In [3]:

```python
import pyodbc
```

In [4]:

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

In [5]:

```python
import pandas as pd
```

In [6]:

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

In [293]:

```python
df.head()
```

Out[293]:

	index	ca_customertransactionid	c_gender	ca_avgmonthlycanbepaid	ca_avgpayrollincome	ca_maxmonthlycanbepaid
0	2243	{C0ECCA37-216A-41C5-8139-2400B19C8A6B}	Female	750	1850	750
1	2244	{C0F69883-CBD4-4508-B483-0139CEC9C169}	Female	250	1250	500
2	2245	{C101FE5A-C985-49F5-BBB9-1E9F85647618}	Male	1500	1239	1500
3	2246	{C110DC3F-FB6B-41C1-A49F-F035A48C7703}	Male	2400	3850	2400
4	2247	{C1170BB3-ECCC-4DDA-A345-BB19855ECE97}	Male	1750	1250	2000

5 rows × 32 columns

In [7]:

```python
#make binary target integer and check it
df['target_var']=df['target_var'].astype(int)
```

In [12]:

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2983 entries, 0 to 2982
Data columns (total 32 columns):
index
2983 non-null object
ca_customertransactionid
2983 non-null object
c_gender
2983 non-null object
ca_avgmonthlycanbepaid
2983 non-null int64
ca_avgpayrollincome
2983 non-null int64
ca_maxmonthlycanbepaid
2983 non-null int64
```

```
ca_maxpayrollincome
2983 non-null int64
ca_minmonthlycanbepaid
2983 non-null int64
ca_minpayrollincome
2983 non-null int64
ca_occupation
2983 non-null object
ca_preferbank1
2983 non-null object
ca_preferbank2
2983 non-null object
VAR_ca_score
2983 non-null int64
ca_totalamount
2983 non-null float64
cs_education
2983 non-null object
cs_homeowner
2983 non-null object
cs_workcity
2983 non-null object
cs_workperiod
2983 non-null object
cs_worksector
2983 non-null object
cs_worktitle
2983 non-null object
VAR_ctb_average_months_on_time_x_creditcard_loan_last_3months
2983 non-null float64
ctb_avg_months_60day_delinquent_x_open_loan_x_personal_loan_last_6months
2983 non-null float64
ctb_avg_months_90day_delinquent_last_18months
2983 non-null float64
ctb_avg_months_90day_delinquent_x_overdraft_acct_last_6months
2983 non-null float64
avg_ratio_totaldebt_to_creditlimit_last_3months
2983 non-null float64
ratio_avg_months_30day_delinquent_last_6months_to_avg_months_30day_delinquent_last_18months
2983 non-null float64
VAR_Inverse_of_ratio_avg_months_30day_delinquent_last_6months_to_avg_months_30day_delinquent_last_1
ths    2983 non-null float64
VAR_Inverse_of_ctb_avg_months_90day_delinquent_last_18months
2983 non-null float64
VAR_Log_of_ctb_avg_months_60day_delinquent_x_open_loan_x_personal_loan_last_6months
2983 non-null float64
VAR_Log_of_avg_ratio_totaldebt_to_creditlimit_last_3months
2983 non-null float64
VAR_Inverse_of_ctb_avg_months_90day_delinquent_x_overdraft_acct_last_6months
2983 non-null float64
target_var
2983 non-null int32
dtypes: float64(12), int32(1), int64(7), object(12)
memory usage: 734.2+ KB
```

In [13]:

```
df.describe()
```

Out[13]:

	ca_avgmonthlycanbepaid	ca_avgpayrollincome	ca_maxmonthlycanbepaid	ca_maxpayrollincome	ca_minmonthlyc
count	2983.000000	2983.000000	2983.000000	2983.000000	2983.000000
mean	1322.834060	2523.687563	1438.908481	2668.005364	1206.731478
std	1137.273983	1993.746431	1184.830968	2119.980183	1110.778669
min	40.000000	50.000000	40.000000	100.000000	0.000000
25%	600.000000	1500.000000	600.000000	1500.000000	500.000000
50%	1000.000000	2000.000000	1000.000000	2000.000000	1000.000000
75%	1750.000000	3000.000000	2000.000000	3000.000000	1500.000000

max	13000.000000	45000.000000	13000.000000	50000.000000	13000.000000
	ca_avgmonthlycanbepaid	ca_avgpayrollincome	ca_maxmonthlycanbepaid	ca_maxpayrollincome	ca_minmonthlyc

In [8]:

```
#modify table just for model variables
df_model=df[['VAR_ca_score', 'VAR_ctb_average_months_on_time_x_creditcard_loan_last_3months',

'VAR_Inverse_of_ratio_avg_months_30day_delinquent_last_6months_to_avg_months_30day_delinquent_last_
nths',
           'VAR_Inverse_of_ctb_avg_months_90day_delinquent_last_18months',
           'VAR_Log_of_ctb_avg_months_60day_delinquent_x_open_loan_x_personal_loan_last_6months',
           'VAR_Log_of_avg_ratio_totaldebt_to_creditlimit_last_3months',
           'VAR_Inverse_of_ctb_avg_months_90day_delinquent_x_overdraft_acct_last_6months','target_v
ar']]
```

In [9]:

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

In [107]:

```
y=df_model[['target_var']]
x=df_model[[i for i in df_vars if i not in y]]
```

In [85]:

```
from sklearn import datasets
from sklearn.linear_model import LogisticRegression
import numpy as np
```

In [108]:

```
logreg = LogisticRegression()
```

In [112]:

```
#apply sklearn logistic Regression
logreg.fit(x,y)
```

```
C:\Users\Tekkredi\Anaconda3\lib\site-packages\sklearn\utils\validation.py:578:
DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change th
e shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[112]:

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

In [113]:

```
list(x)
```

Out[113]:

```
['VAR_ca_score',
 'VAR_ctb_average_months_on_time_x_creditcard_loan_last_3months',

'VAR_Inverse_of_ratio_avg_months_30day_delinquent_last_6months_to_avg_months_30day_delinquent_last_
nths',
 'VAR_Inverse_of_ctb_avg_months_90day_delinquent_last_18months',
 'VAR_Log_of_ctb_avg_months_60day_delinquent_x_open_loan_x_personal_loan_last_6months',
 'VAR_Log_of_avg_ratio_totaldebt_to_creditlimit_last_3months',
 'VAR_Inverse_of_ctb_avg_months_90day_delinquent_x_overdraft_acct_last_6months']
```

In [122]:

```
#see coefficients and intercept
logreg.coef_
```

Out[122]:

```
array([[-1.10015689e-03, -2.56399666e-01, -7.17939079e-02,
        -3.10114905e-02,  3.40134336e-01,  1.89733498e+00,
        -1.56054970e-02]])
```

In [121]:

```
logreg.intercept_
```

Out[121]:

```
array([2.16891027])
```

In [136]:

```
#create predicted class(sklearn predict function works with 0.5 class prob. threshold)
y_pred = logreg.predict(x)
```

In [197]:

```
#create class probabilities
y_pred_proba = logreg.predict_proba(x)[:, 1]
```

In [298]:

```
#create predicted class for the 0.3 class prob. threshold)
y_pred = (y_pred_proba> 0.3).astype(int)
```

In [249]:

```
from sklearn.metrics import roc_curve, auc, log_loss, accuracy_score, confusion_matrix
```

In [173]:

```
#calculate roc curve measures
[fpr, tpr, thr] = roc_curve(y, y_pred_proba)
```

In [300]:

```
print(logreg.__class__.__name__+" accuracy is %2.3f" % accuracy_score(y, y_pred))
print(logreg.__class__.__name__+" log_loss is %2.3f" % log_loss(y, y_pred_proba))
print(logreg.__class__.__name__+" auc is %2.3f" % auc(fpr, tpr))
```

```
LogisticRegression accuracy is 0.764
LogisticRegression log_loss is 0.456
LogisticRegression auc is 0.814
```

In [302]:

```
#calculate the confusion matrix
tn, fp, fn, tp = confusion_matrix(y, y_pred).ravel()
```

In [303]:

```
(tn, fp, fn, tp)
```

Out[303]:

```
(1755, 410, 294, 524)
```

In [304]:

```
print("TPR is %2.3f" % (tp/(tp+fn)))
print("FPR is %2.3f" % (fp/(fp+tn)))
```

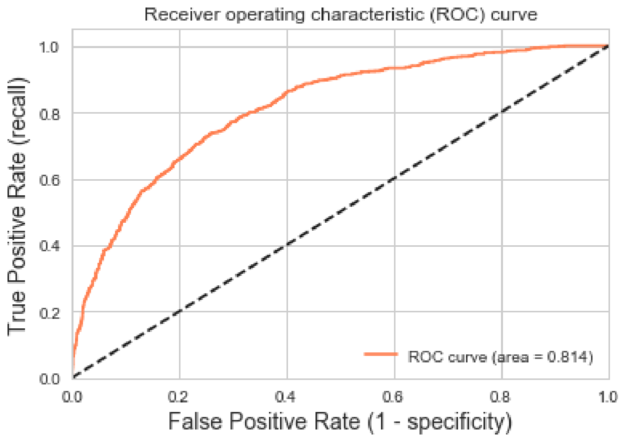
```
TPR is 0.641
FPR is 0.189
```

In [148]:

```
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 [305]:

```
#draw the roc curve for sklearn logreg model
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 [131]:

```
#import statsmodels to apply a 2. logreg model
import statsmodels.api as sm
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
```

In [124]:

```
#create a constant for statsmodels logreg intercept
x_sm=sm.add_constant(x, prepend=False)
```

In [133]:

```
#apply statsmodels logistic Regression
model=sm.Logit(y,x_sm)
```

In [134]:

```
result=model.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.455363
         Iterations 7
```

In [135]:

```
#see coefficients and intercept
result.summary()
```

Out[135]:

Logit Regression Results

Dep. Variable:	target_var	**No. Observations:**	2983
Model:	Logit	**Df Residuals:**	2975
Method:	MLE	**Df Model:**	7
Date:	Fri, 06 Apr 2018	**Pseudo R-squ.:**	0.2248
Time:	18:11:31	**Log-Likelihood:**	-1358.3
converged:	True	**LL-Null:**	-1752.2
		LLR p-value:	7.958e-166

	co
VAR_ca_score	- 0.0
VAR_ctb_average_months_on_time_x_creditcard_loan_last_3months	- 0.2
VAR_Inverse_of_ratio_avg_months_30day_delinquent_last_6months_to_avg_months_30day_delinquent_last_18months	- 0.0
VAR_Inverse_of_ctb_avg_months_90day_delinquent_last_18months	- 0.0
VAR_Log_of_ctb_avg_months_60day_delinquent_x_open_loan_x_personal_loan_last_6months	0.3
VAR_Log_of_avg_ratio_totaldebt_to_creditlimit_last_3months	2.5
VAR_Inverse_of_ctb_avg_months_90day_delinquent_x_overdraft_acct_last_6months	- 0.0
const	2.1

In [306]:

```
#create class probabilities
y_pred_proba_sm=result.predict(x_sm)
```

In [307]:

```
#create predicted class for the 0.3 class prob. threshold)
y_pred_sm = (y_pred_proba_sm > 0.3).astype(int)
```

In [308]:

```
#calculate roc curve measures
[fpr, tpr, thr] = roc_curve(y, y_pred_proba_sm)
```

In [310]:

```
print(logreg.__class__.__name__+" accuracy is %2.3f" % accuracy_score(y, y_pred_sm))
print(logreg.__class__.__name__+" log_loss is %2.3f" % log_loss(y, y_pred_proba_sm))
print(logreg.__class__.__name__+" auc is %2.3f" % auc(fpr, tpr))
```

```
LogisticRegression accuracy is 0.764
LogisticRegression log_loss is 0.455
LogisticRegression auc is 0.814
```

In [286]:

```
#calculate the confusion matrix
tn, fp, fn, tp = confusion_matrix(y, y_pred_sm).ravel()
```

In [311]:

```
(tn, fp, fn, tp)
```

Out[311]:

```
(1755, 410, 294, 524)
```

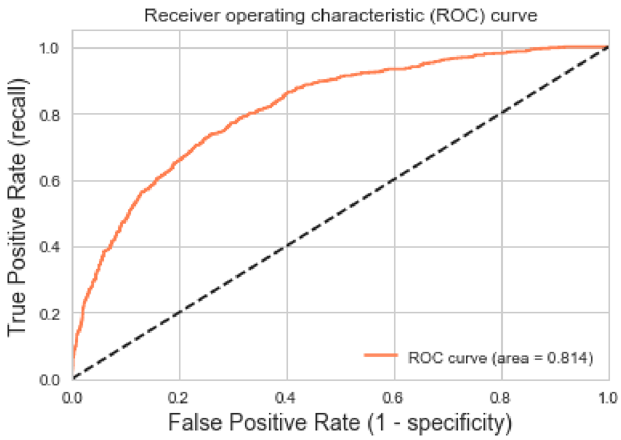
In [288]:

```
print("TPR is %2.3f" % (tp/(tp+fn)))
print("FPR is %2.3f" % (fp/(fp+tn)))
```

```
TPR is 0.652
FPR is 0.194
```

In [312]:

```
#draw the roc curve for statsmodels logreg model
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 [347]:

```
#same procedures for test dataset
df_test = pd.read_sql("select * from dbo.IndusTest_Model", conn)
```

In [228]:

```
df_test['target_var']=df_test['target_var'].astype(int)
```

In [229]:

```
df_test_model=df_test[['VAR_ca_score',
'VAR_ctb_average_months_on_time_x_creditcard_loan_last_3months',

'VAR_Inverse_of_ratio_avg_months_30day_delinquent_last_6months_to_avg_months_30day_delinquent_last_
nths',
               'VAR_Inverse_of_ctb_avg_months_90day_delinquent_last_18months',
'VAR_Log_of_ctb_avg_months_60day_delinquent_x_open_loan_x_personal_loan_last_6months',
               'VAR_Log_of_avg_ratio_totaldebt_to_creditlimit_last_3months',
               'VAR_Inverse_of_ctb_avg_months_90day_delinquent_x_overdraft_acct_last_6months','targ
et_var']]
```

In [2]:

```
df_test_vars=df_test_model.columns.values.tolist()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-d678568fd4f9> in <module>()
----> 1 df_test_vars=df_test_model.columns.values.tolist()

NameError: name 'df_test_model' is not defined
```

In [1]:

```
df_test_vars
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-197d63d53b70> in <module>()
----> 1 df_test_vars

NameError: name 'df_test_vars' is not defined
```

In [316]:

```
y_test=df_test_model[['target_var']]
x_test=df_test_model[[i for i in df_vars if i not in y]]
```

In [325]:

```
x_test=sm.add_constant(x_test, prepend=False)
```

In [326]:

```
y_test_pred_proba=result.predict(x_test)
```

In [327]:

```
y_test_pred = (y_test_pred_proba > 0.3).astype(int)
```

In [328]:

```
[fpr, tpr, thr] = roc_curve(y_test, y_test_pred_proba)
```

In [331]:

```
print(logreg.__class__.__name__+" accuracy is %2.3f" % accuracy_score(y_test, y_test_pred))
print(logreg.__class__.__name__+" log_loss is %2.3f" % log_loss(y_test, y_test_pred_proba))
print(logreg.__class__.__name__+" auc is %2.3f" % auc(fpr, tpr))
```

```
LogisticRegression accuracy is 0.748
LogisticRegression log_loss is 0.506
LogisticRegression auc is 0.785
```

In [335]:

```
#calculate the confusion matrix
tn, fp, fn, tp = confusion_matrix(y_test, y_test_pred).ravel()
```

In [336]:

```
(tn, fp, fn, tp)
```

```
(tn, fp, fn, tp)
```

Out[336]:

```
(1108, 293, 208, 381)
```

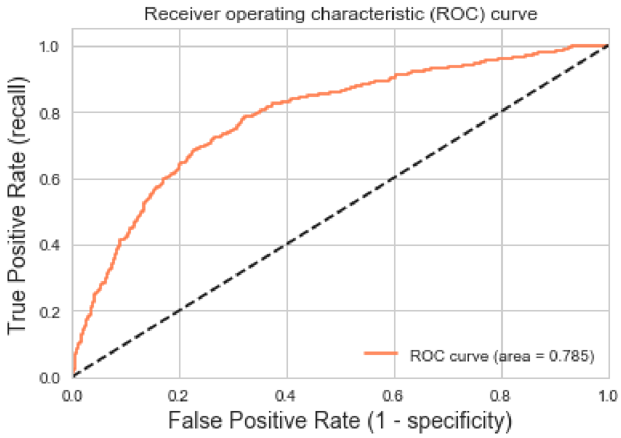
In [337]:

```
print("TPR is %2.3f" % (tp/(tp+fn)))
print("FPR is %2.3f" % (fp/(fp+tn)))
```

```
TPR is 0.647
FPR is 0.209
```

In [338]:

```
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 [486]:

```
#adding predicted probability results to test data
df_test_result=df_test_model
```

In [487]:

```
df_test_result = df_test_result.assign(pred_prob=y_test_pred_proba.values)
```

In [488]:

```
df_test_rank_order = df_test_result[['target_var', 'pred_prob']]
df_test_rank_bureau_score = df_test_result[['target_var', 'VAR_ca_score']]
```

In [546]:

```
df_test_rank_order['decile'] = pd.qcut(df_test_rank_order['pred_prob'], 10, labels=np.arange(10, 0, 
-1))
df_test_rank_bureau_score['decile'] = pd.qcut(df_test_rank_bureau_score['VAR_ca_score'], 10, 
labels=False)+1
```

```
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
```

```
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
```

In [548]:

```
df_test_rank_order['decile']=df_test_rank_order['decile'].astype(int)
df_test_rank_bureau_score['decile']=df_test_rank_bureau_score['decile'].astype(int)
```

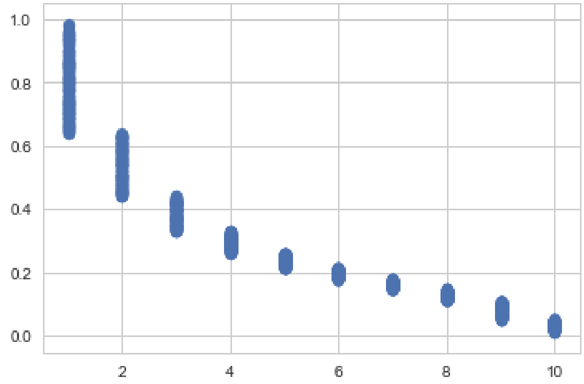
```
C:\Users\Tekkredi\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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
```

In [530]:

```
plt.scatter(df_test_rank_order['decile'], df_test_rank_order['pred_prob'])
```

Out[530]:

```
<matplotlib.collections.PathCollection at 0x17fae33e390>
```

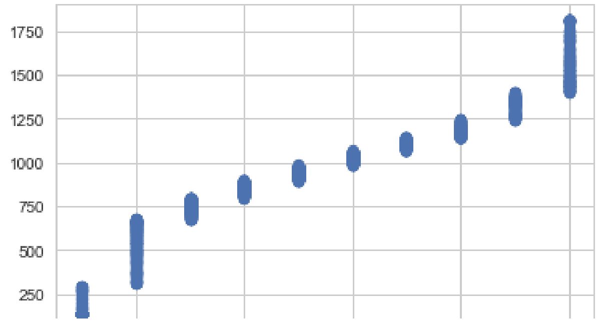


In [549]:

```
plt.scatter(df_test_rank_bureau_score['decile'], df_test_rank_bureau_score['VAR_ca_score'])
```

Out[549]:

```
<matplotlib.collections.PathCollection at 0x17fae45b1d0>
```



In [506]:

```
df_test_rank_order.groupby(['decile','target_var']).count()
```

Out[506]:

		pred_prob
decile	**target_var**	
10	**0**	185
	1	14
9	**0**	178
	1	21
8	**0**	182
	1	17
7	**0**	168
	1	31
6	**0**	172
	1	27
5	**0**	144
	1	55
4	**0**	127
	1	72
3	**0**	99
	1	100
2	**0**	88
	1	111
1	**0**	58
	1	141

In [550]:

```
badrate_vs_decile = pd.pivot_table(df_test_rank_order, values='pred_prob', index=['decile'],
                      columns=['target_var'], aggfunc='count')
badrate_vs_decile_2 = pd.pivot_table(df_test_rank_bureau_score, values='VAR_ca_score', index=['deci
le'],
                      columns=['target_var'], aggfunc='count')
```

In [551]:

```
test_rank_order = pd.DataFrame(badrate_vs_decile.to_records())
test_rank_order_2 = pd.DataFrame(badrate_vs_decile_2.to_records())
```

In [552]:

```
test_rank_order['bad_rate']=test_rank_order['1']/(test_rank_order['0']+test_rank_order['1'])
test_rank_order_2['bad_rate']=test_rank_order_2['1']/(test_rank_order_2['0']+test_rank_order_2['1'
])
```

In [557]:

```
test_rank_order_merge=test_rank_order.merge(test_rank_order_2, left_on='decile', right_on='decile'
, how='left')
```

In [558]:

```
test_rank_order_merge
```

Out[558]:

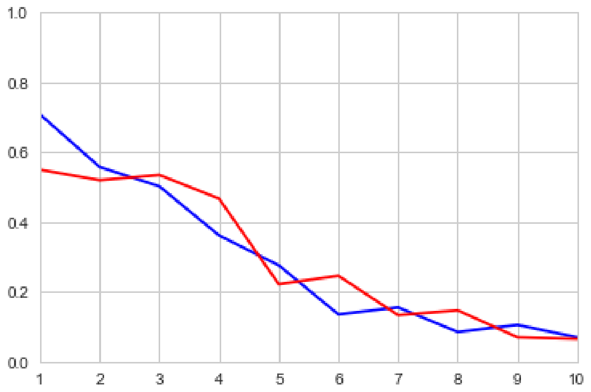
	decile	0_x	1_x	bad_rate_x	0_y	1_y	bad_rate_y
0	1	58	141	0.708543	90	110	0.550000
1	2	88	111	0.557789	95	103	0.520202
2	3	99	100	0.502513	93	107	0.535000
3	4	127	72	0.361809	106	93	0.467337
4	5	144	55	0.276382	154	44	0.222222
5	6	172	27	0.135678	150	49	0.246231
6	7	168	31	0.155779	175	27	0.133663
7	8	182	17	0.085427	168	29	0.147208
8	9	178	21	0.105528	185	14	0.070352
9	10	185	14	0.070352	185	13	0.065657

In [560]:

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

Out[560]:

```
([<matplotlib.axis.XTick at 0x17fae5134a8>,
  <matplotlib.axis.XTick at 0x17fae4f6f28>,
  <matplotlib.axis.XTick at 0x17fae527588>,
  <matplotlib.axis.XTick at 0x17fae554780>,
  <matplotlib.axis.XTick at 0x17fae554eb8>,
  <matplotlib.axis.XTick at 0x17fae55c668>,
  <matplotlib.axis.XTick at 0x17fae55cdd8>,
  <matplotlib.axis.XTick at 0x17fae560588>,
  <matplotlib.axis.XTick at 0x17fae560cf8>,
  <matplotlib.axis.XTick at 0x17fae5654a8>],
 <a list of 10 Text xticklabel objects>)
```


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