**Machine Learning Classification: Prediction of Loan Approval**

**Objective:** We would need to predict whether a Loan Application would be approved or rejected.

**Training Data:** The data is here (refer git repo for this project): [train\_u6lujuX\_CVtuZ9i](https://rajivsworklife.files.wordpress.com/2017/09/train_u6lujux_cvtuz9i.xlsx)

**Step 1 – Exploratory Data Analysis** :

a. All records with blank fields are weeded out.

b. We then determine features that are categorical and those that are continuous.  Out of these we see that the following features are categorical:

Gender  
Married  
Dependents  
Education  
Self\_Employed  
Loan\_Amount\_Term  
Credit\_History  
Property\_Area

The following features are continuous:

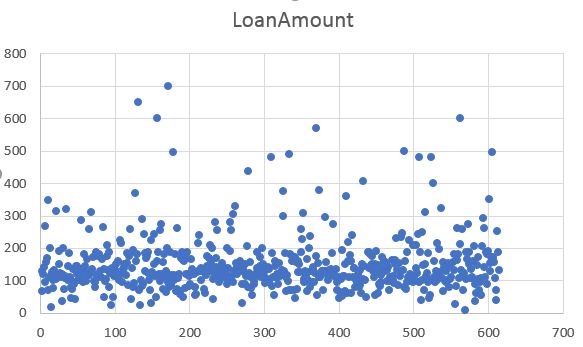
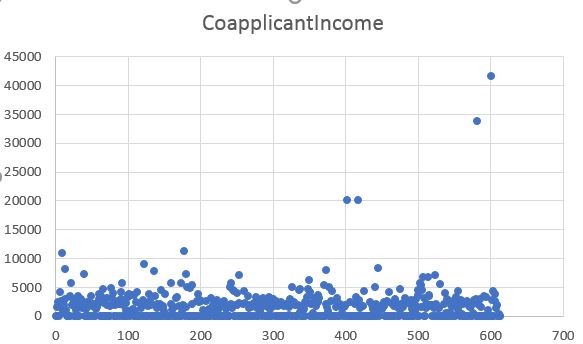
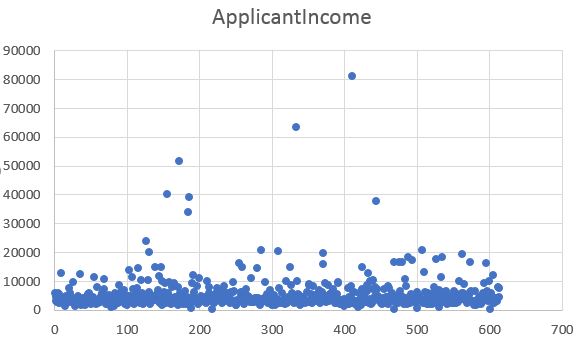
ApplicantIncome  
CoapplicantIncome  
LoanAmount

The Loan\_ID field is the identifier field whereas the **Loan\_Status is the Label field**

c. We next move on to identify the outlier data for the above three continuous features.  Based on the below charts we determine that the outliers for the following features are as below:

Loan Amount – anything above 500 is considered an outlier

Co-applicant Income – anything above 11300  is considered an outlier

Applicant Income – anything above 23803  is considered an outlier********

We use both scatter charts and box charts.  However it is seen that identification of outliers was easier for me using scatter charts.

d. Next we carve out a dataset consisting only of outlier data corresponding to the above three features. We keep this aside for testing later.

e. We finally create a dataset that is devoid of all records with null value features and outlier data.  From this we create two subsets of data : the training data and the validation data.

**Step 2 – Model creation and training** :

The problem statement at hand is to determine whether a loan would be approved or not.  Hence it is a classification problem.  We therefore use multiple classification algorithms to decide the best one.  Using Microsoft Azure Studio for Machine Learning I explored the following five algorithms:

a. Support Vector Machine

b. Logistic Regression

c. Decision Jungle

d. Averaged Perceptron

e. Bayes Point Machine

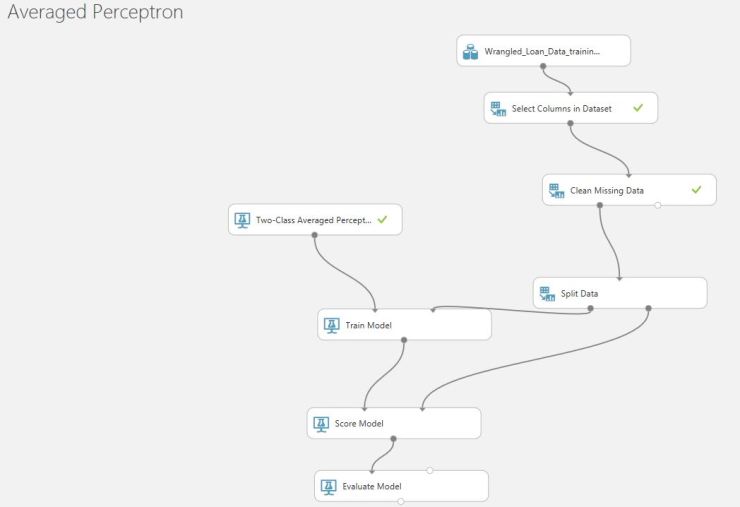
After having created models for each of the above – I also evaluated the metrics for each.  Here is how it looks like:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | AUC | Accuracy | Precision |
| Support Vector Machine | 0.821 | 0.822 | 0.798 |
| Logistic Regression | 0.831 | 0.822 | 0.786 |
| Decision Jungle | 0.812 | 0.822 | 0.792 |
| Averaged Perceptron | 0.799 | 0.829 | 0.797 |
| Bayes Point Machine | 0.804 | 0.818 | 0.788 |

I next tested out these models on the validation dataset and the outlier dataset and the number of errors are as indicated below:

|  |  |  |
| --- | --- | --- |
| Model | Errors in Validation Data | Errors in Outlier Data |
| Support Vector Machine | 5 | 4 |
| Logistic Regression | 5 | 5 |
| Decision Jungle | 5 | 5 |
| Averaged Perceptron | 4 | 5 |
| Bayes Point Machine | 4 | 7 |

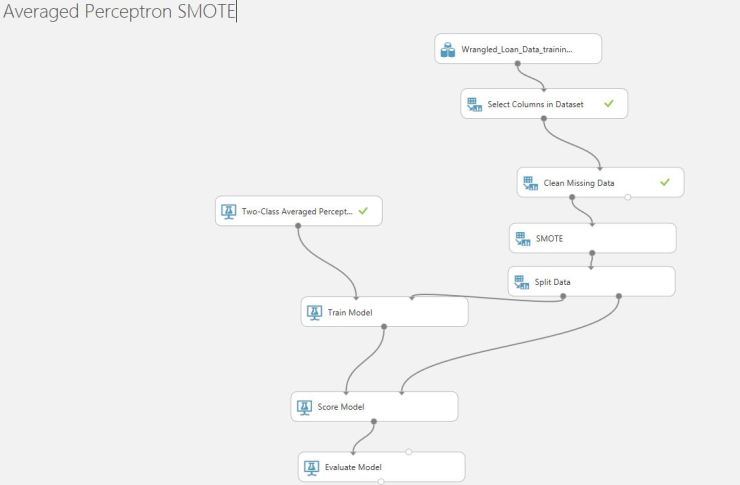
Based on the above two sets of metrics I was able to determine that the Averaged Perceptron algorithm was the best followed by the Support Vector Machine and finally the Logistic Regression.

****

While doing the checks on the validation and outlier datasets – I realized that the training data was heavily biased to those who had a successful loan approval.  I later brought in SMOTE to see if I could get better model and better results ([Check my earlier blog posting on SMOTE](https://rajivsworklife.wordpress.com/2017/09/11/dealing-with-biased-unbalanced-data-for-classification-problems/)).  However – while the AUC metrics improved quite a bit, the Accuracy and Precision metrics deteriorated and the actual validations on the validation and outlier datasets – seemed to paint a dreary picture:

|  |  |  |  |
| --- | --- | --- | --- |
| Model | AUC | Accuracy | Precision |
| Support Vector Machine | 0.855 | 0.765 | 0.652 |
| Logistic Regression | 0.863 | 0.812 | 0.722 |
| Averaged Perceptron | 0.844 | 0.796 | 0.723 |

|  |  |
| --- | --- |
| Model | Errors in Validation + Outlier Data |
| Support Vector Machine | 18 |
| Logistic Regression | 11 |
| Averaged Perceptron | 13 |

****

I had to revert back to the model without the usage of SMOTE – as that seemed to be doing better.

**Step 3 – Trained Models**

After having done all that here are the final models that were created – which were run on the test data ([test\_loan](https://rajivsworklife.files.wordpress.com/2017/09/test_loan.xlsx" \o "test_loan) – refer git repo)

[**Averaged Perceptron (Predictive Exp.)-09-13-2017 17-26-03**](https://rajivsworklife.files.wordpress.com/2017/09/averaged-perceptron-predictive-exp-09-13-2017-17-26-03.xlsx)

[**Two-Class Logistic Regression (Predictive Exp.)-09-13-2017 17-46-46**](https://rajivsworklife.files.wordpress.com/2017/09/two-class-logistic-regression-predictive-exp-09-13-2017-17-46-46.xlsx)

[**Two Class Support Vector Model (Predictive Exp.)-09-13-2017 17-52-26**](https://rajivsworklife.files.wordpress.com/2017/09/two-class-support-vector-model-predictive-exp-09-13-2017-17-52-26.xlsx)