In [1]:

```
import pyodbc
```

In [2]:

```
conn = pyodbc.connect(r'DSN=tekkredi;UID=yavuzs;PWD=18651438-155E-4450-859D-803181407D18')
```

In [3]:

```
import pandas as pd
import numpy as np
```

In [4]:

```
#database connection and import table
df = pd.read_sql("select * from dbo.Sample5", conn)
```

In [5]:

```
df.head()
```

Out[5]:

	RatioCons	RatioCred	RatioOD	RatioMorg	RatioCar	RatioFullDebtLimit	RatioConsDebtLimit	RatioCredDebtLimit	Rati
0	0.307692	0.384615	0.307692	NaN	None	0.440091	0.260075	0.730099	0.68
1	0.777778	0.037037	0.185185	NaN	None	0.257212	0.201451	0.993843	NaN
2	0.538462	0.230769	0.230769	NaN	None	0.249186	0.258512	-0.000192	0.36
3	0.428571	0.250000	0.321429	NaN	None	0.179675	0.225764	0.076379	0.21
4	0.333333	0.208333	0.333333	0.125	None	0.425787	0.246722	0.136672	0.00

5 rows × 184 columns

In [ ]:

```
df_edit = df.drop(['ApplyId', 'ApplyTime', 'CustomerId', 'ApplyHour', 'ApplyMonth', 'Age', 'Gender'
, 'CityName',
                   'MonthlyCanBePaid', 'Income', 'Device', 'MailFlag', 'Source', 'Medium',
'Campaign', 'Adgroup',
                   'Keyword', 'Occupation', 'Education', 'Profession', 'WorkName', 'WorkCity',
'WorkCounty', 'WorkSector',
                   'WorkTitle', 'WorkPeriod', 'Homeowner', 'PreferBank1', 'PreferBank2', 'DeedType'
, 'PaymentMessage',
                   'PaymentStatus', 'CardType', 'InsScore', 'Bounced', 'Unsubscribed', 'FacebookId'
, 'IsDeleted'], axis=1)
```

In [ ]:

```
df_vars = df_edit.columns.values.tolist()
```

In [ ]:

```
df_model = df_edit[[i for i in df_vars if df_edit[i].isnull().sum() / len(df_edit) < 0.3]]
```

In [ ]:

```
df_vars_2 = df_model.columns.values.tolist()
```

In [ ]:

```
df_nonNA = df_model.dropna()
df_nonNA['Default'] = np.where(df_nonNA['Def'] > 0, 'Def', 'Perf')
y_nonNA = df_nonNA[['Default']]
X_nonNA = df_nonNA[[i for i in df_vars_2 if i not in df_nonNA[['Def', 'Default']]]]
```

In [ ]:

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
# create a base classifier used to evaluate a subset of attributes
model = LogisticRegression()
# create the RFE model and select 3 attributes
rfe = RFE(model, 10)
rfe = rfe.fit(X_nonNA.values, y_nonNA.values)
# summarize the selection of the attributes
print(rfe.support_)
print(rfe.ranking_)
```

In [ ]:

```
list(X_nonNA)
```

In [ ]:

```
from sklearn import tree
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_nonNA.values, y_nonNA.values)
```

In [ ]:

```
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None,
                                feature_names=X_nonNA.columns.values.tolist(),
                                class_names=y_nonNA['Default'].unique(),
                                max_depth=3,
                                filled=True, rounded=True,
                                special_characters=True)
graph = graphviz.Source(dot_data)
graph.render('tree2.png')
```

In [ ]:

```
correlations = df_nonNA[['Def', 'Score', 'FullAvg60Mon6', 'CredAvg60Mon18', 'Ratio30of3to18', 'Full
Avg90Mon3',
                                'CredAvg90Mon18', 'FullAvg90Mon18', 'FullAvg60Mon3',
'RatioConsDebtLimit']].corr()
```

In [ ]:

```
correlations
```

In [ ]:

```
corr_all = df_nonNA.drop(['Default'], axis=1).corr()
```

In [ ]:

```
corr_all[['Def']][(corr_all.Def >= 0.25) | (corr_all.Def <= -0.25)]
```

In [ ]:

```
corr_25 = corr_all[list(corr_all[['Def']][(corr_all.Def >= 0.25) | (corr_all.Def <= -0.25)].index)]
[(corr_all.Def >= 0.25) | (corr_all.Def <= -0.25)]
```

In [ ]:

```
df_25 = df_nonNA[list(corr_all[['Def']][(corr_all.Def >= 0.25) | (corr_all.Def <= -0.25)].index)]
```

In [ ]:

```
corr_25
```

In [ ]:

```
list(corr_all[['Def']][(corr_all.Def >= 0.25) | (corr_all.Def <= -0.25)].index)
```

In [ ]:

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.subplots(figsize=(15, 10))
sns.heatmap(corr_25, annot=True, cmap="RdYlGn")
plt.show()
```

In [6]:

```
df_2 = df[['Score', 'ConsAvgOnTimeMon3', 'FullAvg30Mon6', 'FullAvg60Mon6', 'ConsAvg60Mon6',
'CredAvg60Mon6',
           'FullAvg90Mon6', 'CredAvgOnTimeMon6', 'Def']]
```

In [15]:

```
df_2['Score'] = df_2['Score'].fillna(df_2['Score'].mean())
df_2['ConsAvgOnTimeMon3'] = df_2['ConsAvgOnTimeMon3'].fillna(df_2['ConsAvgOnTimeMon3'].mean())
df_2['FullAvg30Mon6'] = df_2['FullAvg30Mon6'].fillna(df_2['FullAvg30Mon6'].mean())
df_2['FullAvg60Mon6'] = df_2['FullAvg60Mon6'].fillna(df_2['FullAvg60Mon6'].mean())
df_2['ConsAvg60Mon6'] = df_2['ConsAvg60Mon6'].fillna(df_2['ConsAvg60Mon6'].mean())
df_2['CredAvg60Mon6'] = df_2['CredAvg60Mon6'].fillna(df_2['CredAvg60Mon6'].mean())
df_2['FullAvg90Mon6'] = df_2['FullAvg90Mon6'].fillna(df_2['FullAvg90Mon6'].mean())
df_2['CredAvgOnTimeMon6'] = df_2['CredAvgOnTimeMon6'].fillna(df_2['CredAvgOnTimeMon6'].mean())
```

```
C:\Users\Tekkredi\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  """Entry point for launching an IPython kernel.
C:\Users\Tekkredi\Anaconda3\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy

C:\Users\Tekkredi\Anaconda3\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  This is separate from the ipykernel package so we can avoid doing imports until
C:\Users\Tekkredi\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  after removing the cwd from sys.path.
C:\Users\Tekkredi\Anaconda3\lib\site-packages\ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  """
C:\Users\Tekkredi\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy

C:\Users\Tekkredi\Anaconda3\lib\site-packages\ipykernel_launcher.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  import sys
C:\Users\Tekkredi\Anaconda3\lib\site-packages\ipykernel_launcher.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
```

In [16]:

```
df_2[df_2.isnull().any(axis=1)]
```

Out[16]:

	Score	ConsAvgOnTimeMon3	FullAvg30Mon6	FullAvg60Mon6	ConsAvg60Mon6	CredAvg60Mon6	FullAvg90Mon6	Cred

In [21]:

```
df_2_vars=df_2.columns.values.tolist()
```

In [22]:

```
y=df_2[['Def']]
X=df_2[[i for i in df_2_vars if i not in y]]
```

In [24]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

In [25]:

```
df_2_test= pd.concat([X_test, y_test], axis=1, join='inner')
```

In [26]:

```
#standardized xtrain
X_train_std = X_train.std(0)
X_train_mean = X_train.mean(0)
X_train_edit=(X_train-X_train_mean)/X_train_std
```

In [27]:

```
#standardized xtest
X_test_std = X_test.std(0)
X_test_mean = X_test.mean(0)
X_test_edit=(X_test-X_test_mean)/X_test_std
```

In [28]:

```
# Parameters initialization
weights = np.random.normal(0, 0.1, 8)
bias = np.random.normal(0, 0.1)
lr = 0.05
n = X_train_edit.shape[0]
```

In [29]:

```
for epoch in range(300):

    # Logistic function
    Z = np.dot(X_train_edit, weights) + bias
    A = 1 / (1 + np.exp(-Z))

    # Negative log likelihood -- loss function
    J = np.sum(-(y_train['Def'] * np.log(A) + (1 - y_train['Def'])* np.log(1 - A))) / n

    # Gradient computation
    dZ = A - y_train['Def']
    dw = np.dot(dZ, X_train_edit) / n
    db = np.sum(dZ) / n

    # Update weights
    weights = weights - lr * dw
    bias = bias - lr * db

    if epoch % 10 == 0:
        print("epoch %s - loss %s" % (epoch, J))
```

```
epoch 0 - loss 0.7136806606952285
epoch 10 - loss 0.6478192984829566
epoch 20 - loss 0.6195434028333981
epoch 30 - loss 0.6060505799827667
epoch 40 - loss 0.5989744465437518
epoch 50 - loss 0.5949548313742183
epoch 60 - loss 0.5925018612708688
epoch 70 - loss 0.590902000260696
epoch 80 - loss 0.5897924459794422
epoch 90 - loss 0.588979569582716
epoch 100 - loss 0.588355658516485
epoch 110 - loss 0.5878584643813655
epoch 120 - loss 0.5874506240833884
epoch 130 - loss 0.5871088026300271
epoch 140 - loss 0.586817794825525
epoch 150 - loss 0.5865672420015187
epoch 160 - loss 0.5863497648100937
epoch 170 - loss 0.586159879438357
epoch 180 - loss 0.5859933550683916
epoch 190 - loss 0.5858468238185781
epoch 200 - loss 0.5857175373591511
epoch 210 - loss 0.5856032101021257
epoch 220 - loss 0.5855019144269042
epoch 230 - loss 0.5854120078689085
epoch 240 - loss 0.5853320804661958
epoch 250 - loss 0.5852609152247471
epoch 260 - loss 0.5851974574328582
epoch 270 - loss 0.5851407901790132
epoch 280 - loss 0.5850901143897208
epoch 290 - loss 0.5850447322813767
```

In [30]:

```
#logreg prediction score for test dataset
pred_score = 1 / (1 + np.exp(-(np.dot(X_test_edit, weights) + bias)))
```

In [31]:

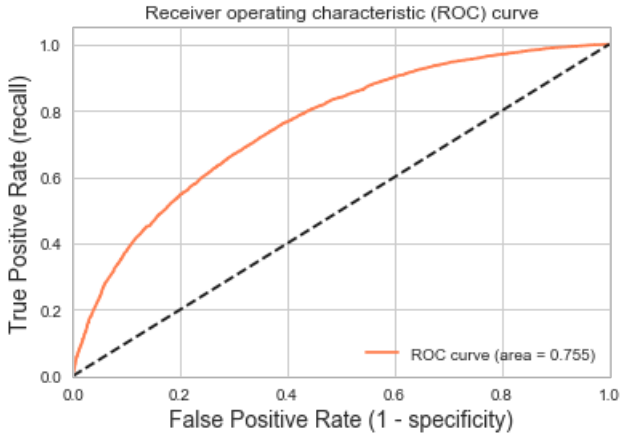
```
from sklearn.metrics import roc_curve, auc, log_loss, accuracy_score, confusion_matrix
[fpr, tpr, thr] = roc_curve(y_test, pred_score)
```

In [32]:

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="white") #white background style for seaborn plots
sns.set(style="whitegrid", color_codes=True)
```

In [33]:

```
#draw the roc curve
plt.figure()
plt.plot(fpr, tpr, color='coral', label='ROC curve (area = %0.3f)' % auc(fpr, tpr))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (1 - specificity)', fontsize=14)
plt.ylabel('True Positive Rate (recall)', fontsize=14)
plt.title('Receiver operating characteristic (ROC) curve')
plt.legend(loc="lower right")
plt.show()
```



In [35]:

```
#calculate gini
2*0.755-1
```

Out[35]:

0.51

In [37]:

```
#adding predicted probability results to test data
df_test_result = df_2_test
df_test_result = df_test_result.assign(pred_prob=pred_score)
```

In [38]:

```
#creating deciles for ranking
df_test_result['decile_pred'] = pd.qcut(df_test_result['pred_prob'], 10, labels=np.arange(10, 0, -1
))
df_test_result['decile_Bureau'] = pd.qcut(df_test_result['Score'].rank(method='first'), 10, labels=
False)+1

#making deciles integers
df_test_result['decile_pred']=df_test_result['decile_pred'].astype(int)
df_test_result['decile_Bureau']=df_test_result['decile_Bureau'].astype(int)
```

In [39]:

```
#creating table for deciles vs. bad rate
badrate_vs_decile_pred = pd.pivot_table(df_test_result, values='pred_prob', index=['decile_pred'],
                     columns=['Def'], aggfunc='count')
badrate_vs_decile_Bureau = pd.pivot_table(df_test_result, values='Score', index=['decile_Bureau'],
                     columns=['Def'], aggfunc='count')

rank_pred = pd.DataFrame(badrate_vs_decile_pred.to_records())
rank_Bureau = pd.DataFrame(badrate_vs_decile_Bureau.to_records())

rank_pred['decile'] = rank_pred['decile_pred']
rank_Bureau['decile'] = rank_Bureau['decile_Bureau']

rank pred['bad rate']=rank pred['1']/(rank pred['0']+rank pred['1'])
```

```
rank_Bureau['bad_rate']=rank_Bureau['1']/(rank_Bureau['0']+rank_Bureau['1'])

rank_merge=rank_pred.merge(rank_Bureau, left_on='decile', right_on='decile', how='left')
```

In [40]:

```
rank_merge
```

Out[40]:

	decile_pred	0_x	1_x	decile	bad_rate_x	decile_Bureau	0_y	1_y	bad_rate_y
0	1	312	1350	1	0.812274	1	592	1070	0.643803
1	2	466	1196	2	0.719615	2	534	1128	0.678700
2	3	644	1017	3	0.612282	3	502	1159	0.697772
3	4	767	895	4	0.538508	4	613	1049	0.631167
4	5	871	790	5	0.475617	5	820	842	0.506619
5	6	941	721	6	0.433815	6	956	705	0.424443
6	7	1076	586	7	0.352587	7	1074	588	0.353791
7	8	1199	462	8	0.278146	8	1206	455	0.273931
8	9	1375	287	9	0.172684	9	1352	310	0.186522
9	10	1491	171	10	0.102888	10	1493	169	0.101685

In [41]:

```
plt.plot(rank_merge['decile'], rank_merge['bad_rate_x'], color='blue')
plt.plot(rank_merge['decile'], rank_merge['bad_rate_y'], color='red')
plt.xlim((1, 10))
plt.ylim((0.0, 1.0))
plt.xticks(np.arange(1, 11, 1))
plt.show()
```