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 [6]:

```
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 [7]:

```
df_2 = df[['Def', 'Score', 'RatioConsDebtLimit', 'Ratio90of3to12', 'Ratio90of12to18',
'FullAvg30Mon3', 'FullAvg60Mon3',
           'ConsAvg90Mon3', 'CredAvg90Mon3', 'ODAvg90Mon3', 'FullAvg90Mon12']]
```

In [ ]:

```
df_2['Score_Inv'] = np.where(df_2.Score == 0, 1 / min(df_2.Score[df_2.Score > 0]), 1 / df_2.Score)
df_2['RatioConsDebtLimit_Inv'] = np.where(df_2.RatioConsDebtLimit == 0, 1 / min(df_2.RatioConsDebtL
imit[df_2.RatioConsDebtLimit > 0]), 1 / df_2.RatioConsDebtLimit)
df_2['Ratio90of3to12_Inv'] = np.where(df_2.Ratio90of3to12 == 0, 1 / min(df_2.Ratio90of3to12[df_2.Ra
tio90of3to12 > 0]), 1 / df_2.Ratio90of3to12)
df_2['Ratio90of12to18_Inv'] = np.where(df_2.Ratio90of12to18 == 0, 1 / min(df_2.Ratio90of12to18[df_2
.Ratio90of12to18 > 0]), 1 / df_2.Ratio90of12to18)
df_2['FullAvg30Mon3_Inv'] = np.where(df_2.FullAvg30Mon3 == 0, 1 / min(df_2.FullAvg30Mon3[df_2.FullA
vg30Mon3 > 0]), 1 / df_2.FullAvg30Mon3)
df_2['FullAvg60Mon3_Inv'] = np.where(df_2.FullAvg60Mon3 == 0, 1 / min(df_2.FullAvg60Mon3[df_2.FullA
vg60Mon3 > 0]), 1 / df_2.FullAvg60Mon3)
df_2['ConsAvg90Mon3_Inv'] = np.where(df_2.ConsAvg90Mon3 == 0, 1 / min(df_2.ConsAvg90Mon3[df_2.ConsA
vg90Mon3 > 0]), 1 / df_2.ConsAvg90Mon3)
df_2['CredAvg90Mon3_Inv'] = np.where(df_2.CredAvg90Mon3 == 0, 1 / min(df_2.CredAvg90Mon3[df_2.CredA
vg90Mon3 > 0]), 1 / df_2.CredAvg90Mon3)
df_2['ODAvg90Mon3_Inv'] = np.where(df_2.ODAvg90Mon3 == 0, 1 / min(df_2.ODAvg90Mon3[df_2.ODAvg90Mon3
> 0]), 1 / df_2.ODAvg90Mon3)
df_2['FullAvg90Mon12_Inv'] = np.where(df_2.FullAvg90Mon12 == 0, 1 / min(df_2.FullAvg90Mon12[df_2.Fu
llAvg90Mon12 > 0]), 1 / df_2.FullAvg90Mon12)

df_2['Score_Log'] = np.where(df_2.Score == 0, np.log(min(df_2.Score[df_2.Score > 0])), np.log(df_2.
Score))
df_2['RatioConsDebtLimit_Log'] = np.where(df_2.RatioConsDebtLimit == 0, np.log(min(df_2.RatioConsDe
btLimit[df_2.RatioConsDebtLimit > 0])), np.log(df_2.RatioConsDebtLimit))
df_2['Ratio90of3to12_Log'] = np.where(df_2.Ratio90of3to12 == 0, np.log(min(df_2.Ratio90of3to12[df_2
.Ratio90of3to12 > 0])), np.log(df_2.Ratio90of3to12))
df_2['Ratio90of12to18_Log'] = np.where(df_2.Ratio90of12to18 == 0, np.log(min(df_2.Ratio90of12to18[d
f_2.Ratio90of12to18 > 0])), np.log(df_2.Ratio90of12to18))
df_2['FullAvg30Mon3_Log'] = np.where(df_2.FullAvg30Mon3 == 0, np.log(min(df_2.FullAvg30Mon3[df_2.Fu
llAvg30Mon3 > 0])), np.log(df_2.FullAvg30Mon3))
df_2['FullAvg60Mon3_Log'] = np.where(df_2.FullAvg60Mon3 == 0, np.log(min(df_2.FullAvg60Mon3[df_2.Fu
llAvg60Mon3 > 0])), np.log(df_2.FullAvg60Mon3))
df_2['ConsAvg90Mon3_Log'] = np.where(df_2.ConsAvg90Mon3 == 0, np.log(min(df_2.ConsAvg90Mon3[df_2.Co
nsAvg90Mon3 > 0])), np.log(df_2.ConsAvg90Mon3))
df_2['CredAvg90Mon3_Log'] = np.where(df_2.CredAvg90Mon3 == 0, np.log(min(df_2.CredAvg90Mon3[df_2.Cr
edAvg90Mon3 > 0])), np.log(df_2.CredAvg90Mon3))
df_2['ODAvg90Mon3_Log'] = np.where(df_2.ODAvg90Mon3 == 0, np.log(min(df_2.ODAvg90Mon3[df_2.ODAvg90M
on3 > 0])), np.log(df_2.ODAvg90Mon3))
df_2['FullAvg90Mon12_Log'] = np.where(df_2.FullAvg90Mon12 == 0, np.log(min(df_2.FullAvg90Mon12[df_2
.FullAvg90Mon12 > 0])), np.log(df_2.FullAvg90Mon12))
```

In [ ]:

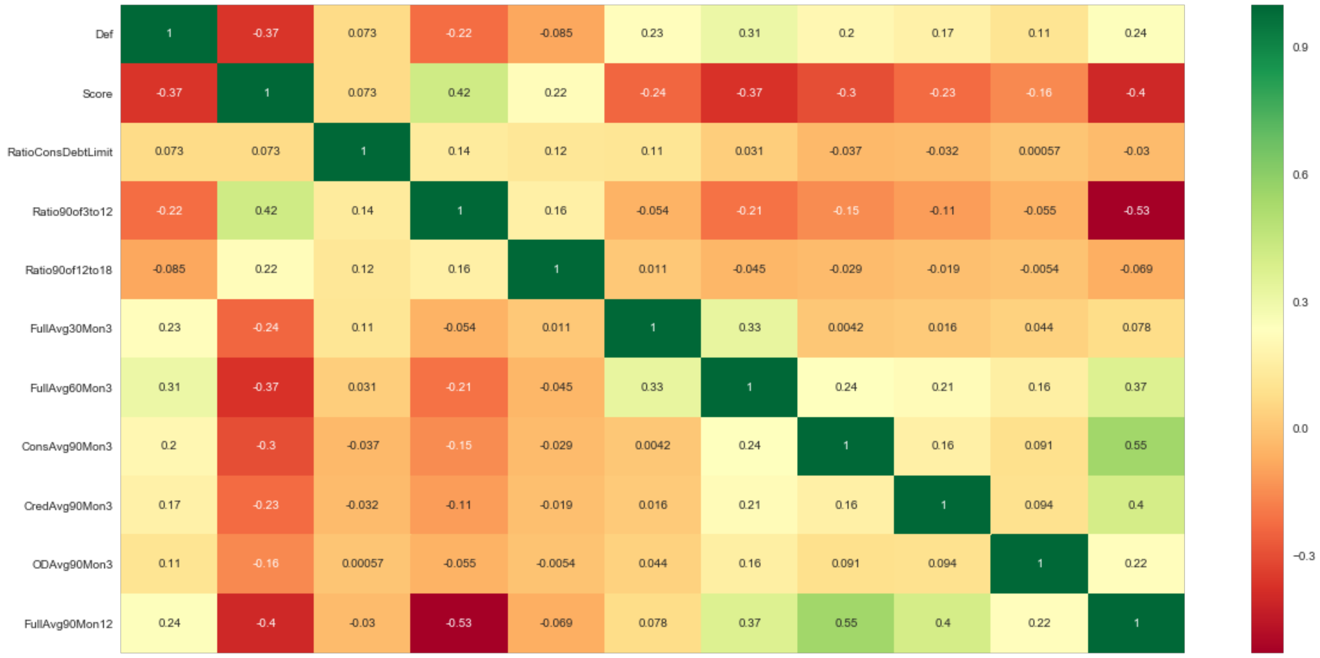
```
df_2.describe()
```

In [43]:

```
corr_1 = df_2.corr()
```

In [44]:

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.subplots(figsize=(20, 10))
sns.heatmap(corr_1, annot=True, cmap="RdYlGn")
plt.show()
```



Def	Score	RatioConsDebtLimit	Ratio90of3to12	Ratio90of12to18	FullAvg30Mon3	FullAvg60Mon3	ConsAvg90Mon3	CredAvg90Mon3	ODAvg90Mon3	FullAvg90Mon12

In [8]:

```
df_2['RatioConsDebtLimit'] = df_2['RatioConsDebtLimit'].fillna(df_2['RatioConsDebtLimit'].mean())
df_2['Ratio90of3to12'] = df_2['Ratio90of3to12'].fillna(df_2['Ratio90of3to12'].mean())
df_2['Ratio90of12to18'] = df_2['Ratio90of12to18'].fillna(df_2['Ratio90of12to18'].mean())
df_2['FullAvg30Mon3'] = df_2['FullAvg30Mon3'].fillna(df_2['FullAvg30Mon3'].mean())
df_2['FullAvg60Mon3'] = df_2['FullAvg60Mon3'].fillna(df_2['FullAvg60Mon3'].mean())
df_2['ConsAvg90Mon3'] = df_2['ConsAvg90Mon3'].fillna(df_2['ConsAvg90Mon3'].mean())
df_2['CredAvg90Mon3'] = df_2['CredAvg90Mon3'].fillna(df_2['CredAvg90Mon3'].mean())
df_2['ODAvg90Mon3'] = df_2['ODAvg90Mon3'].fillna(df_2['ODAvg90Mon3'].mean())
df_2['FullAvg90Mon12'] = df_2['FullAvg90Mon12'].fillna(df_2['FullAvg90Mon12'].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

C:\Users\Tekkredi\Anaconda3\lib\site-packages\ipykernel_launcher.py:9: 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
  if __name__ == '__main__':
```

```
if __name__ == '__main__':
```

In [9]:

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

Out[9]:

	Def	Score	RatioConsDebtLimit	Ratio90of3to12	Ratio90of12to18	FullAvg30Mon3	FullAvg60Mon3	ConsAvg90Mon3	Cre

In [11]:

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

In [35]:

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

In [36]:

```
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 [37]:

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

In [45]:

```
#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 [46]:

```
#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 [47]:

```
# Parameters initialization
weights = np.random.normal(0, 0.1, 10)
bias = np.random.normal(0, 0.1)
lr = 0.05
n = X_train_edit.shape[0]
```

In [48]:

```
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 0.6731099431713983
epoch 10 - loss 0.6367877695717422
epoch 20 - loss 0.616139112812915
epoch 30 - loss 0.6036143608295776
epoch 40 - loss 0.5956123119537848
epoch 50 - loss 0.5902864743232956
epoch 60 - loss 0.5866222149437182
epoch 70 - loss 0.5840299688584913
epoch 80 - loss 0.5821515005539408
epoch 90 - loss 0.580761053907593
epoch 100 - loss 0.5797119674610682
epoch 110 - loss 0.5789064648608515
epoch 120 - loss 0.5782778822504294
epoch 130 - loss 0.5777798634271202
epoch 140 - loss 0.577379603374761
epoch 150 - loss 0.5770535184529614
epoch 160 - loss 0.5767844110263288
epoch 170 - loss 0.5765595765369712
epoch 180 - loss 0.5763695175681626
epoch 190 - loss 0.5762070563103725
epoch 200 - loss 0.5760667130422781
epoch 210 - loss 0.575944265037956
epoch 220 - loss 0.5758364296303956
epoch 230 - loss 0.5757406338630138
epoch 240 - loss 0.5756548452870904
epoch 250 - loss 0.575577446445119
epoch 260 - loss 0.5755071409080369
epoch 270 - loss 0.5754428823372971
epoch 280 - loss 0.5753838205091868
epoch 290 - loss 0.5753292599468717
```

In [62]:

```
weights, bias
```

Out[62]:

```
(array([-0.4608249 ,  0.21033903, -0.14422634, -0.06532443,  0.27781995,
         0.40399846,  0.2029573 ,  0.16374308,  0.09746318,  0.07586096]),
 -0.16477774283800228)
```

In [50]:

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

In [51]:

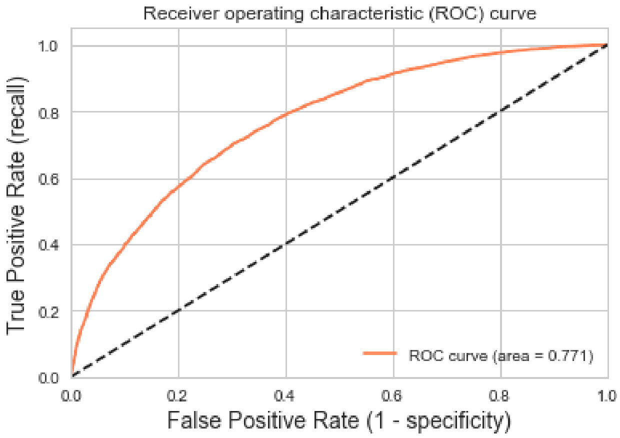
```
from sklearn.metrics import roc_curve, auc, log_loss, accuracy_score, confusion_matrix
[fpr, tpr, thr] = roc_curve(y_test, pred_score)
```

In [52]:

```
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 [53]:

```
#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 [54]:

```
#calculate gini
2*0.771-1
```

Out[54]:

```
0.542
```

In [55]:

```
#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 [57]:

```
#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 [58]:

```
#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 [59]:

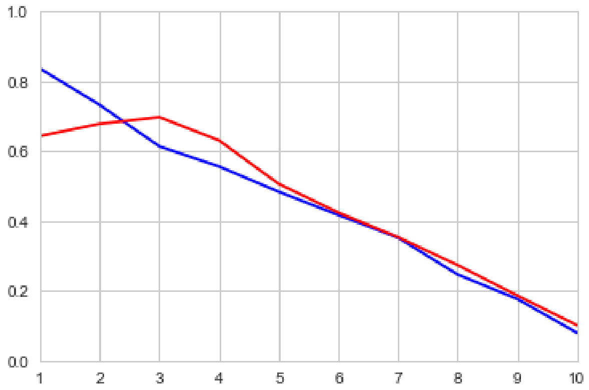
```
rank_merge
```

Out[59]:

	decile_pred	0_x	1_x	decile	bad_rate_x	decile_Bureau	0_y	1_y	bad_rate_y
0	1	271	1391	1	0.836943	1	592	1070	0.643803
1	2	444	1218	2	0.732852	2	534	1128	0.678700
2	3	641	1020	3	0.614088	3	502	1159	0.697772
3	4	737	925	4	0.556558	4	613	1049	0.631167
4	5	857	804	5	0.484046	5	820	842	0.506619
5	6	968	694	6	0.417569	6	956	705	0.424443
6	7	1076	586	7	0.352587	7	1074	588	0.353791
7	8	1251	410	8	0.246839	8	1206	455	0.273931
8	9	1368	294	9	0.176895	9	1352	310	0.186522
9	10	1529	133	10	0.080024	10	1493	169	0.101685

In [60]:

```
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()
```


1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
5
6
7
8
9
10