	precision	recall	f1-score	support
0	0.72	0.83	0.78	4085
1	0.75	0.61	0.67	3293
accuracy			0.73	7378
macro avg	0.74	0.72	0.72	7378
weighted avg	0.73	0.73	0.73	7378

0.7324478178368121

```
print(acc)
print(pre)
print(recall)
print(f1)
```

```
[[3404  681]
 [1293 2000]]
0.7324478178368121
0.7459903021260723
0.6073489219556635
0.6695681285570806
```

Comparison of all Models

	Logistic Regression		Random Forest		KNN Algorithm		SVM Model	
	Log Reg	PCA	Log Reg	PCA	Log Reg	PCA	Log Reg	PCA
Accuracy	0.724	0.678	0.817	0.826	0.750	0.742	0.753	0.732
Precision Score	0.710	0.666	0.823	0.843	0.715	0.705	0.750	0.746
Recall Score	0.643	0.559	0.751	0.749	0.733	0.727	0.669	0.607
F1 Score	0.675	0.608	0.785	0.793	0.785	0.793	0.707	0.670

Conclusion: From the above comparison table using Logistic Regression and PCA we find that **Random Forest** is the best model with high accuracy and F1 score that predicts the payment default status of the customers.

THANK YOU