CreditOne Regression

July 17, 2018

In [1]:
```
#Import required modules
import numpy as np
import pandas as pd
import scipy
from math import sqrt
import matplotlib.pyplot as plt
```

In [3]:
```
#import estimators
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn import linear_model
```

In [4]:
```
#import model metrics
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_score
```

In [5]:
```
#import Cross Validation
from sklearn.cross_validation import train_test_split
```

```
C:\Users\Voleti\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning
  "This module will be removed in 0.20.", DeprecationWarning)
```

In [10]:
```
#Import the the raw data from CSV file
credit = pd.read_csv('D:/Python Data Science UTA/default of credit card clients.csv',
credit.head()
```

Out[10]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	\
0	1	20000	2	2	1	24	2	2	-1	-1	
1	2	120000	2	2	2	26	-1	2	0	0	
2	3	90000	2	2	2	34	0	0	0	0	
3	4	50000	2	2	1	37	0	0	0	0	
4	5	50000	1	2	1	57	-1	0	-1	0	

	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	\
0	...	0	0	0	0	

```
1             ...                  3272       3455       3261          0
2             ...                 14331      14948      15549       1518
3             ...                 28314      28959      29547       2000
4             ...                 20940      19146      19131       2000

   PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5  PAY_AMT6  \
0       689         0         0         0         0
1      1000      1000      1000         0      2000
2      1500      1000      1000      1000      5000
3      2019      1200      1100      1069      1000
4     36681     10000      9000       689       679

   default payment next month
0                           1
1                           1
2                           0
3                           0
4                           0

[5 rows x 25 columns]
```

In [11]:
```
#Examine the imported raw data from cerdit default file.
credit.info()
credit.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
ID                          30000 non-null int64
LIMIT_BAL                   30000 non-null int64
SEX                         30000 non-null int64
EDUCATION                   30000 non-null int64
MARRIAGE                    30000 non-null int64
AGE                         30000 non-null int64
PAY_0                       30000 non-null int64
PAY_2                       30000 non-null int64
PAY_3                       30000 non-null int64
PAY_4                       30000 non-null int64
PAY_5                       30000 non-null int64
PAY_6                       30000 non-null int64
BILL_AMT1                   30000 non-null int64
BILL_AMT2                   30000 non-null int64
BILL_AMT3                   30000 non-null int64
BILL_AMT4                   30000 non-null int64
BILL_AMT5                   30000 non-null int64
BILL_AMT6                   30000 non-null int64
PAY_AMT1                    30000 non-null int64
PAY_AMT2                    30000 non-null int64
```

```
PAY_AMT3                      30000 non-null int64
PAY_AMT4                      30000 non-null int64
PAY_AMT5                      30000 non-null int64
PAY_AMT6                      30000 non-null int64
default payment next month    30000 non-null int64
dtypes: int64(25)
memory usage: 5.7 MB
```

Out[11]:

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	\
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	
mean	15000.500000	167484.322667	1.603733	1.853133	1.551867	
std	8660.398374	129747.661567	0.489129	0.790349	0.521970	
min	1.000000	10000.000000	1.000000	0.000000	0.000000	
25%	7500.750000	50000.000000	1.000000	1.000000	1.000000	
50%	15000.500000	140000.000000	2.000000	2.000000	2.000000	
75%	22500.250000	240000.000000	2.000000	2.000000	2.000000	
max	30000.000000	1000000.000000	2.000000	6.000000	3.000000	

	AGE	PAY_0	PAY_2	PAY_3	PAY_4	\
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	
mean	35.485500	-0.016700	-0.133767	-0.166200	-0.220667	
std	9.217904	1.123802	1.197186	1.196868	1.169139	
min	21.000000	-2.000000	-2.000000	-2.000000	-2.000000	
25%	28.000000	-1.000000	-1.000000	-1.000000	-1.000000	
50%	34.000000	0.000000	0.000000	0.000000	0.000000	
75%	41.000000	0.000000	0.000000	0.000000	0.000000	
max	79.000000	8.000000	8.000000	8.000000	8.000000	

	...	BILL_AMT4	BILL_AMT5	\
count	...	30000.000000	30000.000000	
mean	...	43262.948967	40311.400967	
std	...	64332.856134	60797.155770	
min	...	-170000.000000	-81334.000000	
25%	...	2326.750000	1763.000000	
50%	...	19052.000000	18104.500000	
75%	...	54506.000000	50190.500000	
max	...	891586.000000	927171.000000	

	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_AMT3	\
count	30000.000000	30000.000000	3.000000e+04	30000.00000	
mean	38871.760400	5663.580500	5.921163e+03	5225.68150	
std	59554.107537	16563.280354	2.304087e+04	17606.96147	
min	-339603.000000	0.000000	0.000000e+00	0.00000	
25%	1256.000000	1000.000000	8.330000e+02	390.00000	
50%	17071.000000	2100.000000	2.009000e+03	1800.00000	
75%	49198.250000	5006.000000	5.000000e+03	4505.00000	
max	961664.000000	873552.000000	1.684259e+06	896040.00000	

```
              PAY_AMT4       PAY_AMT5       PAY_AMT6  default payment next month
count    30000.000000   30000.000000   30000.000000                30000.000000
mean      4826.076867    4799.387633    5215.502567                    0.221200
std      15666.159744   15278.305679   17777.465775                    0.415062
min          0.000000       0.000000       0.000000                    0.000000
25%        296.000000     252.500000     117.750000                    0.000000
50%       1500.000000    1500.000000    1500.000000                    0.000000
75%       4013.250000    4031.500000    4000.000000                    0.000000
max     621000.000000  426529.000000  528666.000000                    1.000000

[8 rows x 25 columns]
```

In [12]:
```
#Raw data loaded from the cedit default file is  split into features and dependent va
#Feature selection
features = credit.iloc[:,12:23]
print('Summary of feature sample')
features.head()
```

```
Summary of feature sample
```

Out[12]:

	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	\
0	3913	3102	689	0	0	0	0	
1	2682	1725	2682	3272	3455	3261	0	
2	29239	14027	13559	14331	14948	15549	1518	
3	46990	48233	49291	28314	28959	29547	2000	
4	8617	5670	35835	20940	19146	19131	2000	

	PAY_AMT2	PAY_AMT3	PAY_AMT4	PAY_AMT5
0	689	0	0	0
1	1000	1000	1000	0
2	1500	1000	1000	1000
3	2019	1200	1100	1069
4	36681	10000	9000	689

In [14]:
```
#Dependent variable selection
depVar = credit['PAY_AMT6']
```

In [15]:
```
#Establish the training set for the X-variables or Feature space (first 1000 rows: on
#Training Set (Feature Space: X Training)
X_train = (features[: 1000])
X_train.head()
```

Out[15]:

	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	\
0	3913	3102	689	0	0	0	0	
1	2682	1725	2682	3272	3455	3261	0	
2	29239	14027	13559	14331	14948	15549	1518	
3	46990	48233	49291	28314	28959	29547	2000	

```
4       8617       5670       35835       20940       19146       19131       2000

   PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5
0       689         0         0         0
1      1000      1000      1000         0
2      1500      1000      1000      1000
3      2019      1200      1100      1069
4     36681     10000      9000       689
```

In [16]:
```
#Establish the training set for the Y-variable or dependent variable (the number of r
#Dependent Variable Training Set (y Training)
y_train = depVar[: 1000]
y_train_count = len(y_train.index)
print('The number of observations in the Y training set are:',str(y_train_count))
y_train.head()
```

```
The number of observations in the Y training set are: 1000
```

Out[16]:
```
0       0
1    2000
2    5000
3    1000
4     679
Name: PAY_AMT6, dtype: int64
```

In [17]:
```
#Establish the testing set for the X-Variables or Feature space
#Testing Set (X Testing)
X_test = features[-100:]
X_test_count = len(X_test.index)
print('The number of observations in the feature testing set is:',str(X_test_count))
print(X_test.head())
```

```
The number of observations in the feature testing set is: 100
       BILL_AMT1  BILL_AMT2  BILL_AMT3  BILL_AMT4  BILL_AMT5  BILL_AMT6  \
29900      16809          0          0          0          0          0
29901      50845      48750     103486      50590      50248      49387
29902      10392     168088     168955     161351     126198     124746
29903      27378      17082      13333         99         99     172104
29904      54952      56021      54126      58732      59306      59728

       PAY_AMT1  PAY_AMT2  PAY_AMT3  PAY_AMT4  PAY_AMT5
29900         0         0         0         0         0
29901         0      6556      3250      1563      1208
29902    168096      6409      7335      4448      4519
29903     10018     13333        99        99    172104
29904      2600      4553      5800      2000      1000
```

```
In [18]: #Establish Ground truth
         #Ground Truth (y_test)
         y_test = depVar[:-100]
         y_test_count = len(y_test.index)
         print('The number of observations in the Y training set are:',str(y_test_count))
         y_test.head()
```

```
The number of observations in the Y training set are: 29900
```

```
Out[18]: 0       0
         1    2000
         2    5000
         3    1000
         4     679
         Name: PAY_AMT6, dtype: int64
```

```
In [19]: #implement Cross Validation by running the following on the X and Y training sets:
         X_train, X_test, y_train, y_test = train_test_split(X_train, y_train)
```

```
In [20]: #use the shape function to check that the split was made as needed:
         X_train.shape, X_test.shape
```

```
Out[20]: ((750, 11), (250, 11))
```

```
In [21]: #Established three different models with the individual variable names
         #Models
         modelSVR = SVR()
         modelRF = RandomForestRegressor()
         modelLR = LinearRegression()
```

```
In [27]: #Fit above three models to our train datasets.
         #Support Vector Regression
         modelSVR.fit(X_train,y_train)
         print(cross_val_score(modelSVR, X_train, y_train))
         modelSVR.score(X_train,y_train)
```

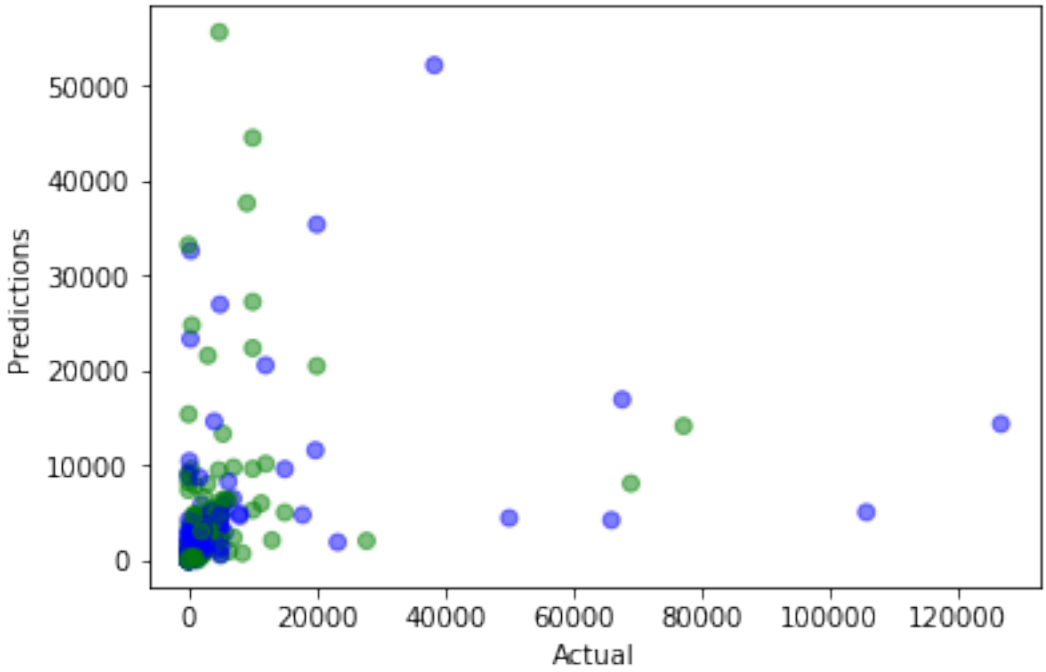
```
[-0.03407411 -0.10966667 -0.01289266]
```

```
Out[27]: -0.023180006062174963
```

```
In [25]: #Random Forest
         modelRF.fit(X_train,y_train)
         print(cross_val_score(modelRF, X_train, y_train))
         modelRF.score(X_train,y_train)
```

```
[ 0.02281547 -0.66664745  0.03746356]
```

```
Out[25]: 0.75962578110099455

In [26]: #Linear Regression
         modelLR.fit(X_train,y_train)
         print(cross_val_score(modelLR, X_train, y_train))
         modelLR.score(X_train,y_train)

[ 0.24578353 -2.14695377  0.40398575]


Out[26]: 0.59133781731919344

In [28]: ##From the model score random forest is the best model out of three.
         #Making Predictions using RF
         predictions = modelRF.predict(X_test)

In [32]: #calculating RMSE
         rmse = sqrt(mean_squared_error(y_test, predictions))
         print('RMSE: %.3f' % rmse)

RMSE: 14395.439


In [33]: #Calculate Rquared value using r2_score function, y_test and predections from previou
         predRsquared = r2_score(y_test,predictions)
         print('R Squared: %.3f' % predRsquared)

R Squared: -0.026


In [36]: #Plotting the Results, create scatterplot using matplotlib pyplot.
         plt.scatter(y_test, predictions, color=['blue','green'], alpha = 0.5)
         plt.xlabel('Actual')
         plt.ylabel('Predictions')
         plt.show();
```


Predictions
50000
40000
30000
20000
10000
0
0
20000
40000
60000
80000
100000
120000
Actual