In [458]:

```
import pyodbc
```

In [459]:

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

In [460]:

```
import pandas as pd
```

In [462]:

```
df = pd.read_sql("select * from dbo.IndusTrain_Model", conn)
```

In [463]:

```
df.head()
```

Out[463]:

	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 [314]:

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

In [30]:

```
list(df)
```

Out[30]:

```
['index',
 'ca_customertransactionid',
 'c_gender',
 'ca_avgmonthlycanbepaid',
 'ca_avgpayrollincome',
 'ca_maxmonthlycanbepaid',
 'ca_maxpayrollincome',
 'ca_minmonthlycanbepaid',
 'ca_minpayrollincome',
 'ca_occupation',
 'ca_preferbank1',
 'ca_preferbank2',
 'VAR_ca_score',
 'ca_totalamount',
 'cs_education',
 'cs_homeowner',
```

```
 'cs_workcity',
 'cs_workperiod',
 'cs_worksector',
 'cs_worktitle',
 'VAR_ctb_average_months_on_time_x_creditcard_loan_last_3months',
 'ctb_avg_months_60day_delinquent_x_open_loan_x_personal_loan_last_6months',
 'ctb_avg_months_90day_delinquent_last_18months',
 'ctb_avg_months_90day_delinquent_x_overdraft_acct_last_6months',
 'avg_ratio_totaldebt_to_creditlimit_last_3months',
 'ratio_avg_months_30day_delinquent_last_6months_to_avg_months_30day_delinquent_last_18months',

'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_var']
```

In [27]:

```
df.describe()
```

Out[27]:

	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

In [76]:

```
df['cs_workperiod'].unique()
```

Out[76]:

```
array(['1-Mar', 'Unknown', '10+', '5-Oct', '3-May', '0 - 1'], dtype=object)
```

In [74]:

```
df['ca_occupation'].value_counts()
```

Out[74]:

```
?zel Sekt?r ?cretli                      1977
Kamu ?cretli                              601
Serbest Meslek                            141
Emekli                                    132
Diger                                      98
?alismiyor                                 17
"Profesyonel (Doktor, Eczaci, Avukat)"      7
?grenci                                     7
Ev Hanimi                                   3
Name: ca_occupation, dtype: int64
```

In [59]:

```
df.groupby('target_var').mean()
```

Out[59]:

	ca_avgmonthlycanbepaid	ca_avgpayrollincome	ca_maxmonthlycanbepaid	ca_maxpayrollincome	ca_minmont
target_var					
0	1342.658199	2581.677598	1456.745958	2726.966282	1228.550115
1	1270.365526	2370.205379	1391.698044	2511.953545	1148.984108

In [250]:

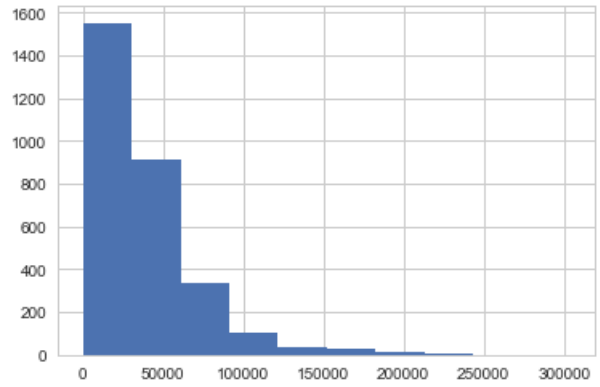
```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="white") #white background style for seaborn plots
sns.set(style="whitegrid", color_codes=True)
```

In [251]:

```
df.ca_totalamount.hist()
```

Out[251]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x26c8e163940>
```



In [253]:

```
%matplotlib inline
```

In [72]:

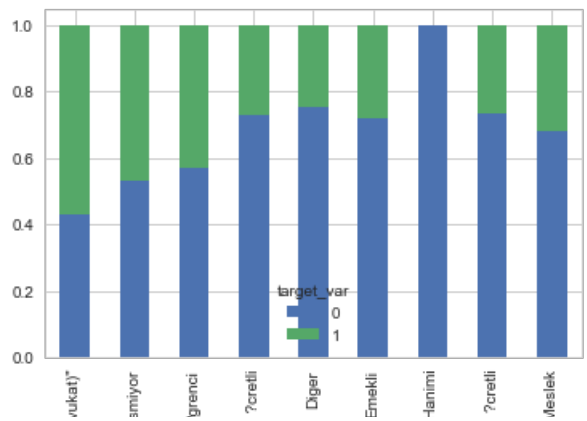
```
table=pd.crosstab(df.ca_occupation,df.target_var)
```

In [421]:

```
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
```

Out[421]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x26c8eb7ebe0>
```



In [316]:

```
cat_vars=df[['c_gender', 'ca_occupation', 'cs_education', 'cs_homeowner',
'cs_workperiod', 'cs_worktitle']]
cat_list = pd.get_dummies(cat_vars, drop_first=True)
df_edit=df.join(cat_list)
```

In [317]:

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

In [318]:

```
vars_to_drop=df[['c_gender', 'ca_occupation', 'cs_education', 'cs_homeowner',
                 'cs_workperiod', 'cs_worksector',
'cs_worktitle','index','ca_customertransactionid',
                 'ca_preferbank1','ca_preferbank2','cs_workcity']]
```

In [319]:

```
df_edit=df_edit[[i for i in df_vars if i not in vars_to_drop]]
```

In [320]:

```
df_vars2=df_edit.columns.values.tolist()
```

In [321]:

```
y=df_edit[['target_var']]
x=df_edit[[i for i in df_vars2 if i not in y]]
```

In [109]:

```
from sklearn import datasets
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
```

In [256]:

```
logreg = LogisticRegression()
rfe = RFE(logreg, 11)
rfe = rfe.fit(x, y.values.ravel())
print(list(x.columns[rfe.support_]))
```

```
['VAR_ctb_average_months_on_time_x_creditcard_loan_last_3months',
'ctb_avg_months_90day_delinquent_last_18months',
'ctb_avg_months_90day_delinquent_x_overdraft_acct_last_6months',
'avg_ratio_totaldebt_to_creditlimit_last_3months',
'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', 'cs_education_Y?ksek Lisans / Doktor
a', 'cs_homeowner_Diger', 'cs_worktitle_Asistan', 'cs_worktitle_Direkt?r', 'cs_worktitle_Sirket sa
hibi / ortagi']
```

In [322]:

```
from sklearn.feature_selection import RFECV
# Create the RFE object and compute a cross-validated score.
# The "accuracy" scoring is proportional to the number of correct classifications
```

```
rfecv = RFECV(estimator=LogisticRegression(), step=1, cv=10, scoring='accuracy')
rfecv.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[322]:

```
RFECV(cv=10,
   estimator=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),
   n_jobs=1, scoring='accuracy', step=1, verbose=0)
```

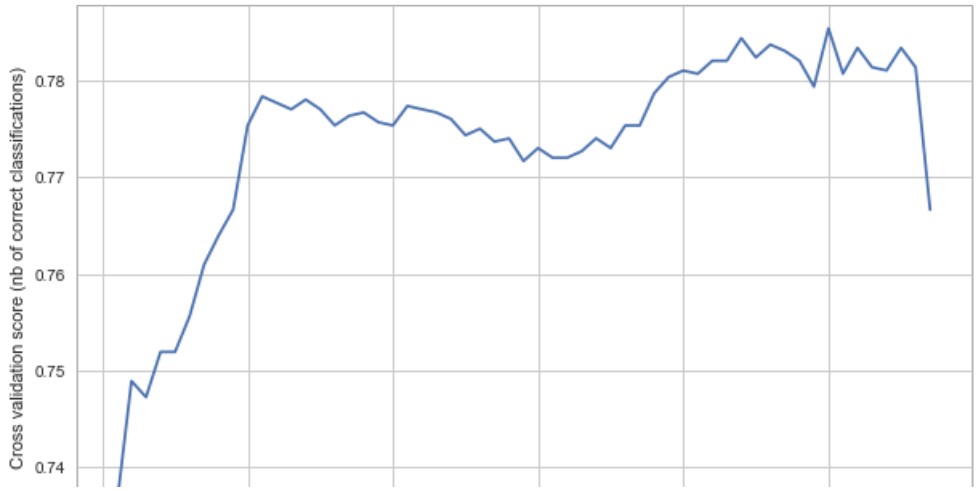
In [248]:

```
print("Optimal number of features: %d" % rfecv.n_features_)
print('Selected features: %s' % list(x.columns[rfecv.support_]))
```

```
Optimal number of features: 50
Selected features: ['ca_avgmonthlycanbepaid', 'ca_maxmonthlycanbepaid', 'ca_minmonthlycanbepaid',
'VAR_ca_score', 'VAR_ctb_average_months_on_time_x_creditcard_loan_last_3months',
'ctb_avg_months_60day_delinquent_x_open_loan_x_personal_loan_last_6months',
'ctb_avg_months_90day_delinquent_last_18months',
'ctb_avg_months_90day_delinquent_x_overdraft_acct_last_6months',
'avg_ratio_totaldebt_to_creditlimit_last_3months',
'ratio_avg_months_30day_delinquent_last_6months_to_avg_months_30day_delinquent_last_18months', 'VA
R_Inverse_of_ratio_avg_months_30day_delinquent_last_6months_to_avg_months_30day_delinquent_last_18m
s', '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', 'c_gender_Male', '
ca_occupation_?alismiyor', 'ca_occupation_?grenci', 'ca_occupation_?zel Sekt?r ?cretli',
'ca_occupation_Emekli', 'ca_occupation_Ev Hanimi', 'ca_occupation_Serbest Meslek',
'cs_education_Diger', 'cs_education_Ilkokul', 'cs_education_Lise', 'cs_education_Ortaokul',
'cs_education_Y?ksek Lisans / Doktora', 'cs_education_Y?ksekokul', 'cs_homeowner_Baska birinin yan
inda yasiyorum', 'cs_homeowner_Diger', 'cs_homeowner_Kendimin', 'cs_homeowner_Kira',
'cs_homeowner_Lojman', 'cs_workperiod_1-Mar', 'cs_workperiod_10+', 'cs_workperiod_3-May',
'cs_workperiod_5-Oct', 'cs_workperiod_Unknown', 'cs_worktitle_Asistan', 'cs_worktitle_Diger', 'cs_
worktitle_Direkt?r', 'cs_worktitle_Genel M?d?r', 'cs_worktitle_M?d?r', 'cs_worktitle_Sirket sahibi
/ ortagi', 'cs_worktitle_Sorumlu', 'cs_worktitle_Stajyer', 'cs_worktitle_Tekniker / Operat?r',
'cs_worktitle_Unknown', 'cs_worktitle_Uzman', 'cs_worktitle_Y?netmen / Y?netici']
```

In [255]:

```
# Plot number of features VS. cross-validation scores
plt.figure(figsize=(10,6))
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.plot(range(1, len(rfecv.grid_scores_) + 1), rfecv.grid_scores_)
plt.show()
```



In [323]:

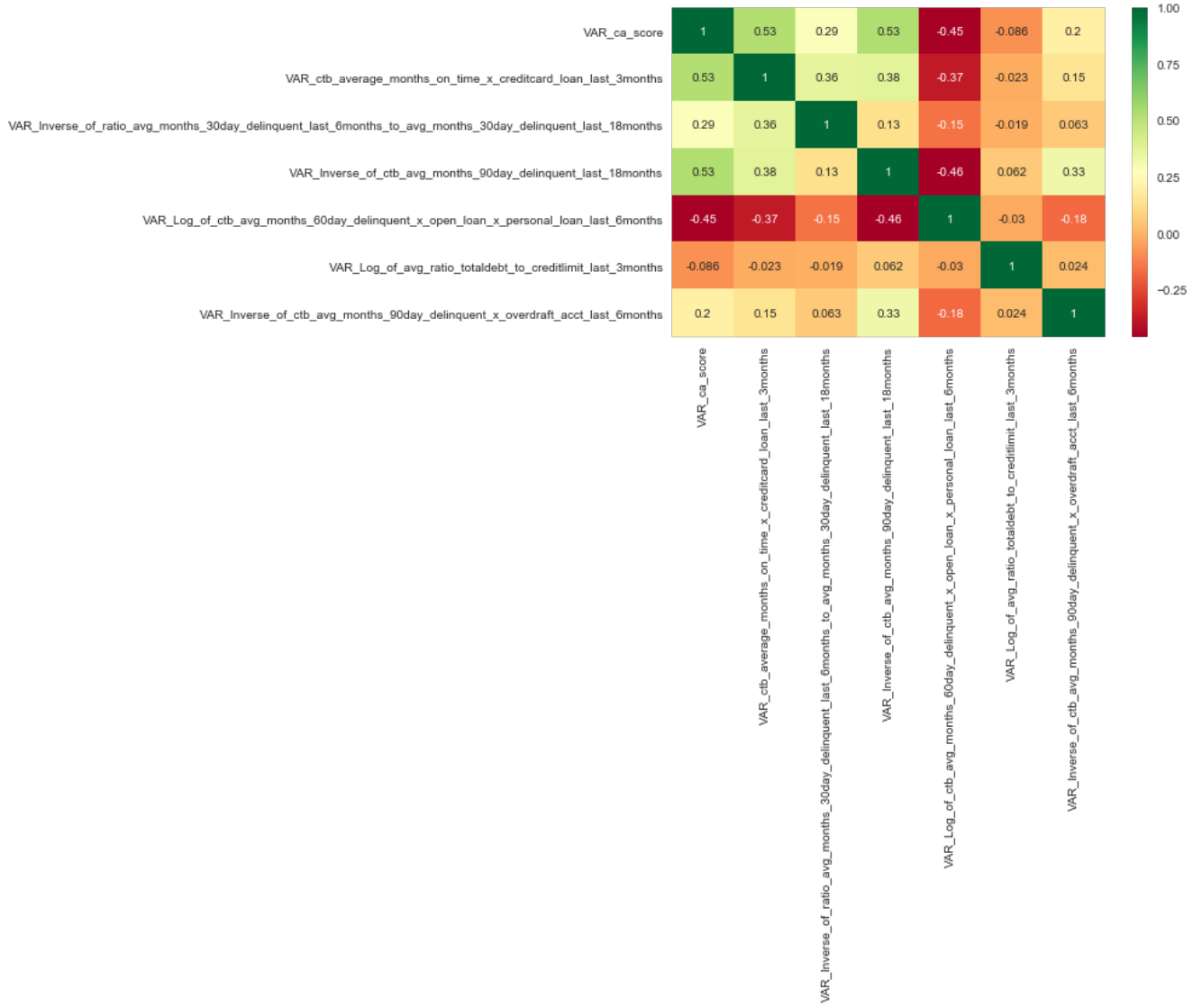
```
df_model=df_edit[['VAR_ca_score', 'VAR_ctb_average_months_on_time_x_creditcard_loan_last_3months',

'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 [324]:

```
Y=df_model[['target_var']]
X=df_model[[i for i in df_model.columns.values.tolist() if i not in y]]
```

In [325]:

```
plt.subplots(figsize=(8, 5))
sns.heatmap(X.corr(), annot=True, cmap="RdYlGn")
plt.show()
```



In [260]:

```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, classification_report, precision_score, recall_score
```

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

In [326]:

```
# use train/test split with different random_state values
# we can change the random_state values that changes the accuracy scores
# the scores change a lot, this is why testing scores is a high-variance estimate
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

In [ ]:

```
# make cross validation
```

In [475]:

```
# check classification scores of logistic regression
logreg = LogisticRegression()
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[475]:

```
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 [ ]:

```
y_pred = logreg.predict(X_test)
```

In [476]:

```
logreg.coef_
```

Out[476]:

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

In [477]:

```
y_pred_proba = logreg.predict_proba(X)[:, 1]
```

In [453]:

```
[fpr, tpr, thr] = roc_curve(Y_test, y_pred_proba)
```

In [454]:

```
print('Train/Test split results:')
print(logreg.__class__.__name__+" accuracy is %2.3f" % accuracy_score(Y_test, y_pred))
print(logreg.__class__.__name__+" log_loss is %2.3f" % log_loss(Y_test, y_pred_proba))
print(logreg.__class__.__name__+" auc is %2.3f" % auc(fpr, tpr))
```

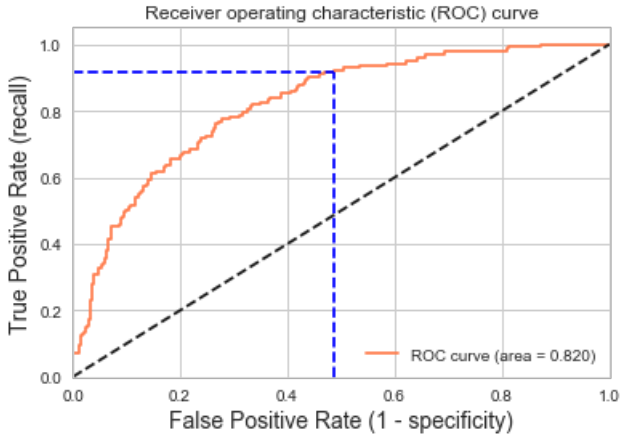
```
Train/Test split results:
LogisticRegression accuracy is 0.796
LogisticRegression log_loss is 0.444
LogisticRegression auc is 0.820
```

In [350]:

```
idx = np.min(np.where(tpr > 0.95)) # index of the first threshold for which the sensibility > 0.95
```

In [455]:

```
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.plot([0,fpr[idx]], [tpr[idx],tpr[idx]], 'k--', color='blue')
plt.plot([fpr[idx],fpr[idx]], [0,tpr[idx]], 'k--', color='blue')
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 [456]:

```
2*0.82-1
```

Out[456]:

```
0.6399999999999999
```

In [478]:

```
import statsmodels.api as sm
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
```

In [443]:

```
X_sm=sm.add_constant(X, prepend=False)
```

In [444]:

```
model=sm.Logit(Y,X_sm)
```

In [445]:

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

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

In [446]:

```
result.summary()
```

Out[446]:

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:	12:04:34	**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