ML Assignment Group: 119

Problem Statement

Predict whether the credit card using the customer is going to default or not.

- Import the data from the default of credit card clients (https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients) (2 points)
- Consider all columns as independent variables and assign to variable X except the last column and consider the last column as dependent variable and assign to variable y. Remove columns which don't help the problem statement. (1 point)
- Compute some basic statistical details like percentile, mean, standard deviation of dataset (1 point)
- Do Feature Scaling on Independent variables (2 points)
- Split the data into train and test dataset (1 point)
- Use sklearn library to train on train dataset on random forest and predict on test dataset (3 points)
- Compute the accuracy and confusion matrix. (2 points)

Contributors

J Manoj Balaji (2019AB04228@wilp.bits-pilani.ac.in)
Siddharth Verma (2019AD04097@wilp.bits-pilani.ac.in)
Randeep Singh (2019AD04069@wilp.bits-pilani.ac.in)

Import necessary dependencies ¶

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from collections import Counter
%matplotlib inline
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,confusion_matrix,classification_rep
ort
```

Import the dataset

Dropping ID variable since that's a unique variable for each row and doesn't help in prediction

In [2]:
```
df = pd.read_excel('default of credit card clients.xls',skiprows=1,usecols="B:
Y")
```

In [3]:
```
df.head()
```

Out[3]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	PAY_5	...
0	20000	2	2	1	24	2	2	-1	-1	-2	...
1	120000	2	2	2	26	-1	2	0	0	0	...
2	90000	2	2	2	34	0	0	0	0	0	...
3	50000	2	2	1	37	0	0	0	0	0	...
4	50000	1	2	1	57	-1	0	-1	0	0	...

5 rows × 24 columns

Understanding data

Computing some basic statistical details like percentile, mean, standard deviation of dataset

In [4]:
```
df.describe()
```

Out[4]:

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	3
mean	167484.322667	1.603733	1.853133	1.551867	35.485500	-0.016700	
std	129747.661567	0.489129	0.790349	0.521970	9.217904	1.123802	
min	10000.000000	1.000000	0.000000	0.000000	21.000000	-2.000000	
25%	50000.000000	1.000000	1.000000	1.000000	28.000000	-1.000000	
50%	140000.000000	2.000000	2.000000	2.000000	34.000000	0.000000	
75%	240000.000000	2.000000	2.000000	2.000000	41.000000	0.000000	
max	1000000.000000	2.000000	6.000000	3.000000	79.000000	8.000000	

8 rows × 24 columns

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 24 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   LIMIT_BAL                   30000 non-null  int64
 1   SEX                         30000 non-null  int64
 2   EDUCATION                   30000 non-null  int64
 3   MARRIAGE                    30000 non-null  int64
 4   AGE                         30000 non-null  int64
 5   PAY_0                       30000 non-null  int64
 6   PAY_2                       30000 non-null  int64
 7   PAY_3                       30000 non-null  int64
 8   PAY_4                       30000 non-null  int64
 9   PAY_5                       30000 non-null  int64
 10  PAY_6                       30000 non-null  int64
 11  BILL_AMT1                   30000 non-null  int64
 12  BILL_AMT2                   30000 non-null  int64
 13  BILL_AMT3                   30000 non-null  int64
 14  BILL_AMT4                   30000 non-null  int64
 15  BILL_AMT5                   30000 non-null  int64
 16  BILL_AMT6                   30000 non-null  int64
 17  PAY_AMT1                    30000 non-null  int64
 18  PAY_AMT2                    30000 non-null  int64
 19  PAY_AMT3                    30000 non-null  int64
 20  PAY_AMT4                    30000 non-null  int64
 21  PAY_AMT5                    30000 non-null  int64
 22  PAY_AMT6                    30000 non-null  int64
 23  default payment next month  30000 non-null  int64
dtypes: int64(24)
memory usage: 5.5 MB
```

From the dataset, we can see that SEX, EDUCATION, MARIAGE, AGE, PAY_0, PAY_2, PAY_3, PAY_4, PAY_5, PAY_6 are categorical variables and BILL_AMT1, BILL_AMT2, BILL_AMT3, BILL_AMT4, BILL_AMT5, BILL_AMT6, PAY_AMT1, PAY_AMT2, PAY_AMT3, PAY_AMT4, PAY_AMT5, PAY_AMT6, LIMIT_BAL are continuous variables. And as specified in the problem statement, the last column i.e. default payment next month is the target column. We need to scale the values of continuous variables.

From the description mentioned on the dataset repository page [default of credit card clients. (https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients)](https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients) We assign the following list of continous variables

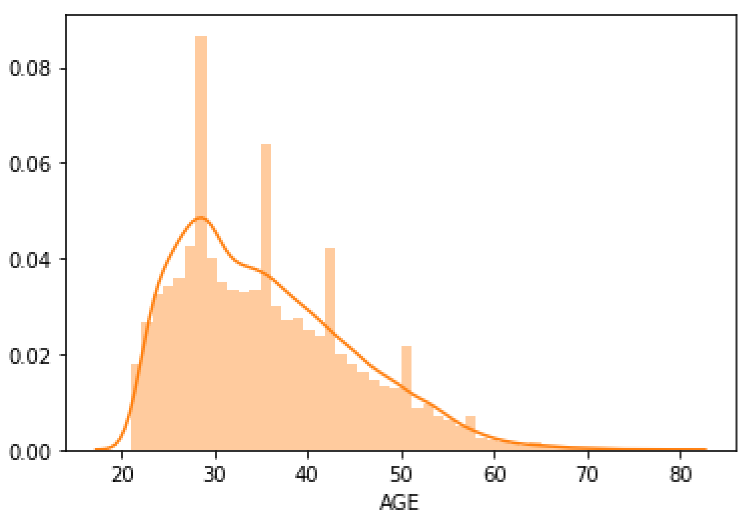
In [6]:
```
continous_list = ['BILL_AMT1','BILL_AMT2' ,'BILL_AMT3','BILL_AMT4','BILL_AMT5'
,'BILL_AMT6' ,'PAY_AMT1' ,'PAY_AMT2' ,'PAY_AMT3' ,'PAY_AMT4' ,'PAY_AMT5' ,'PAY
_AMT6','LIMIT_BAL']
```

In [7]:
```
categorical_list  = np.setdiff1d(df.columns,continous_list)
categorical_list
```

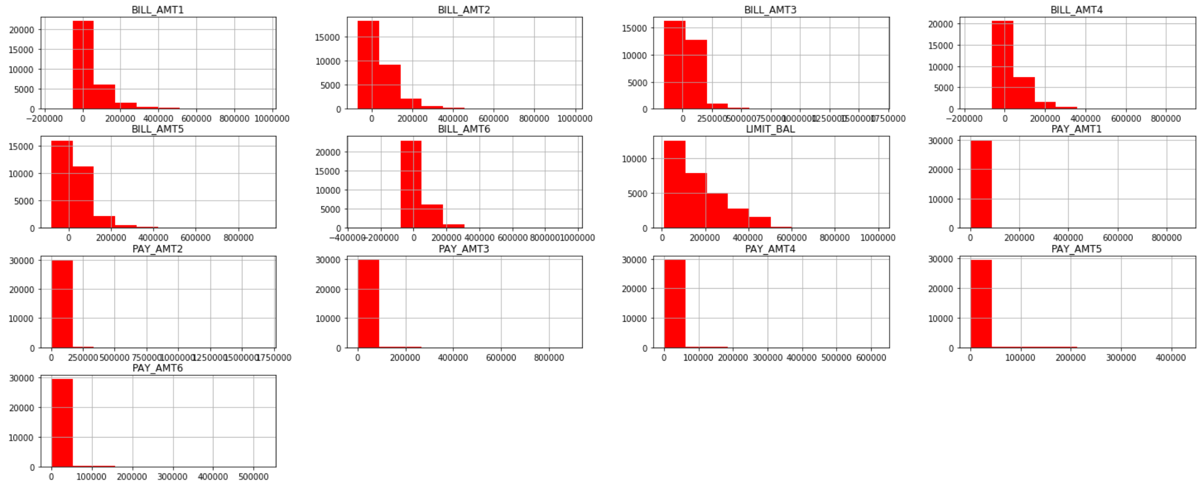
Out[7]:
```
array(['AGE', 'EDUCATION', 'MARRIAGE', 'PAY_0', 'PAY_2', 'PAY_3', 'PAY_4',
       'PAY_5', 'PAY_6', 'SEX', 'default payment next month'],
      dtype=object)
```

In [8]:
```
# understand age to decide whether to consider it as Continuous or Categorica
l, since the values are within a very short range, we can take it as categoric
al variable
sns.distplot(df.AGE, color='tab:orange')
print('Unique age values:', df.AGE.nunique())
```

```
Unique age values: 56
```

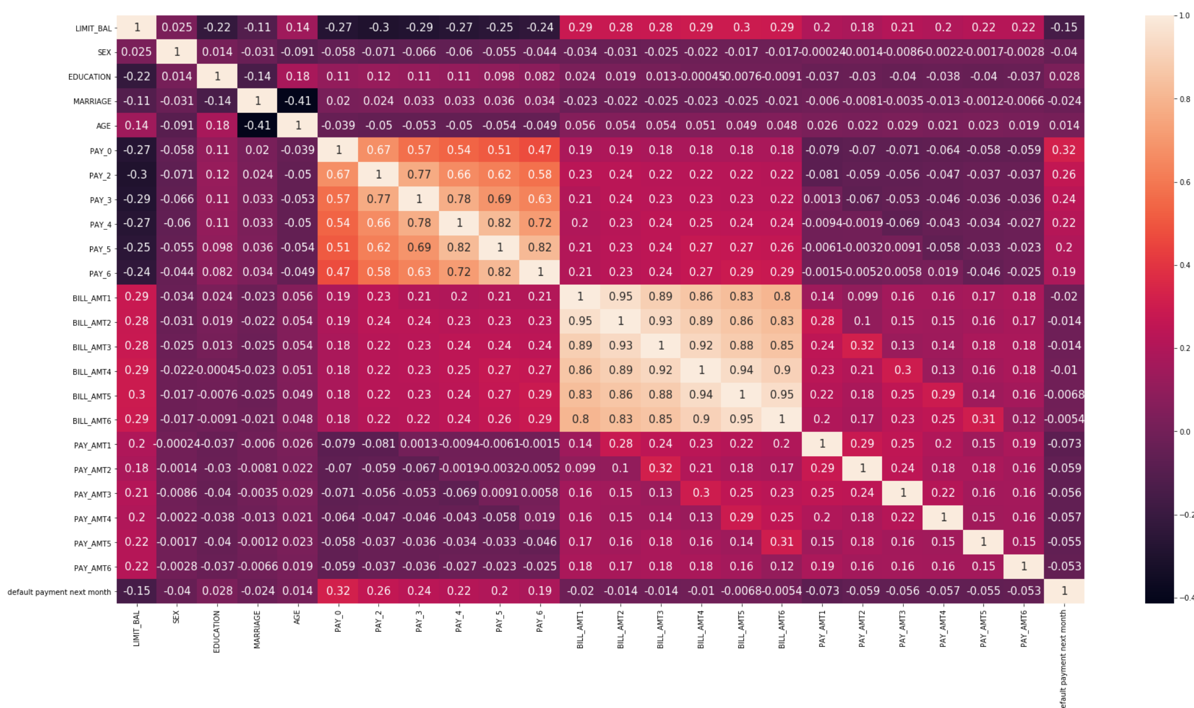


In [9]:
```python
df[continous_list].hist(bins=10, figsize=(25, 10), color='red');
```



In [10]:
```python
fig, ax = plt.subplots(figsize=(30,15))
sns.heatmap(df.corr(), annot=True, ax=ax, annot_kws={"size": 15})
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x1aad6033c18>



Deleting the below attributes because of high correlation amongsts themselves

All BILL_AMT variables have high positive correlation amongsts themselves
Pay 3 - 6 have high correlation amongsts themselves

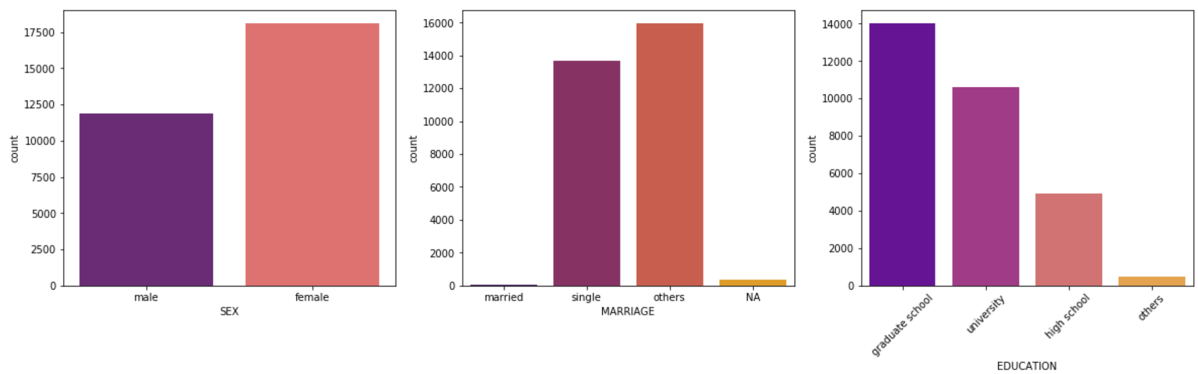
In [11]:
```
continous_list
```

Out[11]:
```
['BILL_AMT1',
 'BILL_AMT2',
 'BILL_AMT3',
 'BILL_AMT4',
 'BILL_AMT5',
 'BILL_AMT6',
 'PAY_AMT1',
 'PAY_AMT2',
 'PAY_AMT3',
 'PAY_AMT4',
 'PAY_AMT5',
 'PAY_AMT6',
 'LIMIT_BAL']
```

In [12]:
```
removable_continous = ['BILL_AMT3' ,'BILL_AMT2' ,'BILL_AMT1','BILL_AMT4','BILL
_AMT5' ,'BILL_AMT6']
removable_categorical = ['PAY_3', 'PAY_4', 'PAY_5', 'PAY_6']
df.drop(removable_continous, axis=1, inplace=True)
df.drop(removable_categorical, axis=1, inplace=True)
continous_list = np.asarray(continous_list)
categorical_list = np.asarray(categorical_list)
continous_list = np.setdiff1d(continous_list, removable_continous)
categorical_list = np.setdiff1d(categorical_list, removable_categorical)
print('continous list:',continous_list)
print('categorical list:',categorical_list)
```

```
continous list: ['LIMIT_BAL' 'PAY_AMT1' 'PAY_AMT2' 'PAY_AMT3' 'PAY_AMT4' 'PAY
_AMT5'
 'PAY_AMT6']
categorical list: ['AGE' 'EDUCATION' 'MARRIAGE' 'PAY_0' 'PAY_2' 'SEX'
 'default payment next month']
```

In [13]:
```
plt.subplots(figsize=(20,5))
plt.subplot(1,3,1)
ax = sns.countplot(df['SEX'], palette = 'magma')
ax.set_xticklabels(['male', 'female'])
plt.subplot(1,3,2)
ax = sns.countplot(df['MARRIAGE'], palette = 'inferno')
ax.set_xticklabels(['married','single','others', 'NA'])
plt.subplot(1,3,3)
ax = sns.countplot(df['EDUCATION'].map({1:'graduate school',2:'university',3:
'high school',4:'others',5:'others', 6:'others'}) , palette = 'plasma')
temp = ax.set_xticklabels(['graduate school','university','high school','other
s'], rotation=45)
```



Data Preprocessing

Feature Scaling on Independent variables

In [14]:
```
scaler = StandardScaler()
```

In [15]:
```
continous_transform_df = pd.DataFrame(scaler.fit_transform(df[continous_list
]),columns=continous_list)
```

In [16]:
```
continous_transform_df.head()
```

Out[16]:

	LIMIT_BAL	PAY_AMT1	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5	PAY_AMT6
0	-1.136720	-0.341942	-0.227086	-0.296801	-0.308063	-0.314136	-0.293382
1	-0.365981	-0.341942	-0.213588	-0.240005	-0.244230	-0.314136	-0.180878
2	-0.597202	-0.250292	-0.191887	-0.240005	-0.244230	-0.248683	-0.012122
3	-0.905498	-0.221191	-0.169361	-0.228645	-0.237846	-0.244166	-0.237130
4	-0.905498	-0.221191	1.335034	0.271165	0.266434	-0.269039	-0.255187

In [17]:
```
categorical_df = df[categorical_list]
```

In [18]:
```
categorical_df.head()
```

Out[18]:

	AGE	EDUCATION	MARRIAGE	PAY_0	PAY_2	SEX	default payment next month
0	24	2	1	2	2	2	1
1	26	2	2	-1	2	2	1
2	34	2	2	0	0	2	0
3	37	2	1	0	0	2	0
4	57	2	1	-1	0	1	0

In [19]:
```
# re-join scaled-continuous and categorical dataset
data_set = categorical_df.merge(continous_transform_df,left_index=True,right_i
ndex=True)
```

In [20]:
```
data_set.head()
```

Out[20]:

	AGE	EDUCATION	MARRIAGE	PAY_0	PAY_2	SEX	default payment next month	LIMIT_BAL	PAY_AMT1	PAY_
0	24	2	1	2	2	2	1	-1.136720	-0.341942	-0.22
1	26	2	2	-1	2	2	1	-0.365981	-0.341942	-0.2
2	34	2	2	0	0	2	0	-0.597202	-0.250292	-0.1
3	37	2	1	0	0	2	0	-0.905498	-0.221191	-0.1
4	57	2	1	-1	0	1	0	-0.905498	-0.221191	1.3

Split dataset to train and test datasets

In [21]:
```
X=data_set.drop('default payment next month',axis=1)
y=data_set['default payment next month']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
m_state=42)
```

In [22]:
```
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[22]: ((24000, 13), (6000, 13), (24000,), (6000,))

Use sklearn library to train random forest model on train dataset

In [23]:
```
rf = RandomForestClassifier(random_state=42)
```

In [24]:
```
rf.fit(X_train,y_train)
```

Out[24]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', 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=100,
                       n_jobs=None, oob_score=False, random_state=42, verbose
=0,
                       warm_start=False)
```

Evaluate the model on test dataset

In [25]:
```
y_predict=rf.predict(X_test)
```

Compute the accuracy and confusion matrix and classification report

In [26]:
```
accuracy = accuracy_score(y_pred = y_predict,y_true = y_test)
```

In [27]:
```
print("Accuracy: {}%".format(round(100*accuracy, 4)))
```

```
Accuracy: 81.4%
```

In [28]:
```
conf_mat = confusion_matrix(y_pred = y_predict,y_true = y_test)
```

In [29]:
```
conf_mat
```

Out[29]:
```
array([[4425,  262],
       [ 854,  459]], dtype=int64)
```

In [30]:
```
print('Classification Report: ')
print(classification_report(y_test, y_predict))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.94      0.89      4687
           1       0.64      0.35      0.45      1313

    accuracy                           0.81      6000
   macro avg       0.74      0.65      0.67      6000
weighted avg       0.79      0.81      0.79      6000
```

To check the imbalance of the target variable

In [31]:
```
Counter(y_test)
```

Out[31]:
```
Counter({0: 4687, 1: 1313})
```

Visualize the Confusion matrix as a heatmap

In [32]:
```
fig, ax = plt.subplots(figsize=(7,5))        # Sample figsize in inches
sns.heatmap(conf_mat, annot=True, ax=ax, cmap='Blues', annot_kws={"size": 25},
fmt="d")
```

Out[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1aad7fa1b00>
```