Task:

Issuing Bank is a significant role in credit card transaction. It functions as a bridge between cardholder, Acquirer Bank, and Payment Network. The report follows the order of data cleansing, data analyzing, modeling and decision making. The prime goal of this report is to introduce a credit score mechanism to reduce Credit default risk and increase the profit of the Issuing Banking. In order to make a better decision for card issuers, the key point is predicting the probability of credit card default precisely.

Data Preparation:

In the data cleansing process, there are 17 duplicate data removed and 698 ‘corrupted’ data removed in total. Values out of range or blank are removed. Outliers are not considered.

Data Analysis

We choose the Chi-square test to test different variables against monthly payment status. We used Pearson Correlation to see relationship between monthly payment status and other month payment status.

		Total	Percentage
Gender	Male	11898	39.8%
	Female	17989	60.2%
Education	Master's degree & Doctoral Degree	10653	35.6%
	Bachelor’s Degree	14099	47.2%
	High School’s Degree	4960	16.6%
	Others	175	0.6%
Marital_Status	Married	13570	45.4%
	Single	15915	53.3%
	Others	402	1.3%
Jan_Repay_Status	Paid on time	22935	76.7%
	Payment Delayed	6952	23.3%
Feb_Repay_Status	Paid on time	25283	84.6%
	Payment Delayed	4604	15.4%
Mar_Repay_Status	Paid on time	25516	85.4%
	Payment Delayed	4371	14.6%
Apr_Repay_Status	Paid on time	26201	87.7%
	Payment Delayed	3686	12.3%
May_Repay_Status	Paid on time	26745	89.5%
	Payment Delayed	3142	10.5%
Jun_Repay_Status	Paid on time	26632	89.1%
	Payment Delayed	3255	10.9%

Comment

Female accounts for 60.2% of the total users, 1.5x of Male users.

Undergrads users are the majority, followed by Grads and Post-Grads

More Single than Married users.

Male users have a lower paid on time rate than Female users.

Users with higher education have a higher paid on time rate.

		Gender			
		Male		Female	
Jan_Repay_Status	Paid on time	8962	75.3%	13973	77.7%
	Payment Delayed	2936	24.7%	4016	22.3%
Feb_Repay_Status	Paid on time	9831	82.6%	15452	85.9%
	Payment Delayed	2067	17.4%	2537	14.1%
Mar_Repay_Status	Paid on time	9939	83.5%	15577	86.6%
	Payment Delayed	1959	16.5%	2412	13.4%
Apr_Repay_Status	Paid on time	10235	86.0%	15966	88.8%
	Payment Delayed	1663	14.0%	2023	11.2%
May_Repay_Status	Paid on time	10471	88.0%	16274	90.5%
	Payment Delayed	1427	12.0%	1715	9.5%
Jun_Repay_Status	Paid on time	10458	87.9%	16174	89.9%
	Payment Delayed	1440	12.1%	1815	10.1%
Max delayed rate			24.7%		22.3%
Min delayed rate			12.0%		9.5%
Average delayed rate			16.1%		13.5%

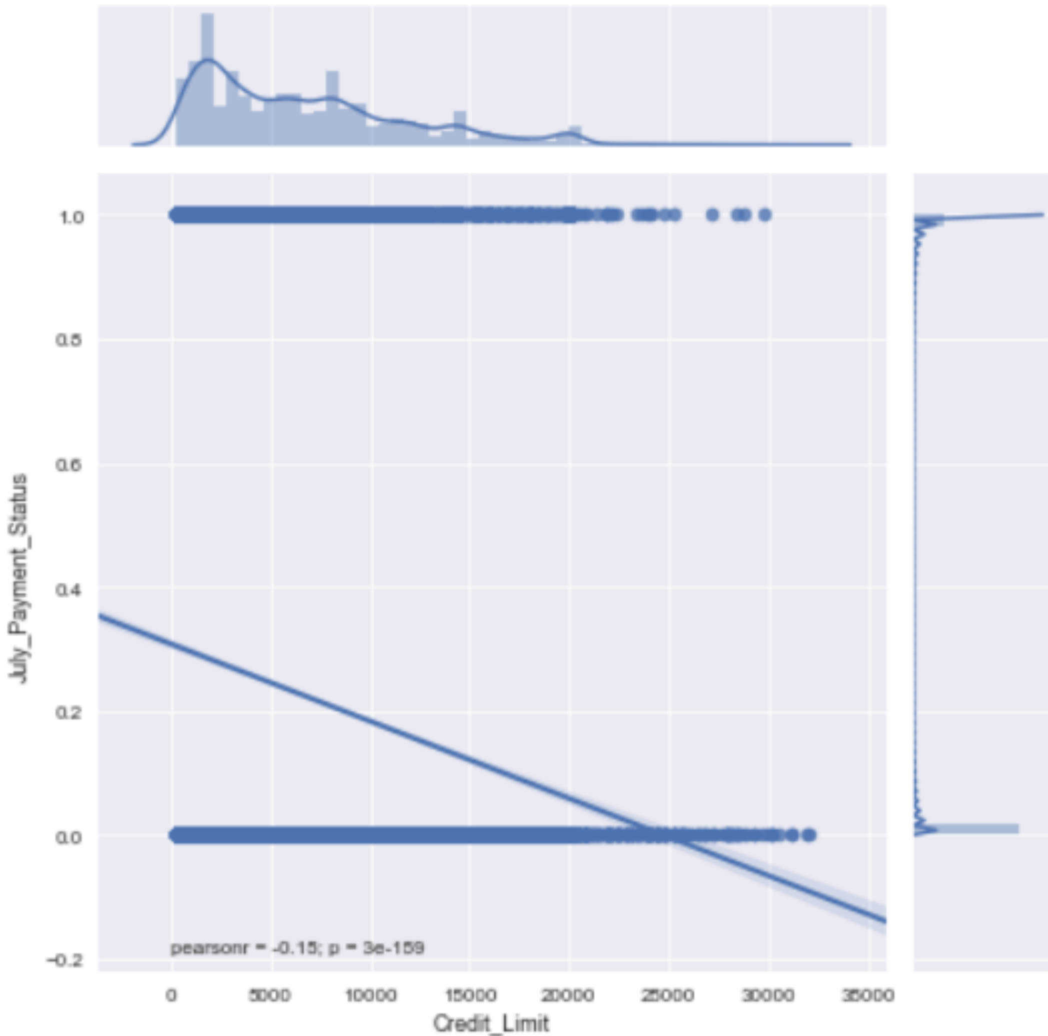
p	0.007965933

		Education							
		Master's degree & Doctoral Degree		Bachelor's Degree		High School's Degree		Others	
Jan_Repay_Status	Paid on time	8473	79.5%	10721	76.0%	3624	73.1%	117	66.9%
	Payment Delayed	2180	20.5%	3378	24.0%	1336	26.9%	58	33.1%
Feb_Repay_Status	Paid on time	9454	88.7%	11689	82.9%	4008	80.8%	132	75.4%
	Payment Delayed	1199	11.3%	2410	17.1%	952	19.2%	43	24.6%
Mar_Repay_Status	Paid on time	9480	89.0%	11850	84.0%	4055	81.8%	131	74.9%
	Payment Delayed	1173	11.0%	2249	16.0%	905	18.2%	44	25.1%
Apr_Repay_Status	Paid on time	9685	90.9%	12190	86.5%	4196	84.6%	130	74.3%
	Payment Delayed	968	9.1%	1909	13.5%	764	15.4%	45	25.7%
May_Repay_Status	Paid on time	9799	92.0%	12473	88.5%	4338	87.5%	135	77.1%
	Payment Delayed	854	8.0%	1626	11.5%	622	12.5%	40	22.9%
Jun_Repay_Status	Paid on time	9719	91.2%	12423	88.1%	4361	87.9%	129	73.7%
	Payment Delayed	934	8.8%	1676	11.9%	599	12.1%	46	26.3%
Max delayed rate			20.5%		24.0%		26.9%		33.1%
Min delayed rate			8.0%		11.5%		12.1%		22.9%
Average delayed rate			11.4%		15.7%		17.4%		26.3%

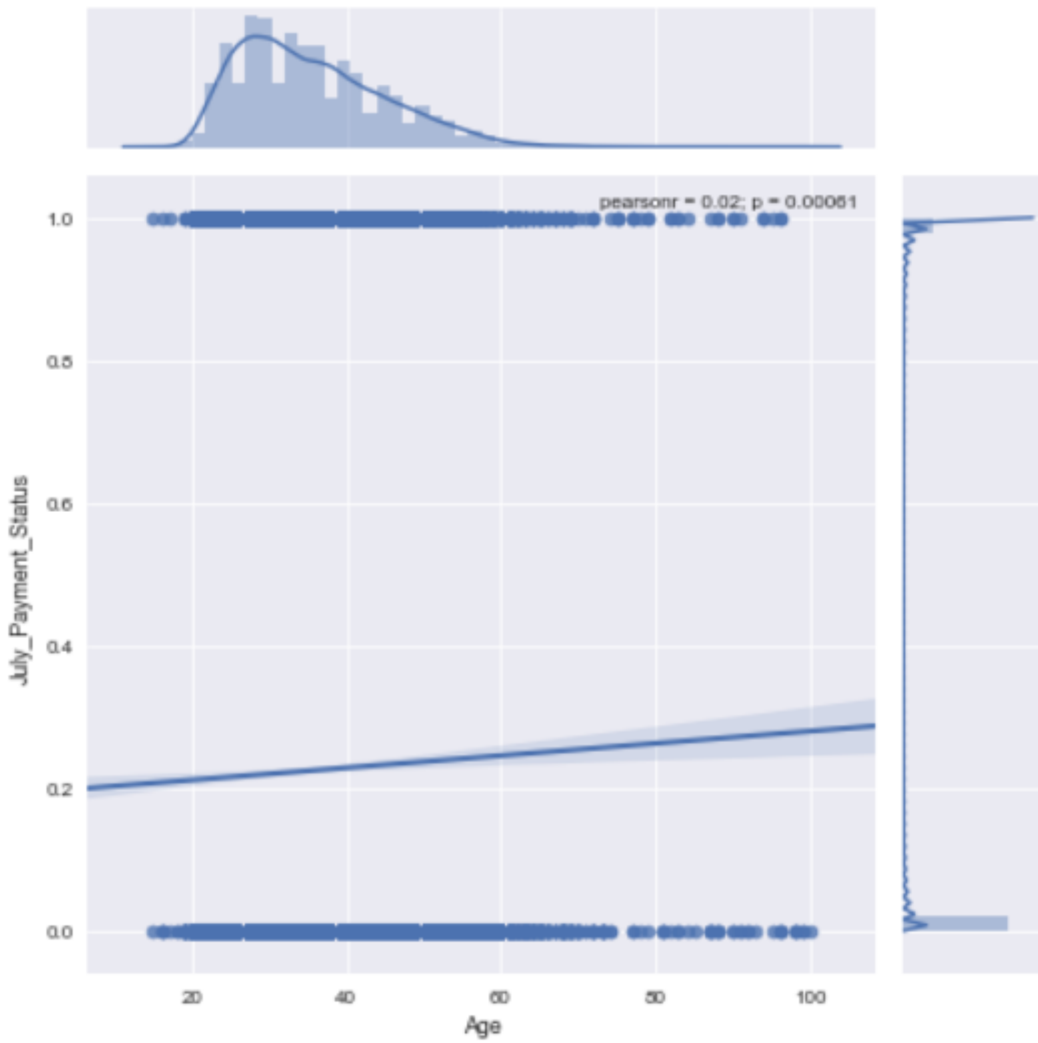
p	2.17298E-09

		Marital_Status					
		Married		Single		Others	
Jan_Repay_Status	Paid on time	10328	76.1%	12358	77.7%	249	61.9%
	Payment Delayed	3242	23.9%	3557	22.3%	153	38.1%
Feb_Repay_Status	Paid on time	11440	84.3%	13560	85.2%	283	70.4%
	Payment Delayed	2130	15.7%	2355	14.8%	119	29.6%
Mar_Repay_Status	Paid on time	11599	85.5%	13634	85.7%	283	70.4%
	Payment Delayed	1971	14.5%	2281	14.3%	119	29.6%
Apr_Repay_Status	Paid on time	11890	87.6%	14022	88.1%	289	71.9%
	Payment Delayed	1680	12.4%	1893	11.9%	113	28.1%
May_Repay_Status	Paid on time	12158	89.6%	14286	89.8%	301	74.9%
	Payment Delayed	1412	10.4%	1629	10.2%	101	25.1%
Jun_Repay_Status	Paid on time	12130	89.4%	14206	89.3%	296	73.6%
	Payment Delayed	1440	10.6%	1709	10.7%	106	26.4%
Max delayed rate			23.9%		22.3%		38.1%
Min delayed rate			10.4%		10.2%		25.1%
Average delayed rate			14.6%		14.1%		29.5%

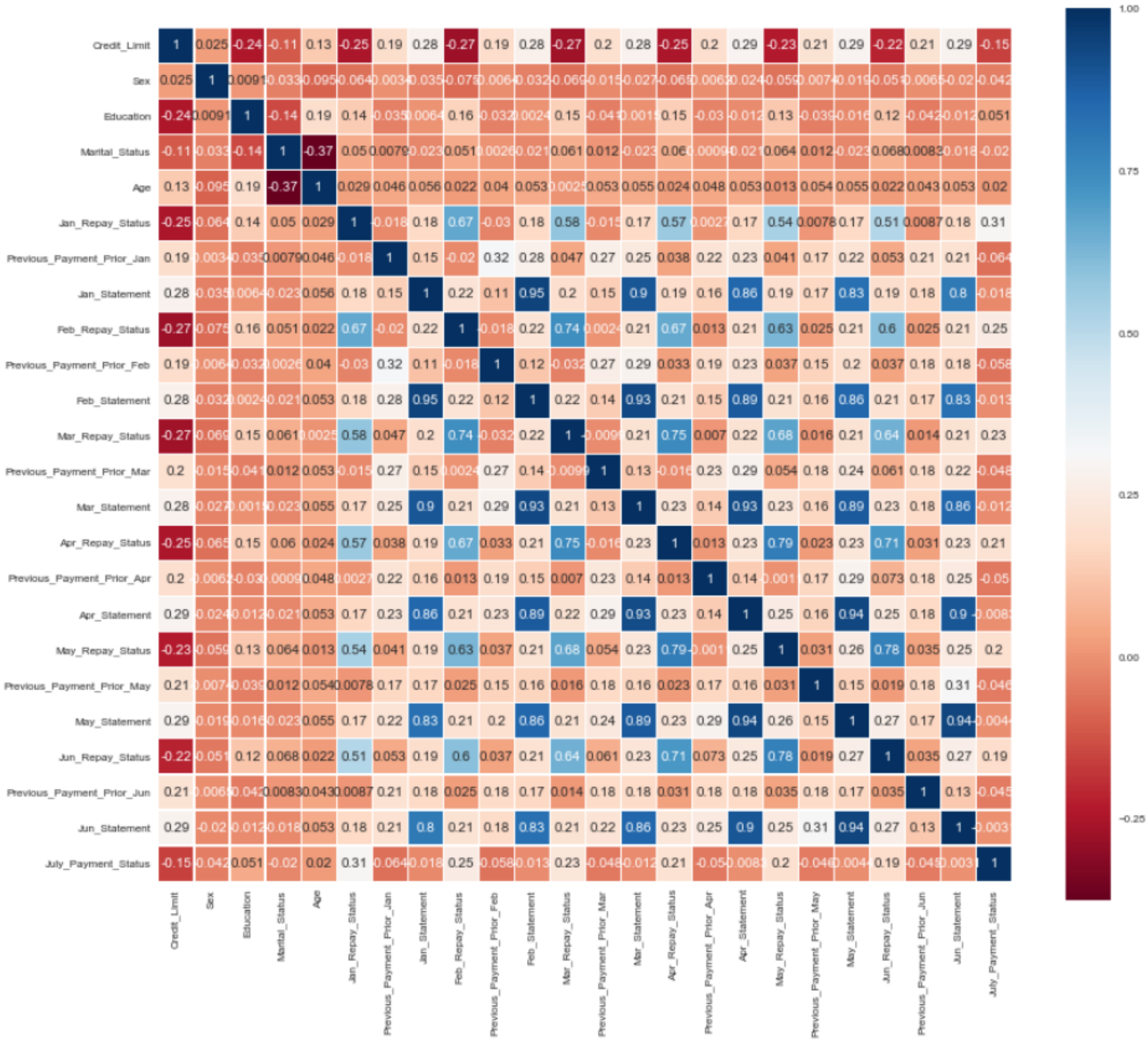
p 0.027100105

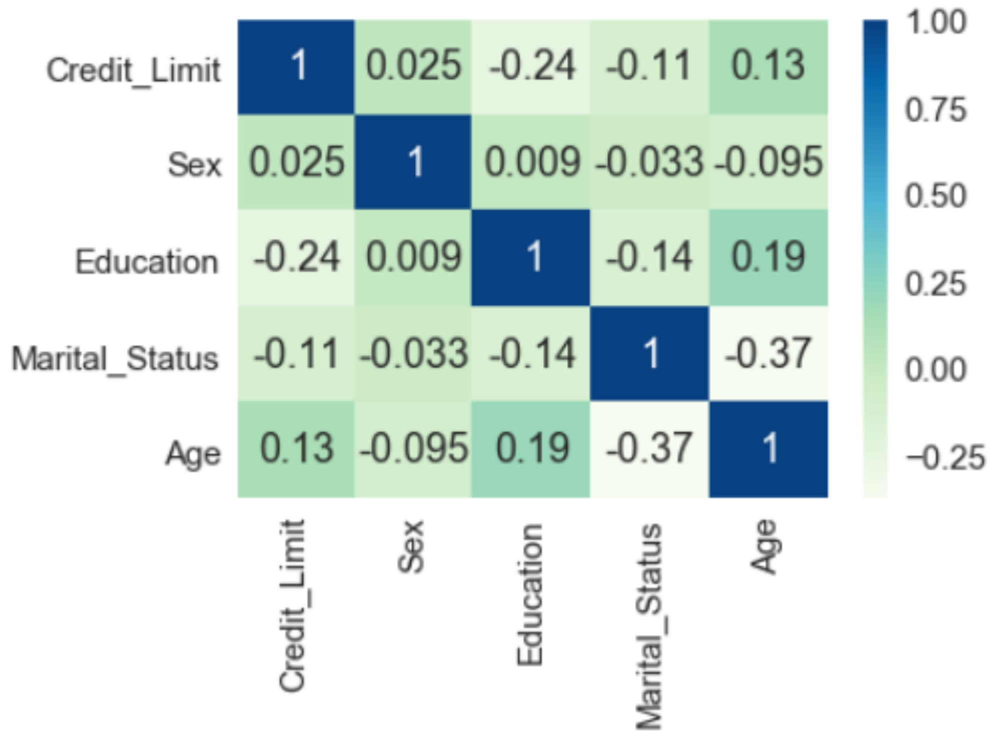


1.0
0.8
0.6
0.4
0.2
0.0
-0.2
July_Payment_Status
pearsonr = -0.15; p = 3e-159
0
5000
10000
15000
20000
25000
30000
35000
Credit_Limit

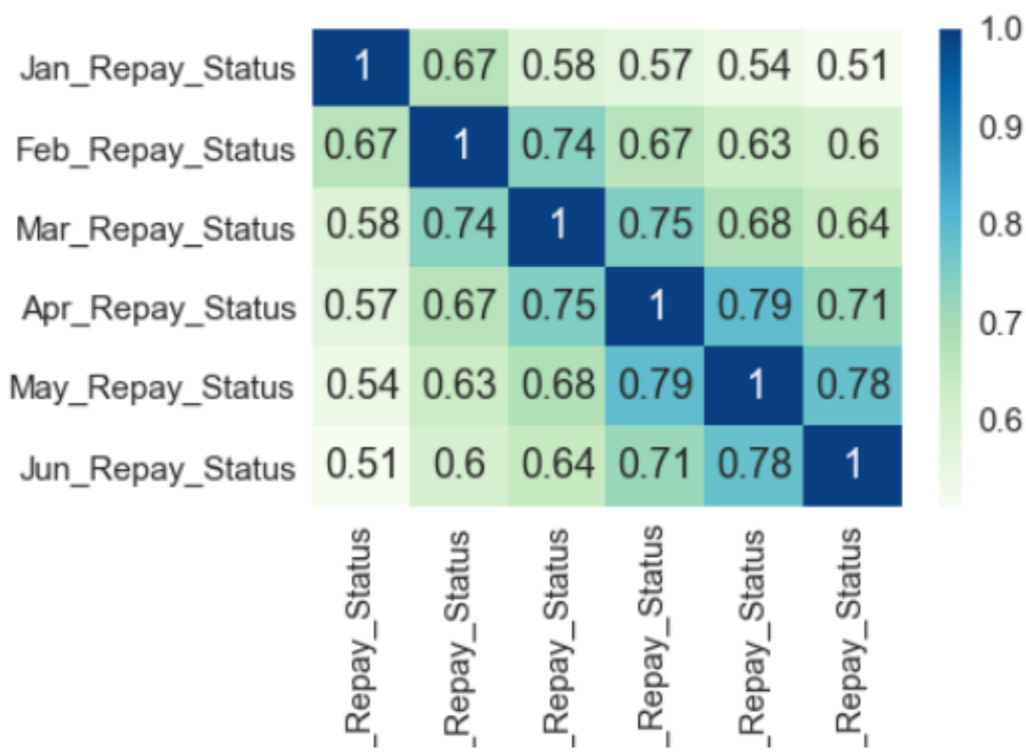

pearsonr = 0.02; p = 0.00061
1.0
0.8
0.6
0.4
0.2
0.0
July_Payment_Status
20
40
60
80
100
Age

Pearson Correlation of Features



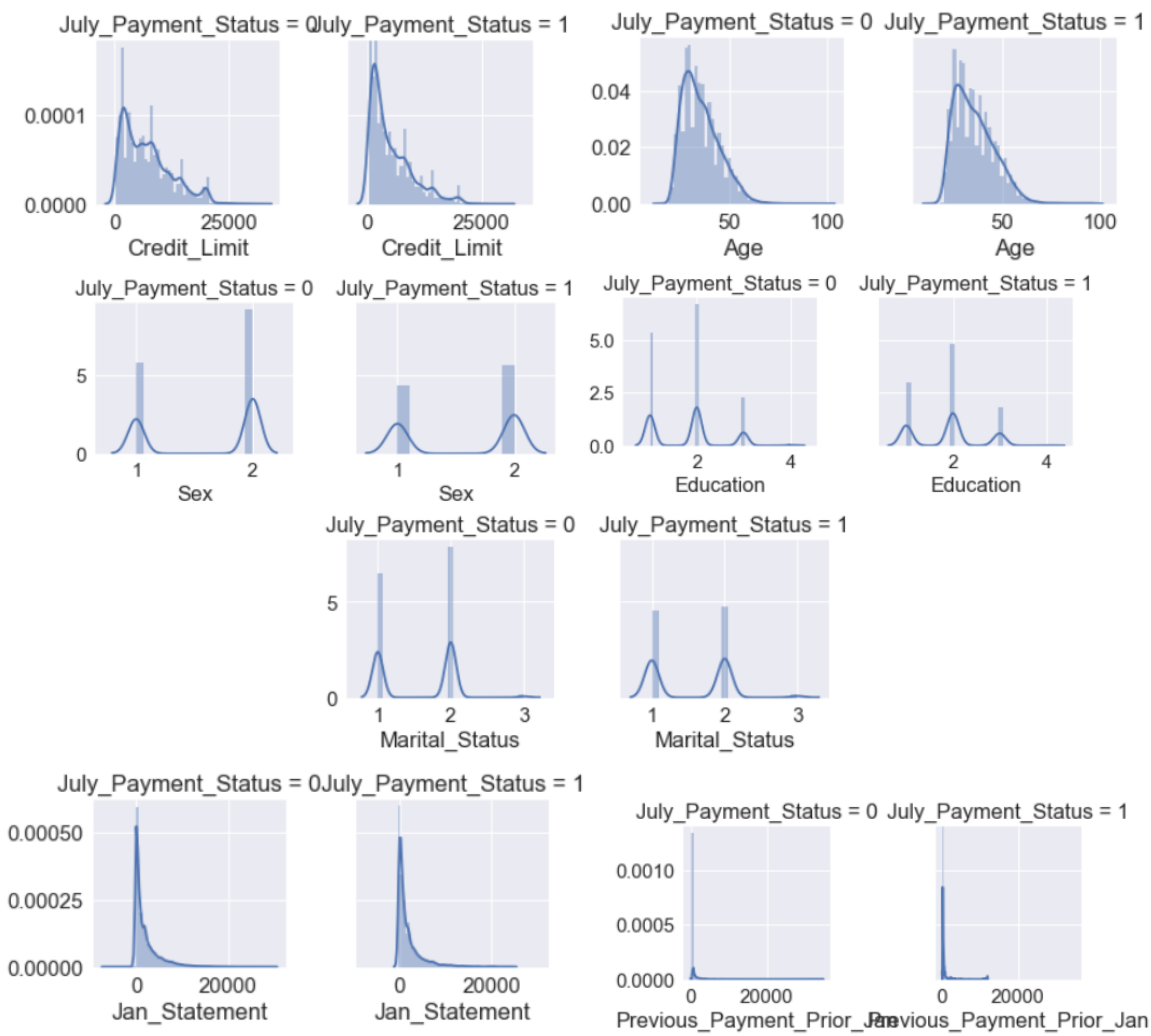


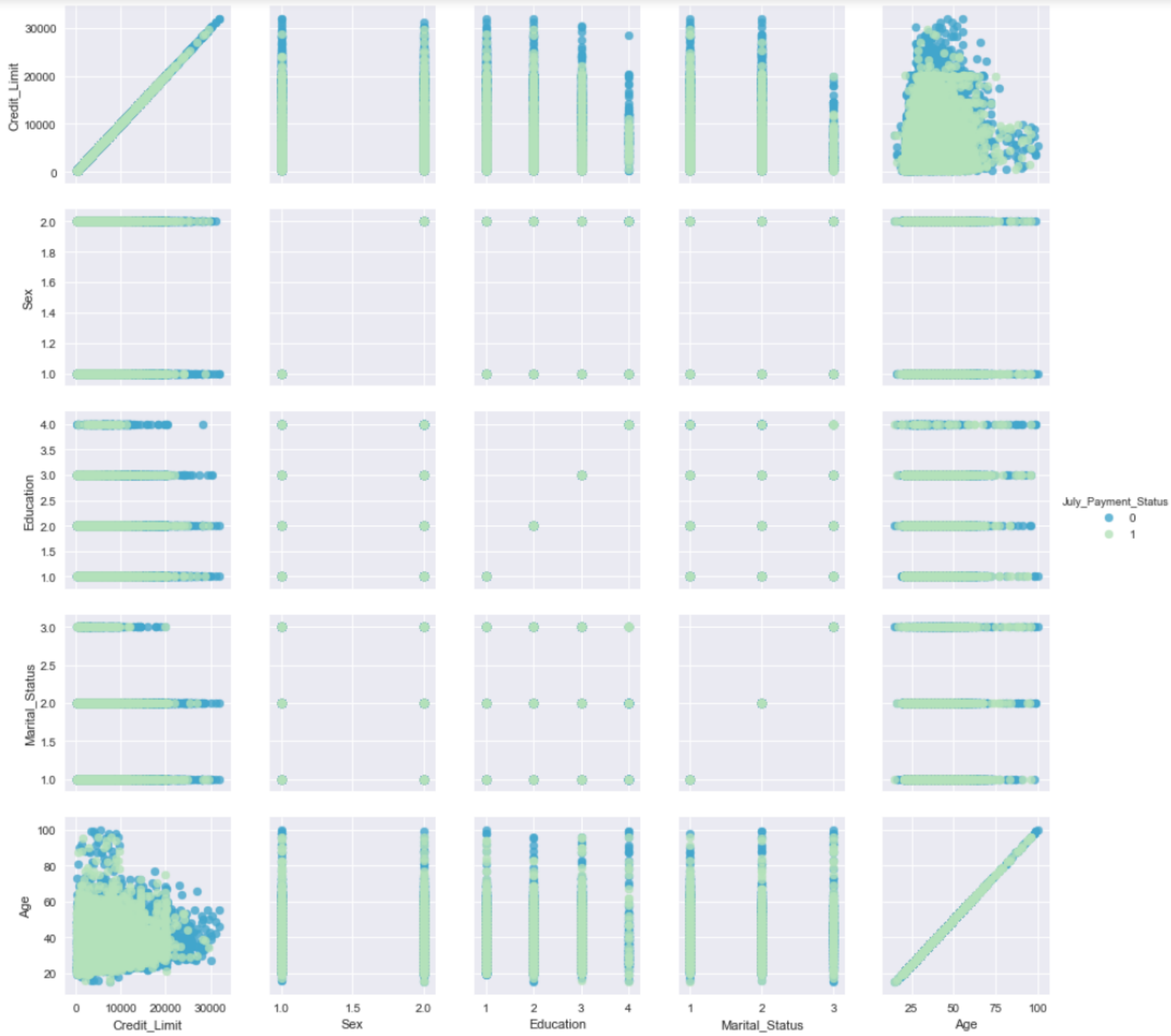
Above variables have weak correlation with each other, which coincides with common sense.



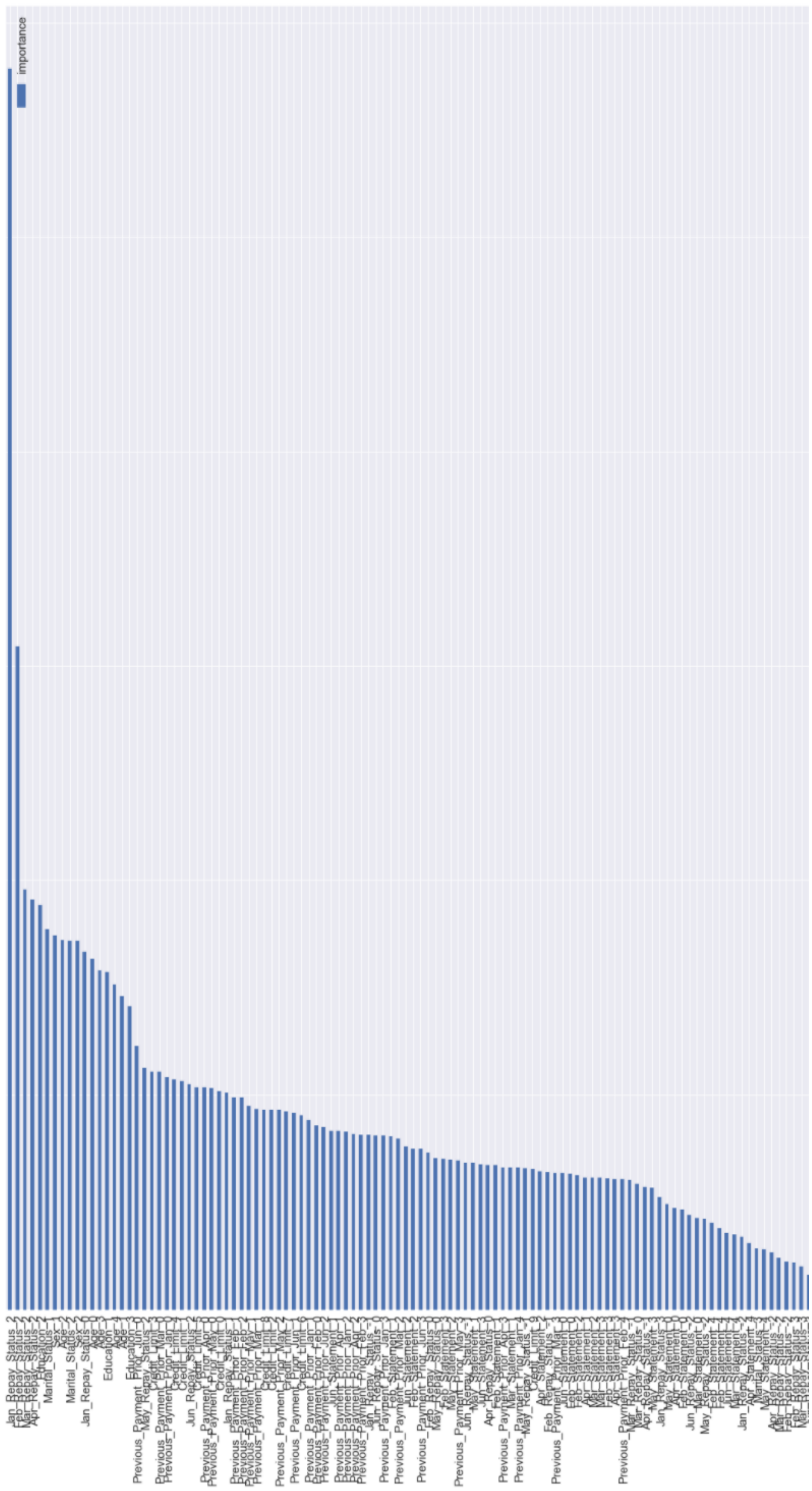
Note that the closer the (previous) month is, the higher the correlation.


Target Distribution
Frequency
0
2000
4000
6000
8000
10000
12000
14000
0
1
July_Payment_Status


July_Payment_Status = 0
July_Payment_Status = 1
Credit_Limit
Age
Sex
Education
Marital_Status
Jan_Statement
Previous_Payment_Prior_Jan


Credit_Limit
Sex
Education
Marital_Status
Age
July_Payment_Status
0
1

Data Modelling (1)



Used qcut for continuous variables

This graph is slightly messy.

It can be further improved by grouping columns together (e.g. repay_status 8~12 etc.)

We have corrected -2~0 as 0 for Repay Status (Nonetheless, the distribution is still somewhat skewed, so it could be improved).

Used random forest classifier to generate probability of default and tested with test smaple.

Out[53]:

	Probability
0	0.589750
1	0.163333
2	0.249895
3	0.117000
4	0.586833

The experimental Value of the Probability is 3503.874, against my test file with Accepted Value 3300 defaulters => 93.82 Accuracy (in a good way since more defaulters are predicted).

We can assign the probability for the new credit card applicant as credit score (simple approach).

Comparing Models

Logistic Regression

```
roc_auc_score: 0.835942238116
accuracy: 0.743745261562
---------------------
Predicted    0     1
Actual
0          795   654
1          360  2148
----------------------------------------------------
             precision    recall  f1-score   support

          0       0.69      0.55      0.61      1449
          1       0.77      0.86      0.81      2508

avg / total       0.74      0.74      0.74      3957
```

Decision Tree Classifier

```
roc_auc_score: 0.84745295386
accuracy: 0.857973212029
---------------------
Predicted     0     1
Actual
0          1171   278
1           284  2224
----------------------------------------------------
             precision    recall  f1-score   support

          0       0.80      0.81      0.81      1449
          1       0.89      0.89      0.89      2508

avg / total       0.86      0.86      0.86      3957
```

RandomForest Classifer

```
roc_auc_score: 0.959704239739
accuracy: 0.899924184989
---------------------
Predicted     0     1
Actual
0          1284   165
1           231  2277
----------------------------------------------------
             precision    recall  f1-score   support

          0       0.85      0.89      0.87      1449
          1       0.93      0.91      0.92      2508

avg / total       0.90      0.90      0.90      3957
```

Gradient Boost Classifier

```
roc_auc_score: 0.963901299142
accuracy: 0.897397017943
---------------------
Predicted     0     1
Actual
0          1310   139
1           267  2241
----------------------------------------------------
             precision    recall  f1-score   support

          0       0.83      0.90      0.87      1449
          1       0.94      0.89      0.92      2508

avg / total       0.90      0.90      0.90      3957
```

Cross-Validation

```
logistric regression: 0.83139841444
decision tree: 0.838458987724
random forest: 0.95322533704
gradient boosting: 0.961566894862
```

GridSearchCV - Logistic Regression

```
roc_auc_score: 0.834148667673
accuracy: 0.742228961334
---------------------
Predicted     0     1
Actual
0           800   649
1           371  2137
-----------------------------------------------------
             precision    recall  f1-score   support

          0       0.68      0.55      0.61      1449
          1       0.77      0.85      0.81      2508

avg / total       0.74      0.74      0.74      3957
```

Voting Based Ensemble Model

```
roc_auc_score: 0.906420090631
accuracy: 0.817538539297
---------------------
Predicted     0     1
Actual
0           984   465
1           257  2251
-----------------------------------------------------
             precision    recall  f1-score   support

          0       0.79      0.68      0.73      1449
          1       0.83      0.90      0.86      2508

avg / total       0.82      0.82      0.81      3957
```

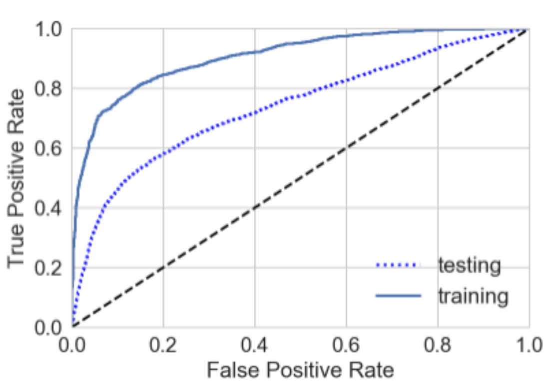
Testing on Acutal Dataset

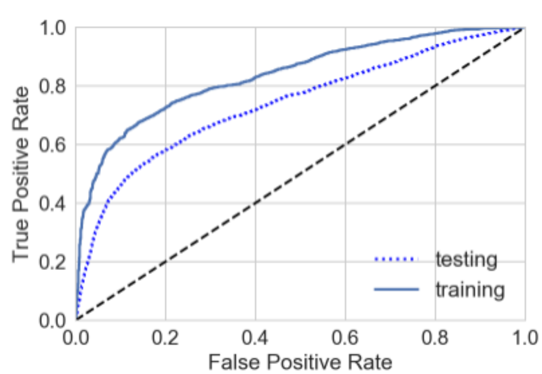
Voting Based Ensemble

```
roc_auc_score: 0.738464767066
accuracy: 0.59904965868
---------------------
Predicted     0     1
Actual
0          6486  5156
1           835  2465
-----------------------------------------------------
             precision    recall  f1-score   support

          0       0.89      0.56      0.68     11642
          1       0.32      0.75      0.45      3300

avg / total       0.76      0.60      0.63     14942
```

Logistics Regression

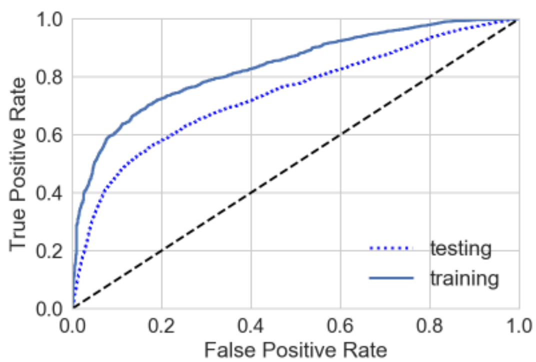
```
roc_auc_score: 0.72174526141
accuracy: 0.51258198367
---------------------
Predicted     0     1
Actual
0          5054  6588
1           695  2605
-----------------------------------------------------
             precision    recall  f1-score   support

          0       0.88      0.43      0.58     11642
          1       0.28      0.79      0.42      3300

avg / total       0.75      0.51      0.54     14942
```





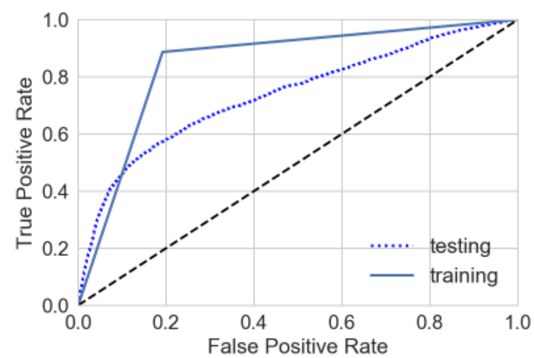
GridSearch - Logistic Regression

```
roc_auc_score: 0.71919089972
accuracy: 0.515392852362
---------------------
Predicted     0     1
Actual
0          5112  6530
1           711  2589
----------------------------------------------------
             precision    recall  f1-score   support

          0       0.88      0.44      0.59     11642
          1       0.28      0.78      0.42      3300

avg / total       0.75      0.52      0.55     14942
```



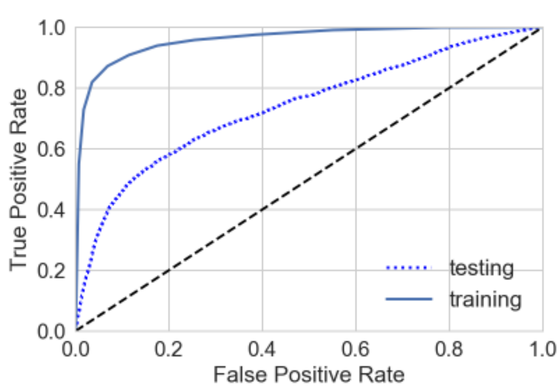
Decision Tree Classifier

```
roc_auc_score: 0.643573529488
accuracy: 0.687525097042
---------------------
Predicted     0     1
Actual
0          8409  3233
1          1436  1864
----------------------------------------------------
             precision    recall  f1-score   support

          0       0.85      0.72      0.78     11642
          1       0.37      0.56      0.44      3300

avg / total       0.75      0.69      0.71     14942
```



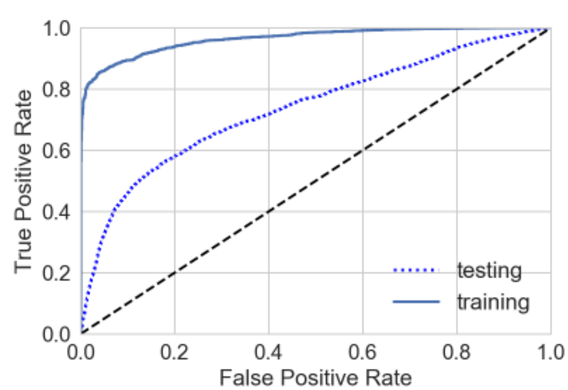
Random Forest Classifier

```
roc_auc_score: 0.741427524689
accuracy: 0.749364208272
---------------------
Predicted     0     1
Actual
0          9300  2342
1          1403  1897
----------------------------------------------------
             precision    recall  f1-score   support

          0       0.87      0.80      0.83     11642
          1       0.45      0.57      0.50      3300

avg / total       0.78      0.75      0.76     14942
```



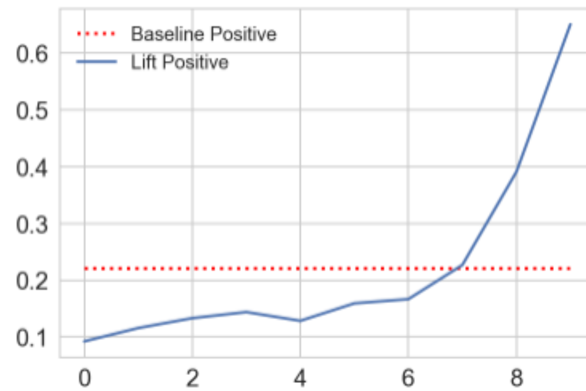
Gradient Boost Classifier

```
roc_auc_score: 0.77039272644
accuracy: 0.775598982733
---------------------
Predicted     0     1
Actual
0          9669  1973
1          1380  1920
----------------------------------------------------
             precision    recall  f1-score   support

          0       0.88      0.83      0.85     11642
          1       0.49      0.58      0.53      3300

avg / total       0.79      0.78      0.78     14942
```



It can be seen clearly that logistic regression gave the best performance among the rest with an AUC score of 0.738, Ensemble, Random Forest and Boost Classifier lead to overfitting and does not show clear improvement on the testing set. Unfortunately, in credit risk an AUC of 0.75 or higher is the industry accepted standard and prerequisite to model acceptance, so clearly there is room for improvement.

Bin_Positive	Baseline_positive	Lift_positive	Lift_positive_index
0	0.220854	0.092308	41.795804
1	0.220854	0.115797	52.431260
2	0.220854	0.133199	60.311103
3	0.220854	0.143909	65.160237
4	0.220854	0.128514	58.189607
5	0.220854	0.159304	72.130867
6	0.220854	0.166667	75.464646
7	0.220854	0.226908	102.741025
8	0.220854	0.390897	176.993388
9	0.220854	0.650836	294.690706



The lift analysis shows little flaws as the few small noticeable peaks are relatively smooth, which shows that the model reflects reality decently. Overall, logistic regression performs within expectations as shown at the rightmost bucket is highest.

Data Modelling (2)

Null Model

PREDICTION TRUE	pay	default	Total
pay	3476	0	3476
default	1008	0	1008
Total	4484	0	4484

Logistic Regression

PREDICTION TRUE	pay	default	Total
pay	3358	118	3476
default	740	268	1008
Total	4098	386	4484

Classification Trees

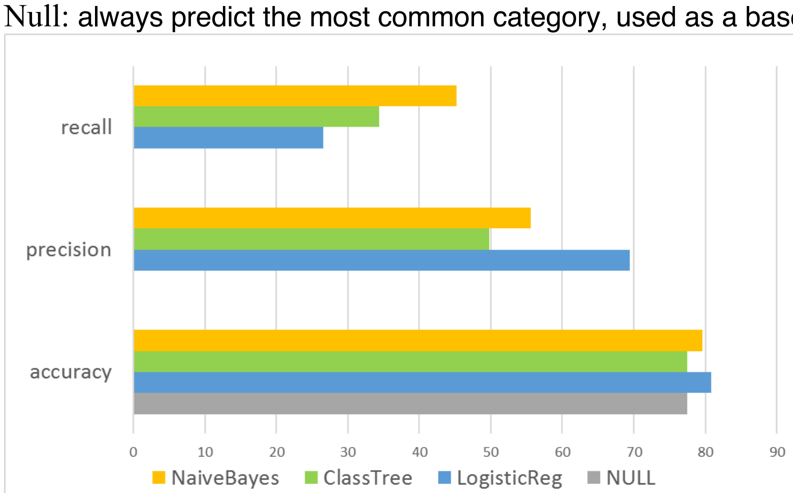
PREDICTION TRUE	pay	default	Total
pay	3126	350	3476
default	661	347	1008
Total	3787	697	4484

Naive Bayes Classifier

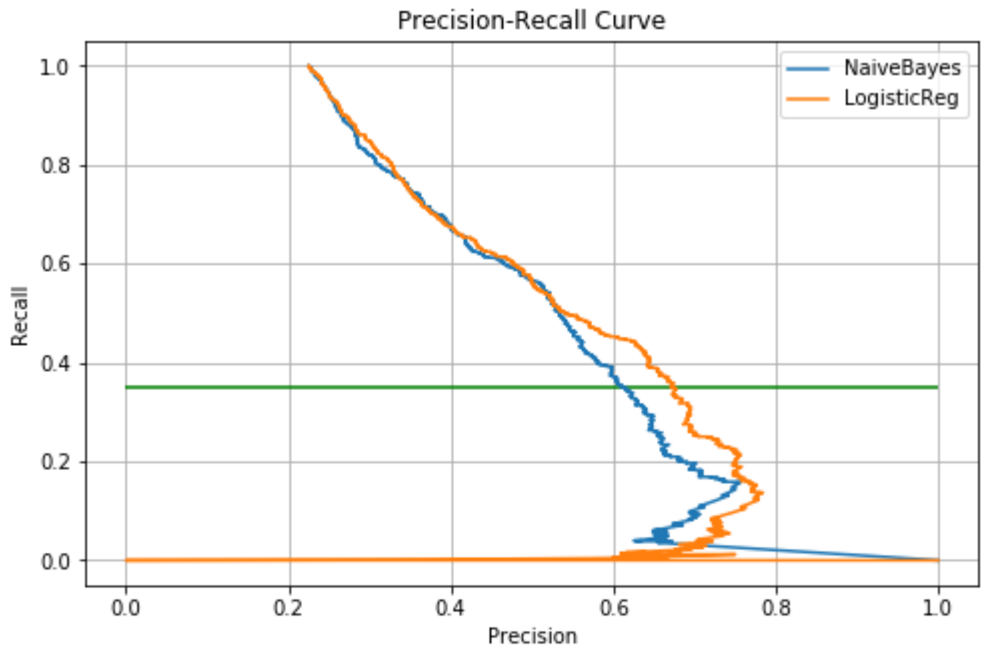
PREDICTION TRUE	pay	default	Total
pay	3112	364	3476
default	552	456	1008
Total	3664	820	4484

	NULL	LogisticReg	ClassTree	NaiveBayes
accuracy	77.5201	80.8653	77.4532	79.5718
precision	0	69.4301	49.7848	55.6098
recall	0	26.5873	34.4246	45.2381

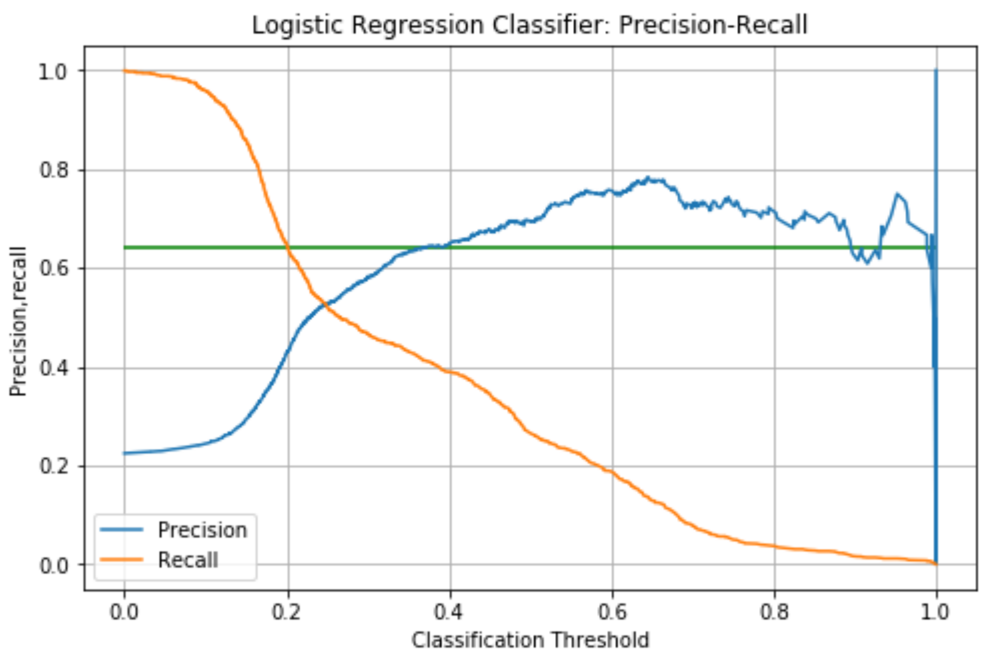
Null: always predict the most common category, used as a base case



best model with precision is LR but best with recall (which is more important) is NB accuracy is relatively equal for the three models
but results could be changed drastically if threshold was changed



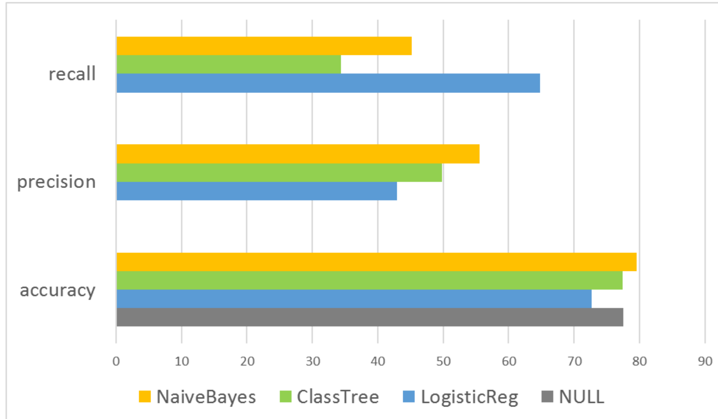
LR is slightly better than NB: for a given level of recall, LR performs better than NB



lower threshold increases the likelihood of being predicted as default, hence increasing recall, but precision will decrease as a result given our conservative consideration (unpredicted default may pose bigger problem), this is an acceptable trade-off

```
Recall: 64.7817460317
Precision: 42.9040735874
Accuracy: 72.7029438002
```

PREDICTION	pay	default	Total
TRUE			
pay	2607	869	3476
default	355	653	1008
Total	2962	1522	4484



Making individual predictions

```
In [17]: def make_ind_prediction(new_data):
             data = new_data.values.reshape(1, -1)
             data = robust_scaler.transform(data)
             prob = logistic_regression.predict_proba(data)[0][1]
             if prob >= 0.2:
                 return 'Will default'
             else:
                 return 'Will pay'
```

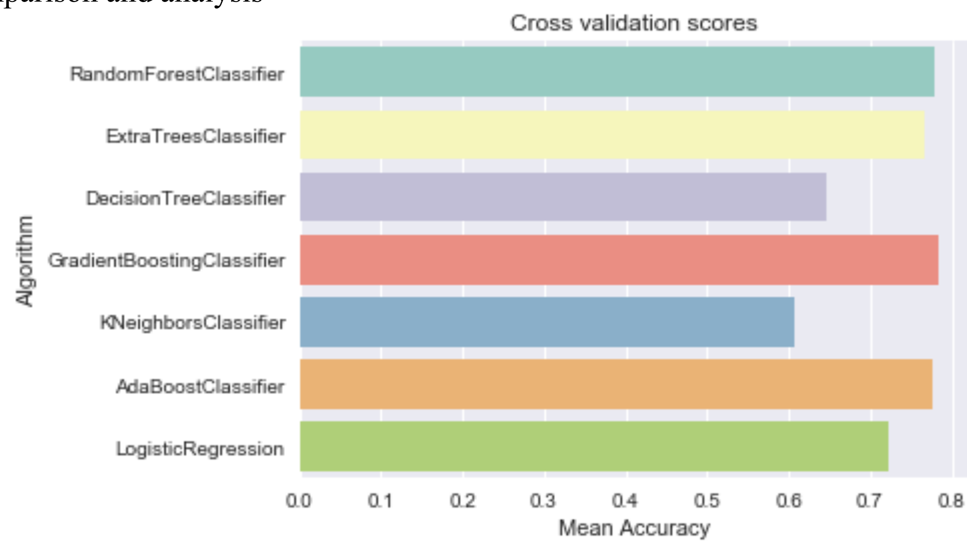
```
In [18]: pay = default[default['default']==0]
```

```
In [20]: from collections import OrderedDict
         new_customer = OrderedDict([('credit_limit', 4000), ('age', 50),
                                     ('jan_repay_status', -1), ('previous_payment_prior_jan', 0),
                                     ('jan_statement', 500),('feb_repay_status', 0),
                                     ('previous_payment_prior_feb', 0),('feb_statement', 35509),
                                      ('mar_repay_status', 0),('previous_payment_prior_mar', 0),
                                     ('mar_statement', 689),('apr_repay_status', 0),
                                      ('previous_payment_prior_apr', 0), ('apr_statement', 0),
                                     ('may_repay_status', 0),('previous_payment_prior_may', 0),
                                     ('may_statement', 0),('jun_repay_status', 0),
                                      ('previous_payment_prior_jun', 0),('jun_statement', 0),
                                      ('july_payment_status', 0),
                                     ('master\'s degree & doctoral degree', 0),
                                     ('bachelor\'s degree', 0), ('high School\'s degree', 0),
                                     ('married', 1),('single', 0)])
         new_customer = pd.Series(new_customer)
         make_ind_prediction(new_customer)

Out[20]: 'Will default'
```

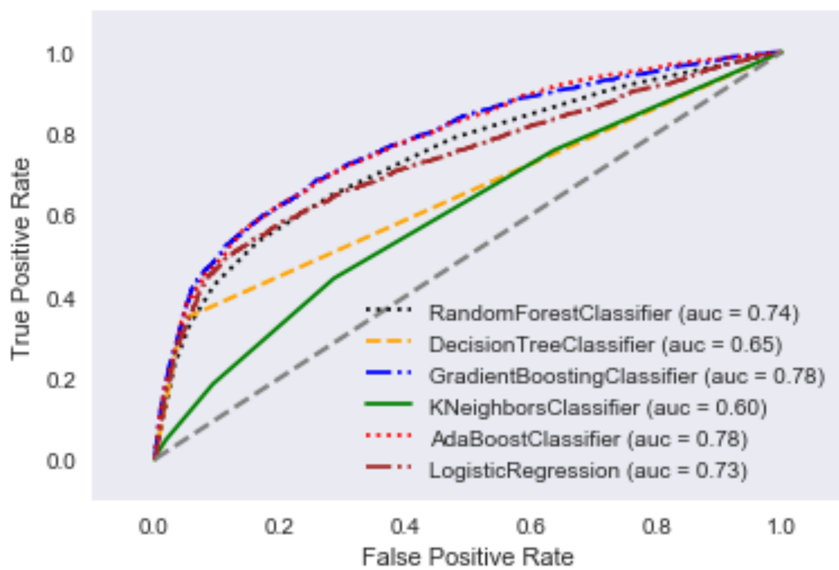
This model is even simpler, simply by predicting if the new applicant will default.

Other Material

Some comparison and analysis



ROC Curve:



Min-Cost Curve

```
|=============================================================|
|                   Feature     Coefficient                    |
|                   -------     -----------                    |
|                 intercept  :  -0.08865856905365048           |
|              Credit_Limit  :  -4.136162845874544e-05         |
|                       Sex  :  -0.4089395762493226            |
|                 Education  :  -0.07261082643287912           |
|            Marital_Status  :  -0.20981889995885875           |
|                       Age  :  -0.002447224223894175          |
|          Jan_Repay_Status  :  0.7738707917817925             |
| Previous_Payment_Prior_Jan  :  -0.00017966609206909718      |
|             Jan_Statement  :  -7.875996407512707e-05         |
|          Feb_Repay_Status  :  0.17802740287324434            |
| Previous_Payment_Prior_Feb  :  -0.0002599402278097557       |
|             Feb_Statement  :  -1.5023880492144783e-05        |
|          Mar_Repay_Status  :  -0.047933717225651894          |
| Previous_Payment_Prior_Mar  :  0.0001133511563180129        |
|             Mar_Statement  :  0.0002740486759693342          |
|          Apr_Repay_Status  :  0.12009493472367604            |
| Previous_Payment_Prior_Apr  :  -0.0006770830869556257       |
|             Apr_Statement  :  -2.3963213260562036e-05        |
|          May_Repay_Status  :  0.13750412073484533            |
| Previous_Payment_Prior_May  :  3.474917423446337e-05        |
|             May_Statement  :  -7.554021592178651e-05         |
|          Jun_Repay_Status  :  0.08465026678573004            |
| Previous_Payment_Prior_Jun  :  0.0001374334193104724        |
|             Jun_Statement  :  -0.00010608076814533911        |
|                                                             |
|                                Training       Test           |
|         Area Under the ROC Curve: 0.78076      0.73501       |
|           Minimum Cost Per Event: 2337.36762   2545.38578    |
|                     at threshold: -0.8273779713              |
|               Condition Incidence: 0.2415       0.2415       |
|      Probability of True Positives: 0.126       0.112        |
|   Test (Classification) Indidence: 0.15083     0.13301       |
|=============================================================|
```

Because the difference Area Under the ROC Curve for the training data and the test data is small, we can tell that this model is quite robust. If we didn't know the cost per false positive and cost per false negative, we'd stop here and know our model is pretty good. But with that information included, we can see that Minimum Cost Per Event for both training and test data are also very close. This also tells us our threshold: -0.7550077141.

We now know that for a bank to maximize profits, they should put each application to a test of:

-0.05993497005393214
Credit_Limit : -2.2643978164886102e-05 |
Sex : -0.3328744363145741 |
Education : -0.07360755731444764 |
Marital_Status : -0.1604192700256958 |
Age : 0.002856709986298127 |
Jan_Repay_Status : 0.5283071703974944 |
Previous_Payment_Prior_Jan : -0.00022701497676949007 |
Jan_Statement : -0.00014932851410334562 |
Feb_Repay_Status : 0.14275210240318223 |

Previous_Payment_Prior_Feb : -0.00025566945568331534 |
Feb_Statement : -1.8480409145272735e-05 |
Mar_Repay_Status : -0.059158684303796484 |
Previous_Payment_Prior_Mar : 0.00013925464219661843 |
Mar_Statement : 0.00027710851720739992 |
Apr_Repay_Status : 0.04959604226000935 |
Previous_Payment_Prior_Apr : -0.0005689907699989463 |
Apr_Statement : -4.1194600110986226e-05 |
May_Repay_Status : 0.09586304419201959 |
Previous_Payment_Prior_May : 3.117998499414399e-05 |
May_Statement : -7.181084692175177e-05 |
Jun_Repay_Status : 0.0011831585562662168 |
Previous_Payment_Prior_Jun : 0.0001769404100671579 |
Jun_Statement : -8.168264199434573e-05

If the sum of them is greater than the threshold, -0.6698867856, the bank should reject the credit card application. In the data given, there is a 24.15% default rate on 3000 applications, with a cost per false negative of $5000. If the bank were to simply accept all applications, the defaulted credit card accounts would cost the bank

24.15% ∗5000∗2000= $2,415,000

which results in a cost per application event of 2415000 / 2000 = $1207.5. This logistic regression/binary classification model has a cost per event of $2594.55 on the test set, which indicates that this model would save the bank 2594.55 - 1207.5 = $1387.05 per application, if we assume that these 3000 applications are representative of all future applications.