1.Getting the data files in dataframe

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
train = pd.read_csv('https://raw.githubusercontent.com/kuldeep27396/AVH/main/train-file.csv')
train.head()
```

	ID	Gender	Age	Region_Code	Occupation	Channel_Code	Vintage	Credit_Product
0	NNVBBKZB	Female	73	RG268	Other	X3	43	No
1	IDD62UNG	Female	30	RG277	Salaried	X1	32	No
2	HD3DSEMC	Female	56	RG268	Self_Employed	X3	26	No
3	BF3NC7KV	Male	34	RG270	Salaried	X1	19	No
4	TEASRWXV	Female	30	RG282	Salaried	X1	33	No

```
test = pd.read_csv('https://raw.githubusercontent.com/kuldeep27396/AVH/main/test-file.csv')
test.head()
```

	ID	Gender	Age	Region_Code	Occupation	Channel_Code	Vintage	Credit_Product	Av
0	VBENBARO	Male	29	RG254	Other	X1	25	Yes	
1	CCMEWNKY	Male	43	RG268	Other	X2	49	NaN	
2	VK3KGA9M	Male	31	RG270	Salaried	X1	14	No	
3	TT8RPZVC	Male	29	RG272	Other	X1	33	No	
4	SHQZEYTZ	Female	29	RG270	Other	X1	19	No	

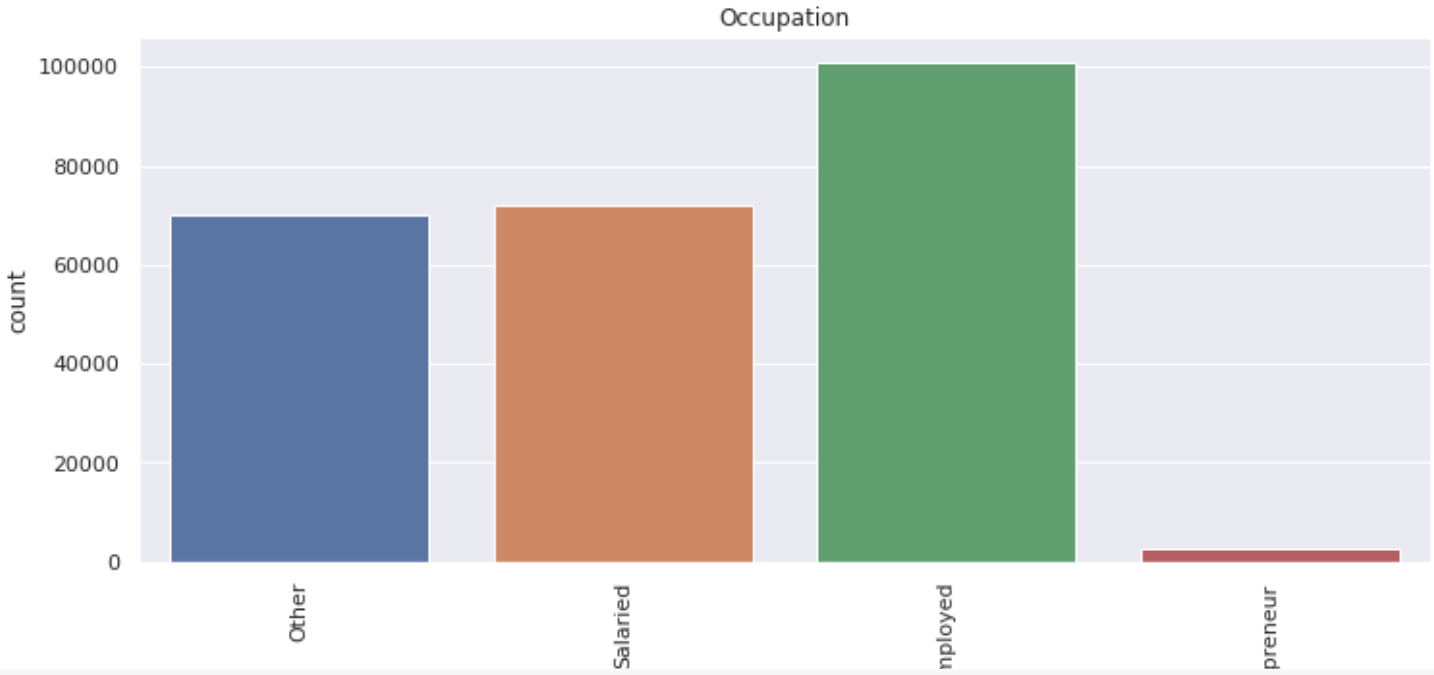
```
#copied train df to ff
ff= train.copy()
```

```
#copied test df to ff
gg = test.copy()
```

EDA

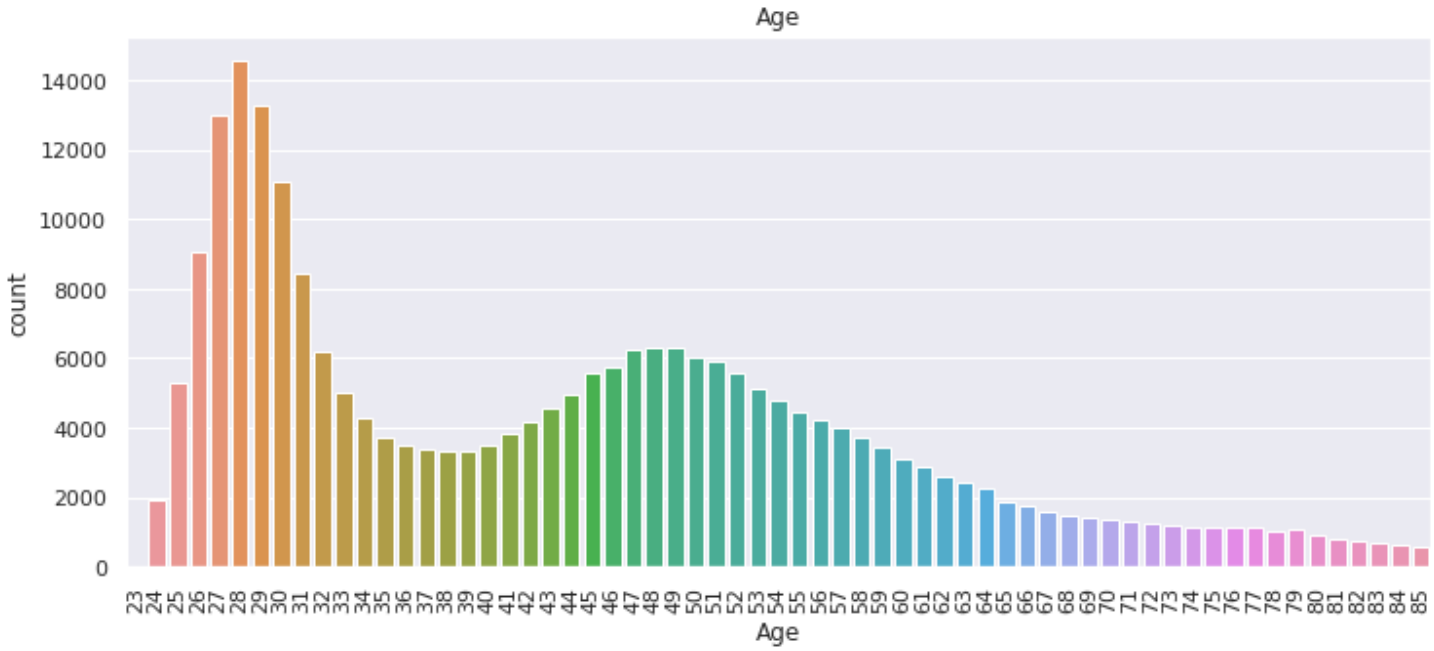
```
sns.countplot(train['Occupation']).tick_params(axis='x', rotation = 90)
plt.title('Occupation')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the follo
  FutureWarning
```



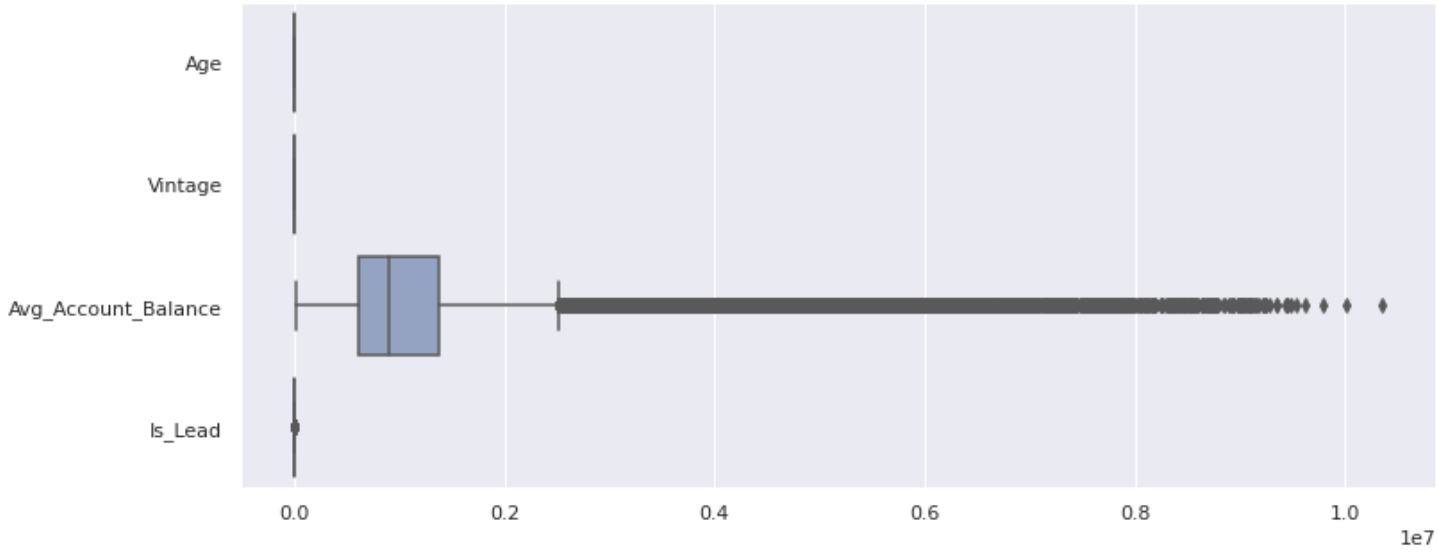

```
sns.set(rc={'figure.figsize':(12,5)})
sns.countplot(ff['Age']).tick_params(axis='x', rotation = 90)
plt.title('Age')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the follo
  FutureWarning
```




```
# To check the correlation among varibles
plt.figure(figsize=(10,5))
sns.heatmap(ff.corr())
plt.show()
```


```
ax = sns.boxplot(data=ff, orient="h", palette="Set2")
```



2.Train Data Preprocessing

- Converted **Credit_Product,Gender, Is_Active** to binary
- Encoded **Occupation, Channel_code** to numeric
- **Dropped ID from train as it has no corelation with our target**

```
import numpy as np
ff.replace(to_replace = np.nan, value =5, inplace =True) #replaced all Nan values to 5(selected
ff['Credit_Product']=np.where(ff['Credit_Product']=='No',0,ff['Credit_Product'])
ff['Credit_Product']=np.where(ff['Credit_Product']=='Yes',1,ff['Credit_Product'])

ff['Is_Active'] = ff['Is_Active'].map({'Yes': 1, 'No': 0})
ff['Gender'] = ff['Gender'].map({'Male': 1, 'Female': 0})
ff['Occupation'] = ff['Occupation'].map({'Other': 0, 'Salaried': 1, 'Self_Employed': 2, 'Entrep
ff['Channel_Code'] = ff['Channel_Code'].map({'X1': 0, 'X2': 1, 'X3': 2, 'X4': 3})
ff.drop(['ID'],axis =1, inplace = True)
```

Encode labels in column 'Region_Code'

```
# Import label encoder
from sklearn import preprocessing
```

```
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'Region_Code'.
ff['Region_Code']= label_encoder.fit_transform(ff['Region_Code'])

ff['Region_Code'].unique()

```

```
array([18, 27, 20, 32, 11, 15, 33,  4, 19,  7, 29, 30,  2, 34,  9, 31,  8,
       16, 10, 24,  6, 25, 23, 17, 22,  1, 12, 14, 28, 26, 13,  0,  5,  3,
       21])
```

```

```

Avg_Account_Balance have outliers as shown in the image, I have done label encoding and also removed outliers as shown below.

Encode labels in column 'Avg_Account_Balance'

```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'Avg_Account_Balance'.
ff['Avg_Account_Balance']= label_encoder.fit_transform(ff['Avg_Account_Balance'])

ff['Avg_Account_Balance'].unique()
```

```
array([ 69898,  28951,  96873, ...,   9819, 112817,  76068])
```

```
#import re
#replace = re.compile("([a-zA-Z]+)")
#ff['Region_Code'] = ff['Region_Code'].astype(str).replace(replace, "")
```

```
#get the dataframe after transformation
ff.head()
```

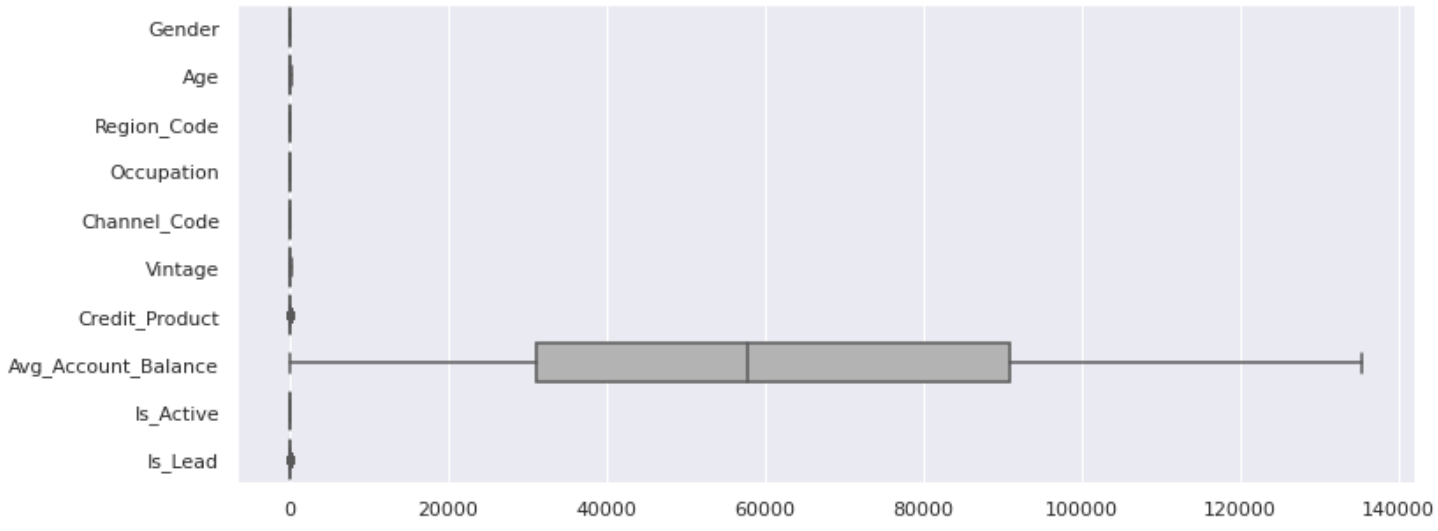
	Gender	Age	Region_Code	Occupation	Channel_Code	Vintage	Credit_Product	Avg_Account_Ba
0	0	73	18	0	2	43	0	
1	0	30	27	1	0	32	0	
2	0	56	18	2	2	26	0	
3	1	34	20	1	0	19	0	
4	0	30	32	1	0	33	0	

Removing outliers from Avg_Account_Balance

```
# Importing
import sklearn
import pandas as pd

# IQR
Q1 = np.percentile(ff['Avg_Account_Balance'], 25,
                   interpolation = 'midpoint')

Q3 = np.percentile(ff['Avg_Account_Balance'], 75,
                   interpolation = 'midpoint')
IQR = Q3 - Q1

print("Old Shape: ", ff.shape)

# Upper bound
upper = np.where(ff['Avg_Account_Balance'] >= (Q3+1.5*IQR))
# Lower bound
lower = np.where(ff['Avg_Account_Balance'] <= (Q1-1.5*IQR))

''' Removing the Outliers '''
ff.drop(upper[0], inplace = True)
ff.drop(lower[0], inplace = True)

print("New Shape: ", ff.shape)

```

```
Old Shape:  (245725, 10)
New Shape:  (245725, 10)
```

```
ax = sns.boxplot(data=ff, orient="h", palette="Set2")
```




```
ff['Credit_Product'].value_counts()
```

```
0    144357
1     72043
5     29325
Name: Credit_Product, dtype: int64
```

3.Test data processing

Everything will be same as train preprocessing

```
gg.head()
```

	ID	Gender	Age	Region_Code	Occupation	Channel_Code	Vintage	Credit_Product	Av
0	VBENBARO	Male	29	RG254	Other	X1	25	Yes	
1	CCMEWNKY	Male	43	RG268	Other	X2	49	NaN	
2	VK3KGA9M	Male	31	RG270	Salaried	X1	14	No	
3	TT8RPZVC	Male	29	RG272	Other	X1	33	No	
4	SHQZEYTZ	Female	29	RG270	Other	X1	19	No	

```
gg.replace(to_replace = np.nan, value =5, inplace =True)
gg['Credit_Product']=np.where(gg['Credit_Product']=='No',0,gg['Credit_Product'])
gg['Credit_Product']=np.where(gg['Credit_Product']=='Yes',1,gg['Credit_Product'])

gg['Channel_Code'] = gg['Channel_Code'].map({'X1': 0, 'X2': 1, 'X3': 2, 'X4': 3})

gg['Gender'] = gg['Gender'].map({'Male': 1, 'Female': 0})

gg['Occupation'] = gg['Occupation'].map({'Other': 0, 'Salaried': 1, 'Self_Employed': 2, 'Entrep
gg['Is_Active'] = gg['Is_Active'].map({'Yes': 1, 'No': 0})
```

```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'Avg_Account_Balance'.
gg['Avg_Account_Balance']= label_encoder.fit_transform(gg['Avg_Account_Balance'])

gg['Avg_Account_Balance'].unique()
```

```
array([27635, 37999,   732, ..., 48831, 63962, 18118])
```

```
# Import label encoder
from sklearn import preprocessing

# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'Region_Code'.
gg['Region_Code']= label_encoder.fit_transform(gg['Region_Code'])

gg['Region_Code'].unique()
```

```
array([ 4, 18, 20, 22,  3,  7, 34, 33, 27, 23, 11, 30, 24, 28, 19, 31,  1,
        2, 29, 32, 12, 13, 14,  6, 15, 10,  8, 21,  0, 25, 26, 17,  5,  9,
       16])
```

```
gg['Is_Active'].value_counts()
```

```
0    63797
1    41515
Name: Is_Active, dtype: int64
```

4.Test-Train Split

```
# Import the required library
from sklearn.model_selection import train_test_split
```

```
X = ff.drop(['Is_Lead'], 1)
X.head()
```

	Gender	Age	Region_Code	Occupation	Channel_Code	Vintage	Credit_Product	Avg_Account_Ba
0	0	73	18	0	2	43	0	6
1	0	30	27	1	0	32	0	2
2	0	56	18	2	2	26	0	9
3	1	34	20	1	0	19	0	1
4	0	30	32	1	0	33	0	5

```
# Putting the target variable in y
y = ff['Is_Lead']
y.head()
```

```
0    0
1    0
2    0
3    0
4    0
Name: Is_Lead, dtype: int64
```

```
# Split the dataset into 70% and 30% for train and test respectively(taken with imp features)
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=0.3, random
```

```
X_train['Is_Active'].value_counts()
```

```
0    105293
1     66714
Name: Is_Active, dtype: int64
```

5.Model Building : Logistic Regression

```
#Let's start with importing necessary libraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model  import Ridge,Lasso,RidgeCV, LassoCV, ElasticNet, ElasticNetCV, Logis
from sklearn.model_selection import train_test_split
```

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas
  import pandas.util.testing as tm
```

```
log_reg = LogisticRegression()
log_reg.fit(X_train,y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Feature selection with RFE

```
# Running RFE with 6 variables as output
from sklearn.feature_selection import RFE
rfe = RFE(log_reg,6)
rfe = rfe.fit(X_train, y_train)
```

```
rfe.support_
```

```
array([ True, False, False,  True,  True,  True,  True, False,  True])
```

```
# Features that have been selected by RFE
list(zip(X_train.columns, rfe.support_, rfe.ranking_))
```

```
[('Gender', True, 1),
 ('Age', False, 2),
 ('Region_Code', False, 3),
 ('Occupation', True, 1),
 ('Channel_Code', True, 1),
 ('Vintage', True, 1),
 ('Credit_Product', True, 1),
 ('Avg_Account_Balance', False, 4),
 ('Is_Active', True, 1)]
```

```
# Put all the columns selected by RFE in the variable 'col'
col = X_train.columns[rfe.support_]
```

```
col
```

```
Index(['Gender', 'Occupation', 'Channel_Code', 'Vintage', 'Credit_Product',
       'Is_Active'],
      dtype='object')
```

```
# Selecting columns selected by RFE
X_train1 = X_train[col]
```

```
X_train1
```

	Gender	Occupation	Channel_Code	Vintage	Credit_Product	Is_Active
40774	1	1	0	27	0	0
158466	1	2	2	45	1	0
82878	0	2	0	50	0	0
117102	1	1	0	14	0	0
28563	1	2	1	56	0	1
...	...	...	...	...	...	...
93553	1	2	1	32	1	0
181568	1	2	1	80	1	0
94735	1	0	0	44	0	0
236669	0	2	2	103	1	0
83209	0	0	2	81	0	1

172007 rows × 6 columns

```
log_reg = LogisticRegression()

log_reg.fit(X_train1,y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
X_test1 = X_test[col]
```

```
y_pred = log_reg.predict(X_test1)
```

```
# Area Under Curve
auc = roc_auc_score(y_test, y_pred)
auc
```

```
0.6994539379268162
```

6.Model Building: Catboost Classifier

```
!pip install CatBoost
from catboost import CatBoostClassifier
```

```
Collecting CatBoost
  Downloading https://files.pythonhosted.org/packages/47/80/8e9c57ec32dfed6ba2922bc5c96462cbf8
     |████████████████████████████████| 67.3MB 61kB/s
Requirement already satisfied: pandas>=0.24.0 in /usr/local/lib/python3.7/dist-packages (from C
Requirement already satisfied: graphviz in /usr/local/lib/python3.7/dist-packages (from CatBoos
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from CatBo
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.7/dist-packages (from Ca
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from CatBoost) (1
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (from CatBoost
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from CatBoost)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-packages (from par
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (fro
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from mat
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/pytho
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.7/dist-packages (from
Installing collected packages: CatBoost
Successfully installed CatBoost-0.25.1
```

```
#with all features
model2 = CatBoostClassifier(iterations=100)

# fit the model with the training data
model2.fit(X_train,y_train,plot=False)
print('\n Model Trainied')

# predict the target on the train dataset
predict_train = model2.predict(X_train)

```

```
42:     learn: 0.3390712        total: 1.47s    remaining: 1.94s
43:     learn: 0.3389433        total: 1.5s     remaining: 1.91s
44:     learn: 0.3388630        total: 1.53s    remaining: 1.87s
45:     learn: 0.3388316        total: 1.56s    remaining: 1.83s
46:     learn: 0.3387661        total: 1.59s    remaining: 1.79s
47:     learn: 0.3386568        total: 1.62s    remaining: 1.76s
48:     learn: 0.3385286        total: 1.65s    remaining: 1.72s
49:     learn: 0.3384583        total: 1.7s     remaining: 1.7s
50:     learn: 0.3383273        total: 1.73s    remaining: 1.66s
51:     learn: 0.3382499        total: 1.77s    remaining: 1.63s
52:     learn: 0.3381685        total: 1.8s     remaining: 1.59s
53:     learn: 0.3381484        total: 1.83s    remaining: 1.56s
54:     learn: 0.3380283        total: 1.86s    remaining: 1.52s
55:     learn: 0.3379584        total: 1.89s    remaining: 1.49s
56:     learn: 0.3378348        total: 1.93s    remaining: 1.45s
57:     learn: 0.3377565        total: 1.96s    remaining: 1.42s
58:     learn: 0.3376638        total: 1.99s    remaining: 1.39s
59:     learn: 0.3375980        total: 2.03s    remaining: 1.35s
60:     learn: 0.3375130        total: 2.06s    remaining: 1.31s
61:     learn: 0.3374044        total: 2.09s    remaining: 1.28s
62:     learn: 0.3373924        total: 2.12s    remaining: 1.24s
63:     learn: 0.3373058        total: 2.15s    remaining: 1.21s
64:     learn: 0.3372280        total: 2.18s    remaining: 1.17s
65:     learn: 0.3371683        total: 2.21s    remaining: 1.14s
66:     learn: 0.3370851        total: 2.24s    remaining: 1.1s
67:     learn: 0.3369372        total: 2.28s    remaining: 1.07s
68:     learn: 0.3368808        total: 2.32s    remaining: 1.04s
69:     learn: 0.3367826        total: 2.35s    remaining: 1.01s
70:     learn: 0.3366921        total: 2.38s    remaining: 974ms
71:     learn: 0.3365919        total: 2.43s    remaining: 944ms
72:     learn: 0.3364775        total: 2.47s    remaining: 912ms
73:     learn: 0.3363537        total: 2.5s     remaining: 878ms
74:     learn: 0.3362945        total: 2.53s    remaining: 843ms
75:     learn: 0.3361972        total: 2.56s    remaining: 809ms
76:     learn: 0.3361174        total: 2.59s    remaining: 774ms
77:     learn: 0.3360278        total: 2.62s    remaining: 740ms
78:     learn: 0.3359646        total: 2.66s    remaining: 706ms
79:     learn: 0.3359204        total: 2.69s    remaining: 672ms
```

```
80:     learn: 0.3358490        total: 2.72s    remaining: 638ms
81:     learn: 0.3357755        total: 2.75s    remaining: 604ms
82:     learn: 0.3357096        total: 2.78s    remaining: 569ms
83:     learn: 0.3356039        total: 2.81s    remaining: 535ms
84:     learn: 0.3355404        total: 2.84s    remaining: 501ms
85:     learn: 0.3354675        total: 2.88s    remaining: 469ms
86:     learn: 0.3354332        total: 2.91s    remaining: 435ms
87:     learn: 0.3353857        total: 2.94s    remaining: 401ms
88:     learn: 0.3353028        total: 2.97s    remaining: 367ms
89:     learn: 0.3352315        total: 3s       remaining: 334ms
90:     learn: 0.3351515        total: 3.03s    remaining: 300ms
91:     learn: 0.3350505        total: 3.06s    remaining: 267ms
92:     learn: 0.3350037        total: 3.1s     remaining: 233ms
93:     learn: 0.3349048        total: 3.13s    remaining: 200ms
94:     learn: 0.3348624        total: 3.16s    remaining: 166ms
95:     learn: 0.3347966        total: 3.19s    remaining: 133ms
96:     learn: 0.3347316        total: 3.22s    remaining: 99.7ms
97:     learn: 0.3346657        total: 3.25s    remaining: 66.4ms
98:     learn: 0.3346313        total: 3.28s    remaining: 33.2ms
99:     learn: 0.3345538        total: 3.32s    remaining: 0us

 Model Trainied
```

```
1  # Running RFE with 4 variables as output
2  from sklearn.feature_selection import RFE
3  rfe = RFE(model2,7)
4  rfe = rfe.fit(X_train, y_train)
```

```
40:     learn: 0.3402027        total: 1.29s    remaining: 1.86s
41:     learn: 0.3401493        total: 1.32s    remaining: 1.82s
42:     learn: 0.3399346        total: 1.35s    remaining: 1.79s
43:     learn: 0.3398523        total: 1.38s    remaining: 1.76s
44:     learn: 0.3397789        total: 1.41s    remaining: 1.73s
45:     learn: 0.3397065        total: 1.44s    remaining: 1.69s
46:     learn: 0.3395877        total: 1.48s    remaining: 1.66s
47:     learn: 0.3395205        total: 1.51s    remaining: 1.64s
48:     learn: 0.3394008        total: 1.54s    remaining: 1.6s
49:     learn: 0.3393227        total: 1.57s    remaining: 1.57s
50:     learn: 0.3392520        total: 1.6s     remaining: 1.54s
51:     learn: 0.3392015        total: 1.63s    remaining: 1.5s
52:     learn: 0.3391566        total: 1.66s    remaining: 1.47s
53:     learn: 0.3391031        total: 1.69s    remaining: 1.44s
54:     learn: 0.3389804        total: 1.72s    remaining: 1.41s
55:     learn: 0.3388843        total: 1.75s    remaining: 1.38s
56:     learn: 0.3386989        total: 1.78s    remaining: 1.35s
57:     learn: 0.3385967        total: 1.82s    remaining: 1.31s
58:     learn: 0.3385083        total: 1.85s    remaining: 1.28s
59:     learn: 0.3384307        total: 1.89s    remaining: 1.26s
60:     learn: 0.3383233        total: 1.92s    remaining: 1.23s
61:     learn: 0.3382443        total: 1.95s    remaining: 1.2s
62:     learn: 0.3381674        total: 1.98s    remaining: 1.16s
63:     learn: 0.3381157        total: 2.01s    remaining: 1.13s
64:     learn: 0.3380388        total: 2.04s    remaining: 1.1s
65:     learn: 0.3379332        total: 2.07s    remaining: 1.07s
66:     learn: 0.3378293        total: 2.1s     remaining: 1.04s
67:     learn: 0.3377673        total: 2.14s    remaining: 1.01s
68:     learn: 0.3377045        total: 2.18s    remaining: 978ms
69:     learn: 0.3376574        total: 2.22s    remaining: 953ms
70:     learn: 0.3375585        total: 2.25s    remaining: 921ms
71:     learn: 0.3374805        total: 2.28s    remaining: 888ms
72:     learn: 0.3374344        total: 2.31s    remaining: 856ms
73:     learn: 0.3374093        total: 2.34s    remaining: 823ms
74:     learn: 0.3373401        total: 2.37s    remaining: 791ms
75:     learn: 0.3372891        total: 2.41s    remaining: 760ms
76:     learn: 0.3372191        total: 2.44s    remaining: 728ms
77:     learn: 0.3371459        total: 2.47s    remaining: 697ms
```

```
78:     learn: 0.3371133        total: 2.5s     remaining: 665ms
79:     learn: 0.3370630        total: 2.53s    remaining: 633ms
80:     learn: 0.3370060        total: 2.56s    remaining: 600ms
81:     learn: 0.3369070        total: 2.59s    remaining: 569ms
82:     learn: 0.3368732        total: 2.62s    remaining: 537ms
83:     learn: 0.3368452        total: 2.65s    remaining: 505ms
84:     learn: 0.3367455        total: 2.69s    remaining: 474ms
85:     learn: 0.3367153        total: 2.72s    remaining: 442ms
86:     learn: 0.3366887        total: 2.75s    remaining: 410ms
87:     learn: 0.3366336        total: 2.77s    remaining: 378ms
88:     learn: 0.3365712        total: 2.8s     remaining: 347ms
89:     learn: 0.3365343        total: 2.84s    remaining: 316ms
90:     learn: 0.3364687        total: 2.87s    remaining: 284ms
91:     learn: 0.3364084        total: 2.9s     remaining: 252ms
92:     learn: 0.3363835        total: 2.93s    remaining: 220ms
93:     learn: 0.3363709        total: 2.96s    remaining: 189ms
94:     learn: 0.3363109        total: 2.99s    remaining: 157ms
95:     learn: 0.3362272        total: 3.02s    remaining: 126ms
96:     learn: 0.3361722        total: 3.05s    remaining: 94.4ms
97:     learn: 0.3361373        total: 3.08s    remaining: 62.9ms
98:     learn: 0.3360994        total: 3.11s    remaining: 31.4ms
99:     learn: 0.3360231        total: 3.15s    remaining: 0us
```

```
1  rfe.support_
```

```
array([False,  True, False,  True,  True,  True,  True,  True,  True])
```

```
1  # Features that have been selected by RFE
2  list(zip(X_train.columns, rfe.support_, rfe.ranking_))
```

```
[('Gender', False, 3),
 ('Age', True, 1),
 ('Region_Code', False, 2),
 ('Occupation', True, 1),
 ('Channel_Code', True, 1),
 ('Vintage', True, 1),
 ('Credit_Product', True, 1),
 ('Avg_Account_Balance', True, 1),
 ('Is_Active', True, 1)]
```

```
1  # Put all the columns selected by RFE in the variable 'col'
2  col = X_train.columns[rfe.support_]
```

```
1  X_train11 = X_train[col]
```

```
1  #with all features
2  model2 = CatBoostClassifier(iterations=100)
3
4  # fit the model with the training data
5  model2.fit(X_train11,y_train,plot=False)
6  print('\n Model Trainied')
7
8  # predict the target on the train dataset
9  predict_train = model2.predict(X_train11)
```

```
42:     learn: 0.3399346        total: 1.36s    remaining: 1.8s
43:     learn: 0.3398523        total: 1.39s    remaining: 1.77s
44:     learn: 0.3397789        total: 1.42s    remaining: 1.74s
45:     learn: 0.3397065        total: 1.46s    remaining: 1.71s
46:     learn: 0.3395877        total: 1.49s    remaining: 1.68s
47:     learn: 0.3395205        total: 1.52s    remaining: 1.65s
```

```
48:     learn: 0.3394008        total: 1.55s    remaining: 1.62s
49:     learn: 0.3393227        total: 1.58s    remaining: 1.58s
50:     learn: 0.3392520        total: 1.61s    remaining: 1.55s
51:     learn: 0.3392015        total: 1.64s    remaining: 1.51s
52:     learn: 0.3391566        total: 1.67s    remaining: 1.48s
53:     learn: 0.3391031        total: 1.7s     remaining: 1.45s
54:     learn: 0.3389804        total: 1.75s    remaining: 1.43s
55:     learn: 0.3388843        total: 1.78s    remaining: 1.4s
56:     learn: 0.3386989        total: 1.82s    remaining: 1.37s
57:     learn: 0.3385967        total: 1.85s    remaining: 1.34s
58:     learn: 0.3385083        total: 1.88s    remaining: 1.31s
59:     learn: 0.3384307        total: 1.91s    remaining: 1.27s
60:     learn: 0.3383233        total: 1.94s    remaining: 1.24s
61:     learn: 0.3382443        total: 1.98s    remaining: 1.21s
62:     learn: 0.3381674        total: 2.01s    remaining: 1.18s
63:     learn: 0.3381157        total: 2.04s    remaining: 1.15s
64:     learn: 0.3380388        total: 2.08s    remaining: 1.12s
65:     learn: 0.3379332        total: 2.1s     remaining: 1.08s
66:     learn: 0.3378293        total: 2.14s    remaining: 1.05s
67:     learn: 0.3377673        total: 2.17s    remaining: 1.02s
68:     learn: 0.3377045        total: 2.2s     remaining: 990ms
69:     learn: 0.3376574        total: 2.23s    remaining: 957ms
70:     learn: 0.3375585        total: 2.26s    remaining: 924ms
71:     learn: 0.3374805        total: 2.29s    remaining: 891ms
72:     learn: 0.3374344        total: 2.32s    remaining: 859ms
73:     learn: 0.3374093        total: 2.35s    remaining: 826ms
74:     learn: 0.3373401        total: 2.38s    remaining: 794ms
75:     learn: 0.3372891        total: 2.42s    remaining: 763ms
76:     learn: 0.3372191        total: 2.45s    remaining: 731ms
77:     learn: 0.3371459        total: 2.48s    remaining: 699ms
78:     learn: 0.3371133        total: 2.5s     remaining: 666ms
79:     learn: 0.3370630        total: 2.54s    remaining: 634ms
80:     learn: 0.3370060        total: 2.56s    remaining: 602ms
81:     learn: 0.3369070        total: 2.6s     remaining: 570ms
82:     learn: 0.3368732        total: 2.63s    remaining: 539ms
83:     learn: 0.3368452        total: 2.66s    remaining: 507ms
84:     learn: 0.3367455        total: 2.7s     remaining: 476ms
85:     learn: 0.3367153        total: 2.73s    remaining: 444ms
86:     learn: 0.3366887        total: 2.76s    remaining: 413ms
87:     learn: 0.3366336        total: 2.79s    remaining: 381ms
88:     learn: 0.3365712        total: 2.82s    remaining: 349ms
89:     learn: 0.3365343        total: 2.86s    remaining: 318ms
90:     learn: 0.3364687        total: 2.89s    remaining: 286ms
91:     learn: 0.3364084        total: 2.92s    remaining: 254ms
92:     learn: 0.3363835        total: 2.96s    remaining: 223ms
93:     learn: 0.3363709        total: 2.99s    remaining: 191ms
94:     learn: 0.3363109        total: 3.02s    remaining: 159ms
95:     learn: 0.3362272        total: 3.05s    remaining: 127ms
96:     learn: 0.3361722        total: 3.09s    remaining: 95.5ms
97:     learn: 0.3361373        total: 3.12s    remaining: 63.6ms
98:     learn: 0.3360994        total: 3.14s    remaining: 31.8ms
99:     learn: 0.3360231        total: 3.17s    remaining: 0us

 Model Trainied
```

```
1  X_test11 = X_test[col]
```

```
1  y_pred11 = model2.predict(X_test11)
2  auc = roc_auc_score(y_test, y_pred11)
3  auc
```

```
0.7512887445265599
```

Improved AUC from 69 to 75

7.Model Building: ANN Classification

```
#converting to scaler
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
from keras.models import Sequential
classifier = Sequential()
```

```
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
```

```
#Initializing Neural Network
from keras.models import Sequential
from keras.layers import Activation, Dense, Dropout
from keras import optimizers
Model = Sequential()
Model.add(Dense(64, kernel_initializer = 'uniform', activation = 'relu', input_dim = 9))
# Adding the second hidden layer
Model.add(Dense(48, kernel_initializer = 'uniform', activation = 'relu'))
Model.add(Dense(48, kernel_initializer = 'uniform', activation = 'relu'))
Model.add(Dense(48, kernel_initializer = 'uniform', activation = 'relu'))
Model.add(Dense(32, kernel_initializer = 'uniform', activation = 'tanh'))
Model.add(Dense(32, kernel_initializer = 'uniform', activation = 'relu'))
Model.add(Dropout(0.2))
# Adding the output layer
Model.add(Dense(1, kernel_initializer = 'uniform', activation = 'sigmoid'))
# Compiling Neural Network
Model.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['AUC'],)
#fitting the neural Network
Model.fit(X_train, y_train, batch_size =48, epochs = 30)
```

```
Epoch 2/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3631 - auc: 0.8579
Epoch 3/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3553 - auc: 0.8625
Epoch 4/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3498 - auc: 0.8676
Epoch 5/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3471 - auc: 0.8681
Epoch 6/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3500 - auc: 0.8654
Epoch 7/30

3584/3584 [==============================] - 7s 2ms/step - loss: 0.3508 - auc: 0.8671
Epoch 8/30
```

```
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3476 - auc: 0.8693
Epoch 9/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3484 - auc: 0.8686
Epoch 10/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3467 - auc: 0.8706
Epoch 11/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3441 - auc: 0.8713
Epoch 12/30
3584/3584 [==============================] - 8s 2ms/step - loss: 0.3445 - auc: 0.8727
Epoch 13/30
3584/3584 [==============================] - 8s 2ms/step - loss: 0.3456 - auc: 0.8707
Epoch 14/30
3584/3584 [==============================] - 8s 2ms/step - loss: 0.3450 - auc: 0.8719
Epoch 15/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3436 - auc: 0.8715
Epoch 16/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3460 - auc: 0.8720
Epoch 17/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3468 - auc: 0.8706
Epoch 18/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3437 - auc: 0.8728
Epoch 19/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3466 - auc: 0.8717
Epoch 20/30
3584/3584 [==============================] - 8s 2ms/step - loss: 0.3434 - auc: 0.8730
Epoch 21/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3465 - auc: 0.8692
Epoch 22/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3445 - auc: 0.8726
Epoch 23/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3439 - auc: 0.8728
Epoch 24/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3455 - auc: 0.8717
Epoch 25/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3459 - auc: 0.8713
Epoch 26/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3402 - auc: 0.8763
Epoch 27/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3461 - auc: 0.8711
Epoch 28/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3439 - auc: 0.8729
Epoch 29/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3432 - auc: 0.8737
Epoch 30/30
3584/3584 [==============================] - 7s 2ms/step - loss: 0.3442 - auc: 0.8729
<keras.callbacks.History at 0x7fc24d8f0ad0>
```

Getting AUC

```
Y_prediction=Model.predict(X_test)
# Area Under Curve
auc = roc_auc_score(y_test, Y_prediction)
auc
```

```
0.8713652724144249
```

Got final AUC:0.8721

1. Got the testing data ID in ID column

2. Dropped ID from testing data to do after preprocessing i.e. converting to Scaler

```
ID = gg['ID']
gg.drop(['ID'], axis =1, inplace =True)
```

Converting to Scaler the testing data

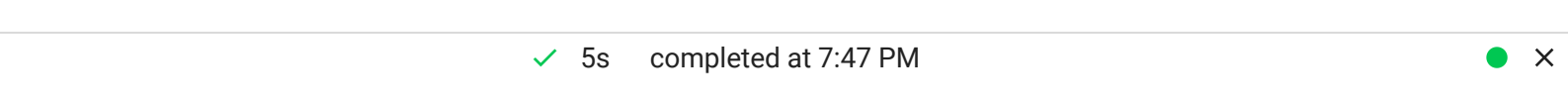
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train55 = sc.fit_transform(gg)
```

```
X_train55
```

```
array([[ 0.90829868, -0.99987782, -1.45176439, ...,  0.06952535,
        -0.45614794, -0.80668205],
       [ 0.90829868, -0.05856336, -0.16779796, ...,  2.61183556,
         0.0167183 , -0.80668205],
       [ 0.90829868, -0.86540432,  0.01562582, ..., -0.56605221,
        -1.68362001, -0.80668205],
       ...,
       [ 0.90829868, -0.59645734, -1.45176439, ..., -0.56605221,
         1.20130207, -0.80668205],
       [ 0.90829868,  0.6138041 , -1.45176439, ..., -0.56605221,
        -0.47010945,  1.23964578],
       [ 0.90829868, -1.13435131, -1.26834061, ..., -0.56605221,
        -0.8903691 , -0.80668205]])
```

Generating the submission file

```
y_pred = Model.predict(X_train55)
y_pred = (y_pred > 0.25)
submission_df = pd.DataFrame({'ID': ID,'Is_Lead': y_pred.flatten().astype(int),
})
submission_df.set_index('ID', inplace=True)
submission_df.to_csv('final_submission.csv')
```


5s
completed at 7:47 PM