Advances in Data Science

Deployment of Machine Learning Models On Cloud

Team 1

Vishal Satam

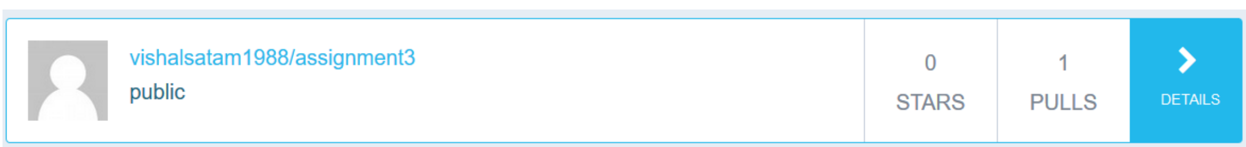
Manasi Dalvi

Overview

This project builds on the previous project where we have developed models for prediction and classification. We have developed 6 machine learning algorithms to perform prediction and classification on the Freddie Mac's loans dataset. 3 prediction algorithms to predict interest rates and 3 classification algorithms for classifying a loan as delinquent. The best performing algorithms are drawn out as described further in this document by performing a comparative analysis of the results obtained by applying the different algorithms available in Microsoft Azure ML Studio. The algorithms for prediction that are being used are Linear Regression, Boosted Decision Tree and Neural Network. The algorithms for classification that are being used are Logistic Regression, Decision Jungle Classification and Bayes Point Machine. The goal of this project is to deploy the machine learning algorithms on a cloud environment so that these algorithms will be available as a REST API. We can then invoke these REST API's by passing the required features which will be used for prediction/classification from a cloud based web application developed in Flask and hosted on IBM Bluemix which takes input from the user from the UI and makes a HTTP request to the REST API hosted on Microsoft Azure ML Studio. The results of the Prediction /Classification are sent back to the UI and presented to the user. The best algorithm for each task has been highlighted on the UI and the user has been provided with options to choose either of the 6 algorithms.

Docker & Execution

The docker execution instructions have been uploaded on the Readme.md file in the github repository. Minimum required memory for Docker machine = 6 GB RAM. The docker repository exists on the docker hub with the image name : vishalsatam1988/assignment3



Dataset & Wrangling

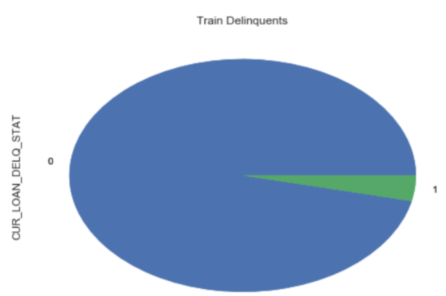
The Freddie Mac's Single Family loans dataset has been used for this project. We have written a script to automatically login, scrape links and download the sample files from the Freddie Mac's website. We use the sample files for this project because the size of the historical files is too big and will not be supported while deploying on Microsoft Azure Machine Learning Studio for the Free account. We would have to summarize these files anyway. This is the reason we have used the already summarized files. From Freddie Mac's documentation, we can see that they have used Simple Random Sampling to generate these files. These should give us the required data. We perform wrangling on these files and generate the required output files for origination summary to predict interest rates and generate train and test files using Random Undersampling described in the next section. We have also learnt from the previous projects that the algorithms tend to perform worse in certain special situations related to the year of the data. That is why we are developing the models on data from all the years to better predict / classify future loans after 2016

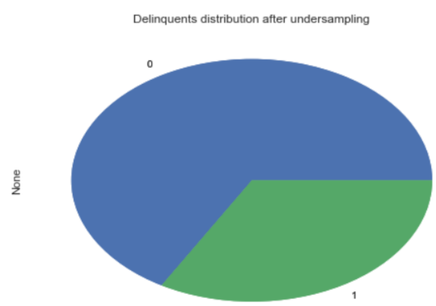
Random Undersampling

Random Undersampling has been chosen to modify the training dataset. Since, we had a very imbalanced dataset, we were not getting good results in any of our models. To allow the algorithms to train on better data, we have performed Random Undersampling on the train dataset. 0.5 was chosen as the ratio. We were able to get a 65:35 ratio of Non-Delinquents to Delinquents. With this sampled dataset, we can apply the classification algorithms on the train dataset to train and develop the models and perform a comparative analysis to determine which model would give us better results.

Random Undersampling was chosen because the other algorithms such as SMOTE takes up a lot of memory and is not practical to execute on local machines or on docker containers. We also thought of executing SMOTE on Microsoft Azure, but that exponentially increases the execution time of the training phase on Microsoft Azure.

Number of Delinquents in the train dataset before and after random undersampling.






```
def create_train_test_sample():
    DATAPATH=os.environ['DATAPATH']+"/"
    CONFIGFILEPATH = os.environ['CONFIGPATH']+"/"

    FILENAMEORIG="originationsummary.csv"
    FILENAMESUMMARY="performancesummary.csv"
    OUTPUTRESAMPLEDTRAIN="train_with_time.csv"
    OUTPUTRESAMPLEDTEST="test_with_time.csv"

    print("Splitting Performance data into train and test.")
    df_summary=pd.read_csv(DATAPATH+FILENAMESUMMARY)
    df_summary['MONTHLY_REPORT_PERIOD']=df_summary['MONTHLY_REPORT_PERIOD'].apply(lambda x : x.replace('-',''))
    cols=['CUR_ACT_UPB','LOAN_AGE','MONTHS_LEGAL_MATURITY','CURR_INTERESTRATE','CURR_DEF_UPB','MONTHLY_REPORT_PERIOD'
    X_train,X_test,y_train,y_test = train_test_split(df_summary[cols],df_summary['CUR_LOAN_DELQ_STAT'],train_size=0.7
    df_summary=""
    print("Performing Random Undersampling on the train data.")
    us=RandomUnderSampler(ratio=0.5,random_state=1)
    X_train, y_train = us.fit_sample(X_train, y_train)
    X_train=pd.DataFrame(data=X_train[0:,0:],columns=cols)
    X_train['CUR_LOAN_DELQ_STAT']=pd.Series(data=y_train)

    X_test['CUR_LOAN_DELQ_STAT']=y_test

    print("Saving Train and Test Files")
    X_train.to_csv(DATAPATH+OUTPUTRESAMPLEDTRAIN,index=False)
    X_test.to_csv(DATAPATH+OUTPUTRESAMPLEDTEST,index=False)
```

Machine Learning Algorithms

Prediction

The training sample used is all the sample files from 1999 to 2016. Each file consists of 50000 records randomly sampled from the respective historical files. We decided to use the sample files for training purpose to avoid biased training of the model.

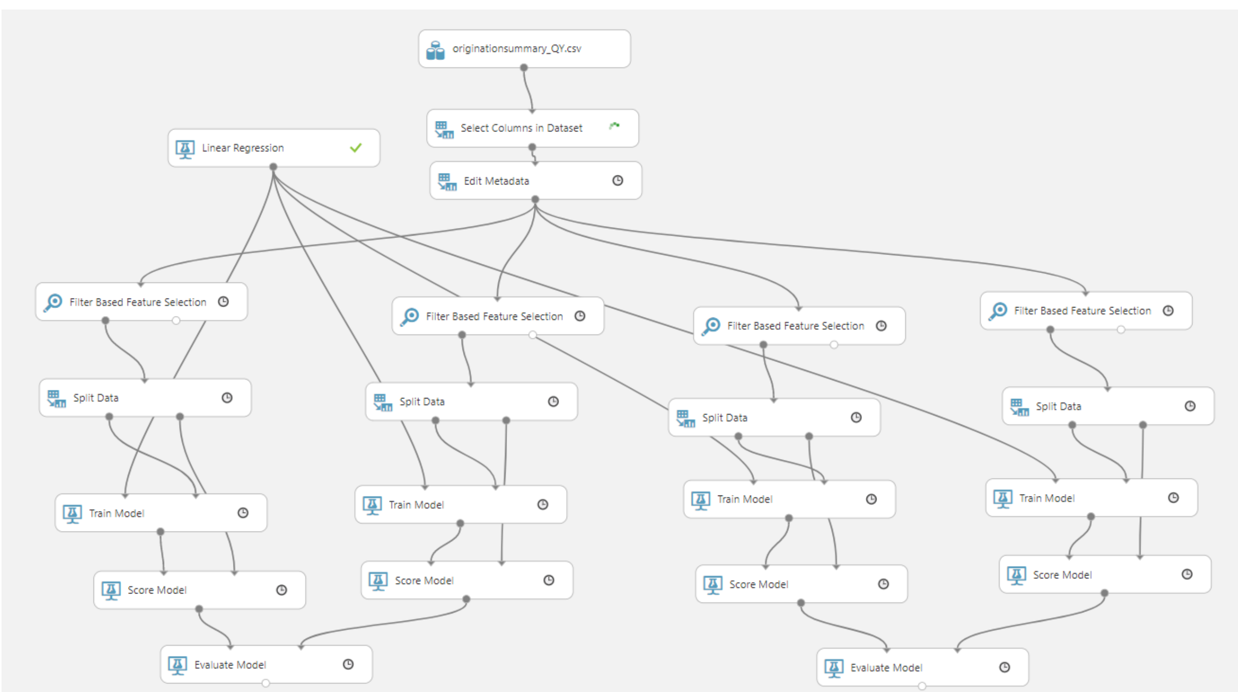
Derived variables: From the LOAN_SEQUENCE_NUMBER we extracted *loan_origination_year, loan_origination_quater*. The OG_OUATERYEAR variable is used as a feature.

```
st = datetime.datetime.fromtimestamp(time.time()).strftime('%Y-%m-%d %H:%M:%S.%f')
print("Summarizing : "+st + ":"+filepath)
logging.info("Summarizing : "+st + ":"+filepath)
df_orig=cleanOrig(filepath)
df_orig['OG_YEAR'] = ['19'+x if x=='99' else '20'+x for x in (df_orig['LOAN_SEQ_NO'].apply(lambda x: x[2:4]))]
df_orig['OG_QUARTER'] = [x for x in (df_orig['LOAN_SEQ_NO'].apply(lambda x: x[4:6]))]
if header is True:
    df_orig.to_csv(file, mode='a', header=True,index=False)
    header = False
```

Code snippet for feature extraction mentioned above

Feature Selection:

The models are trained and built from scratch using Azure. The features are selected using Feature Selection techniques provided by Azure. A basic flow of the model building is given below.



For feature selection we compared results os Pearsons Correaltion, Chi-squared, Spearman Correlation, Mutual Information and tested with a Linear regression model.

The output result set had similar features with similar weights assigned to the features.

OG_QUARTERYEAR	SELLER_NAME	SERVICE_NAME	CHANNEL	OG_LOANTERM	FIRST_HOME_BUYER_FLAG	CREDIT_SCORE	OG_UPB	LOAN_PURPOSE	OG_LTV	OG_CLTV	MI_PERCENT	PROP_TYPE	PROP_STATE	OG_DTI
0.9567	0.600946	0.579678	0.549718	0.441208	0.387022	0.352329	0.298767	0.190638	0.175318	0.172213	0.152382	0.140332	0.104323	0.076048

Pearson’s Correlation

OG_QUARTERYEAR	SELLER_NAME	CHANNEL	SERVICE_NAME	OG_LOANTERM	FIRST_HOME_BUYER_FLAG	CREDIT_SCORE	OG_UPB	OG_LTV	OG_CLTV	LOAN_PURPOSE	MI_PERCENT	PROP_TYPE	PROP_STATE	OG_DTI
0.953676	0.570876	0.560698	0.515465	0.426825	0.383339	0.380147	0.289086	0.184994	0.183498	0.180782	0.1445	0.132799	0.099235	0.08207

Spearman Correlation

OG_QUARTERYEAR	SELLER_NAME	SERVICE_NAME	CHANNEL	OG_LOANTERM	FIRST_HOME_BUYER_FLAG	CREDIT_SCORE	OG_UPB	LOAN_PURPOSE	OG_LTV	OG_CLTV	MI_PERCENT	OG_DTI
2720030.280494	604222.883535	522898.123059	336743.83132	227114.63963	183653.289935	134515.807232	93865.929723	50493.128183	48380.18808	46641.580281	40362.117617	29416.089927

Chi Squared

OG_QUARTERYEAR	SELLER_NAME	SERVICE_NAME	CHANNEL	OG_LOANTERM	FIRST_HOME_BUYER_FLAG	CREDIT_SCORE	OG_UPB	OG_CLTV	LOAN_PURPOSE	OG_LTV	MI_PERCENT	OG_DTI	PROP_TYPE	PROP_STATE
1.175718	0.345041	0.290174	0.248284	0.107738	0.105354	0.082449	0.059097	0.028571	0.028536	0.02704	0.021852	0.018315	0.011938	0.009242

Mutual Information

The Error Metrics

Metrics

Mean Absolute Error	0.254627
Root Mean Squared Error	0.339859
Relative Absolute Error	0.216264
Relative Squared Error	0.0586
Coefficient of Determination	0.9414

Metrics

Mean Absolute Error	0.255172
Root Mean Squared Error	0.340617
Relative Absolute Error	0.216727
Relative Squared Error	0.058861
Coefficient of Determination	0.941139

Pearsons Correlation

Metrics

Mean Absolute Error	0.254627
Root Mean Squared Error	0.339859
Relative Absolute Error	0.216264
Relative Squared Error	0.0586
Coefficient of Determination	0.9414

Spearman Correlation

Metrics

Mean Absolute Error	0.254775
Root Mean Squared Error	0.340106
Relative Absolute Error	0.216389
Relative Squared Error	0.058685
Coefficient of Determination	0.941315

Chi Squared

Mutual Information

We decided to go ahead with Pearson’s Correlation. We fine tuned the features more, starting out with top 10 features to top 7 features. The CHANNEL feature was eliminated as it was adding less value.

Metrics

Mean Absolute Error	0.3059
Root Mean Squared Error	0.403623
Relative Absolute Error	0.259812
Relative Squared Error	0.082651
Coefficient of Determination	0.917349

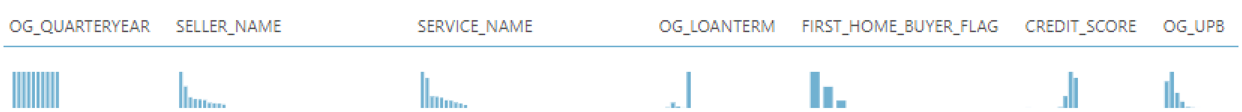
Error Histogram

Metrics

Mean Absolute Error	0.263391
Root Mean Squared Error	0.352472
Relative Absolute Error	0.223707
Relative Squared Error	0.06303
Coefficient of Determination	0.93697

With Channel **Without Channel**

Final set of seven features :



Algorithm Selection:

The selected features were then input to various models. The models and their corresponding error metrics are:

Neural Network

Metrics

Mean Absolute Error	0.240457
Root Mean Squared Error	0.321268
Relative Absolute Error	0.204229
Relative Squared Error	0.052364
Coefficient of Determination	0.947636

Boosted Decision Tree

Metrics

Mean Absolute Error	0.263405
Root Mean Squared Error	0.348794
Relative Absolute Error	0.223719
Relative Squared Error	0.061721
Coefficient of Determination	0.938279

Linear Regression

Metrics

Mean Absolute Error	0.259402
Root Mean Squared Error	0.347199
Relative Absolute Error	0.22032
Relative Squared Error	0.061158
Coefficient of Determination	0.938842

Poisson Regression

Metrics

Mean Absolute Error	0.263306
Root Mean Squared Error	0.350849
Relative Absolute Error	0.223635
Relative Squared Error	0.06245
Coefficient of Determination	0.93755

Bayesian Linear Regression

Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
0.256376	0.34242	0.217749	0.059486	0.940514

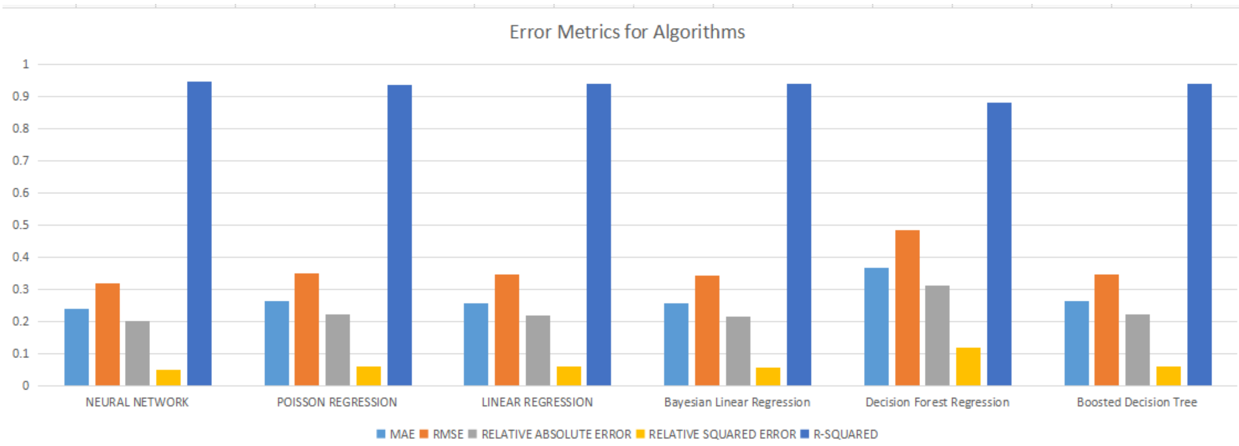
Decision Forest Regression

Mean Absolute Error	Root Mean Squared Error	Relative Absolute Error	Relative Squared Error	Coefficient of Determination
0.368191	0.485764	0.312718	0.119714	0.880286

Decision Forest Regression produced highest amount of error. In place of Decision Forest, we decided to select Boosted Decision Tree algorithm as its compute time was faster than Bayesian Linear Regression. The other algorithms are Linear Regression and Neural Networks.

Following are the error metrics for the six models :

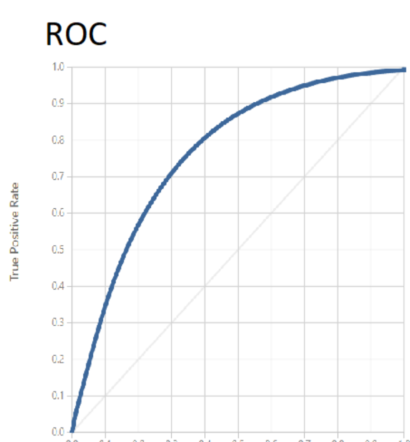
Model	MAE	RMSE	RELATIVE ABSOLUTE ERROR	RELATIVE SQUARED ERROR	R-SQUARED
NEURAL NETWORK	0.240457	0.321268	0.204229	0.052364	0.947636
POISSON REGRESSION	0.263306	0.350849	0.223635	0.06245	0.93755
LINEAR REGRESSION	0.259402	0.347199	0.22032	0.061158	0.938842
Bayesian Linear Regression	0.256376	0.34242	0.217749	0.059486	0.940514
Decision Forest Regression	0.368191	0.485764	0.312718	0.119714	0.880286
Boosted Decision Tree	0.263405	0.348794	0.223719	0.061721	0.938279

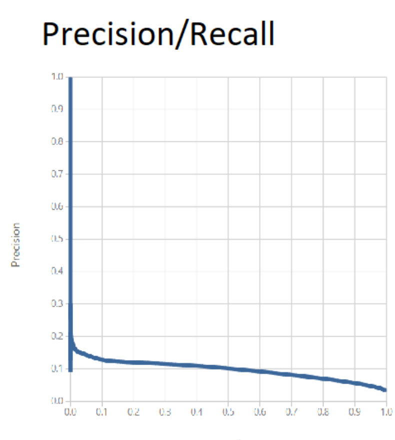


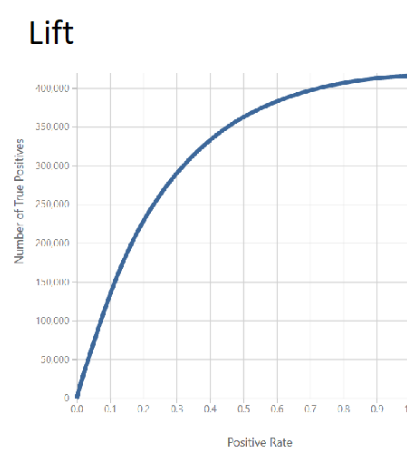
Neural Networks has the least Root Mean Squared Error and an R-squared of 0.94

Classification

Two Class Logistic Regression





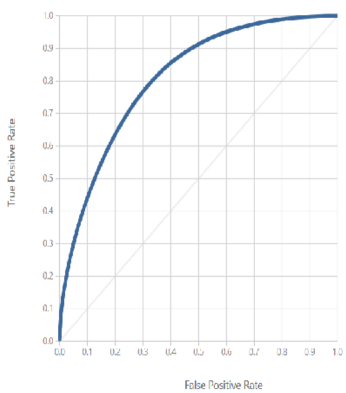


True Positive	False Negative	Accuracy	Precision	Threshold	AUC
146629	273160	0.879	0.112	0.5	0.765
False Positive	**True Negative**	**Recall**	**F1 Score**		
1159202	10261767	0.349	0.170		
Positive Label	**Negative Label**				
1	0				

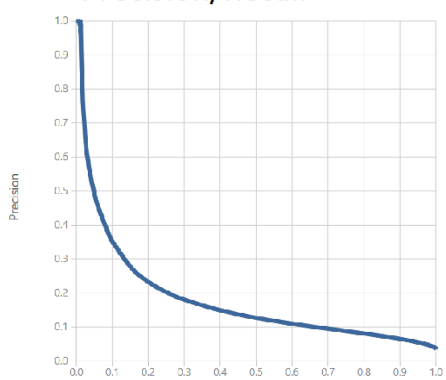
Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	4568	22733	0.002	0.963	0.020	0.167	0.011	0.965	0.998	0.000
(0.800,0.900]	13286	81703	0.010	0.957	0.066	0.146	0.043	0.966	0.991	0.000
(0.700,0.800]	24599	184010	0.028	0.944	0.113	0.128	0.101	0.967	0.975	0.001
(0.600,0.700]	41179	327527	0.059	0.920	0.149	0.120	0.199	0.970	0.946	0.006
(0.500,0.600]	62997	543229	0.110	0.879	0.170	0.112	0.349	0.974	0.899	0.019
(0.400,0.500]	87456	1053003	0.207	0.797	0.163	0.096	0.558	0.980	0.806	0.061
(0.300,0.400]	94546	2038490	0.387	0.633	0.131	0.072	0.783	0.987	0.628	0.183
(0.200,0.300]	61838	3001016	0.645	0.385	0.097	0.051	0.930	0.993	0.365	0.412
(0.100,0.200]	22009	2821461	0.886	0.149	0.076	0.039	0.983	0.995	0.118	0.649
(0.000,0.100]	7311	1347797	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.765

Two Class Decision Jungle

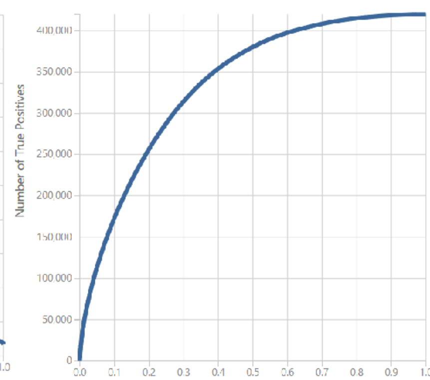
ROC



Precision/Recall



Lift

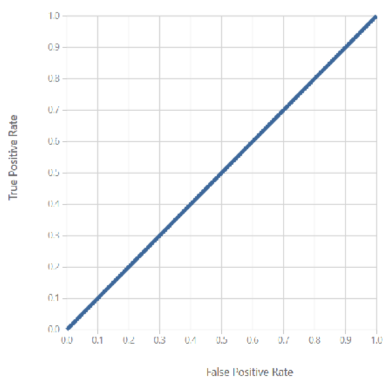


True Positive	False Negative	Accuracy	Precision	Threshold	AUC
220576	199213	0.850	0.123	0.5	0.811
False Positive	**True Negative**	**Recall**	**F1 Score**		
1578654	9842315	0.525	0.199		
Positive Label	**Negative Label**				
1	0				

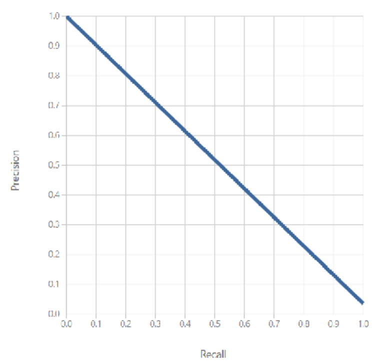
Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	9926	4723	0.001	0.965	0.046	0.678	0.024	0.965	1.000	0.000
(0.800,0.900]	27012	55760	0.008	0.963	0.143	0.379	0.088	0.967	0.995	0.000
(0.700,0.800]	36581	157788	0.025	0.952	0.207	0.252	0.175	0.970	0.981	0.002
(0.600,0.700]	66486	466395	0.070	0.919	0.225	0.170	0.334	0.975	0.940	0.013
(0.500,0.600]	80571	893988	0.152	0.850	0.199	0.123	0.525	0.980	0.862	0.047
(0.400,0.500]	76565	1275106	0.266	0.749	0.166	0.094	0.708	0.986	0.750	0.116
(0.300,0.400]	49666	1280417	0.378	0.645	0.142	0.077	0.826	0.990	0.638	0.203
(0.200,0.300]	35260	1512681	0.509	0.520	0.118	0.063	0.910	0.994	0.506	0.318
(0.100,0.200]	24062	1968794	0.677	0.356	0.096	0.051	0.967	0.996	0.333	0.481
(0.000,0.100]	13660	3805317	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.811

Two Class Neural Network

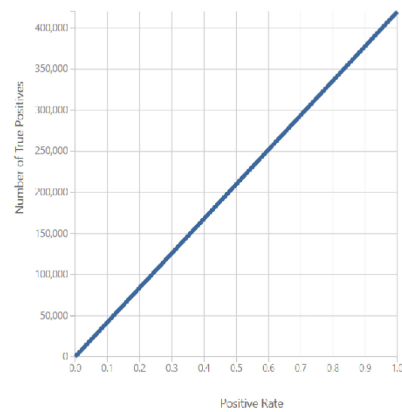
ROC



Precision/Recall



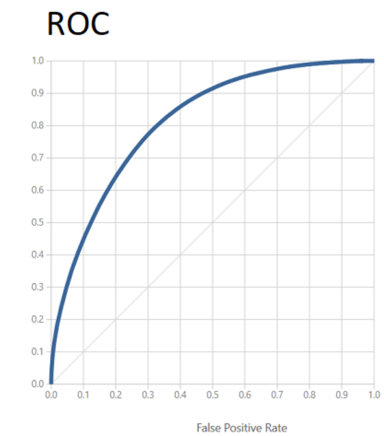
Lift

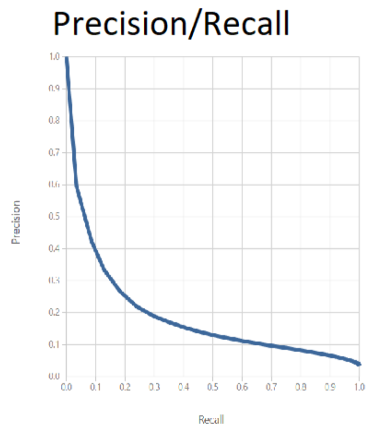


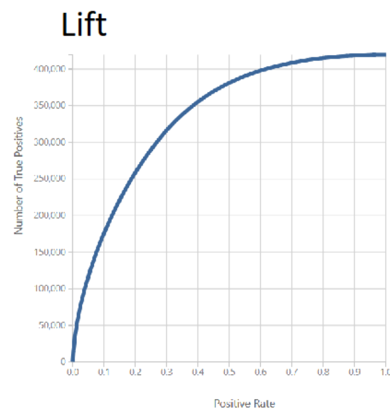
True Positive	False Negative	Accuracy	Precision	Threshold	AUC
0	419789	0.965	1.000	0.5	0.500
False Positive	**True Negative**	**Recall**	**F1 Score**		
0	11420969	0.000	0.000		
Positive Label	**Negative Label**				
1	0				

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	0	0	0.000	0.965	0.000	1.000	0.000	0.965	1.000	0.000
(0.800,0.900]	0	0	0.000	0.965	0.000	1.000	0.000	0.965	1.000	0.000
(0.700,0.800]	0	0	0.000	0.965	0.000	1.000	0.000	0.965	1.000	0.000
(0.600,0.700]	0	0	0.000	0.965	0.000	1.000	0.000	0.965	1.000	0.000
(0.500,0.600]	0	0	0.000	0.965	0.000	1.000	0.000	0.965	1.000	0.000
(0.400,0.500]	0	0	0.000	0.965	0.000	1.000	0.000	0.965	1.000	0.000
(0.300,0.400]	0	0	0.000	0.965	0.000	1.000	0.000	0.965	1.000	0.000
(0.200,0.300]	0	0	0.000	0.965	0.000	1.000	0.000	0.965	1.000	0.000
(0.100,0.200]	0	0	0.000	0.965	0.000	1.000	0.000	0.965	1.000	0.000
(0.000,0.100]	419789	11420969	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.500

Two Class Boosted Decision Tree



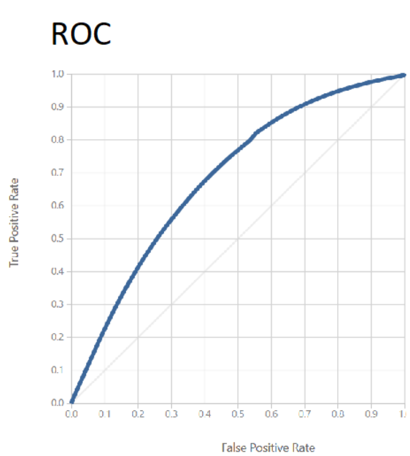


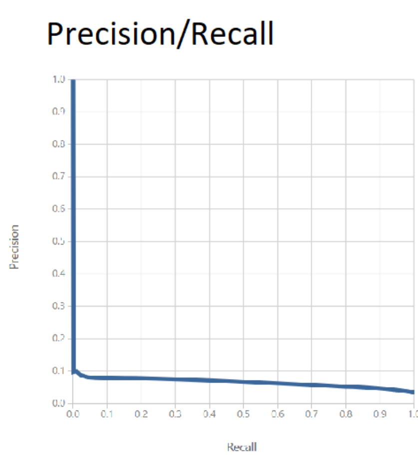


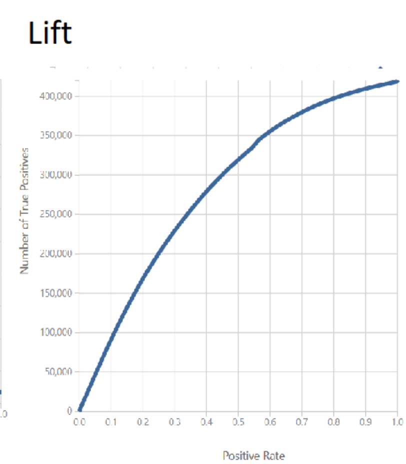
True Positive	False Negative	Accuracy	Precision	Threshold	AUC
419743	46	0.075	0.037	0.5	0.814
False Positive	**True Negative**	**Recall**	**F1 Score**		
10950532	470437	1.000	0.071		
Positive Label	**Negative Label**				
1	0				

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	406153	7577622	0.674	0.359	0.097	0.051	0.968	0.996	0.337	0.481
(0.800,0.900]	9934	1746829	0.823	0.212	0.082	0.043	0.991	0.998	0.184	0.631
(0.700,0.800]	2746	1058720	0.912	0.123	0.075	0.039	0.998	0.999	0.091	0.723
(0.600,0.700]	906	560881	0.960	0.076	0.071	0.037	1.000	1.000	0.042	0.772
(0.500,0.600]	4	6480	0.960	0.075	0.071	0.037	1.000	1.000	0.041	0.772
(0.400,0.500]	0	1605	0.960	0.075	0.071	0.037	1.000	1.000	0.041	0.773
(0.300,0.400]	28	57011	0.965	0.070	0.071	0.037	1.000	1.000	0.036	0.778
(0.200,0.300]	17	230790	0.985	0.051	0.070	0.036	1.000	1.000	0.016	0.798
(0.100,0.200]	1	181031	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.814
(0.000,0.100]	0	0	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.814

Two Class Averaged Perceptron





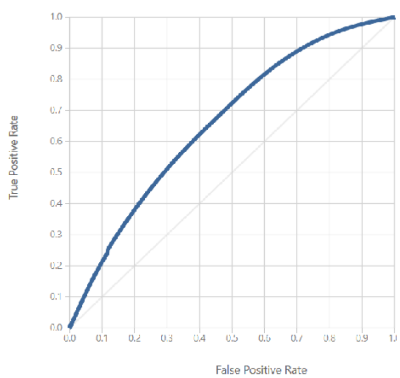


True Positive	False Negative	Accuracy	Precision	Threshold	AUC
419789	0	0.035	0.035	0.5	0.685
False Positive	**True Negative**	**Recall**	**F1 Score**		
11420969	0	1.000	0.068		
Positive Label	**Negative Label**				
1	0				

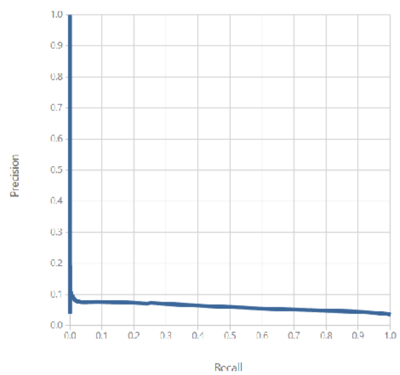
Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	418905	11390335	0.997	0.038	0.069	0.035	0.998	0.972	0.003	0.682
(0.800,0.900]	884	30633	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.685
(0.700,0.800]	0	1	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.685
(0.600,0.700]	0	0	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.685
(0.500,0.600]	0	0	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.685
(0.400,0.500]	0	0	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.685
(0.300,0.400]	0	0	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.685
(0.200,0.300]	0	0	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.685
(0.100,0.200]	0	0	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.685
(0.000,0.100]	0	0	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.685

Two Class Bayes Point Machine

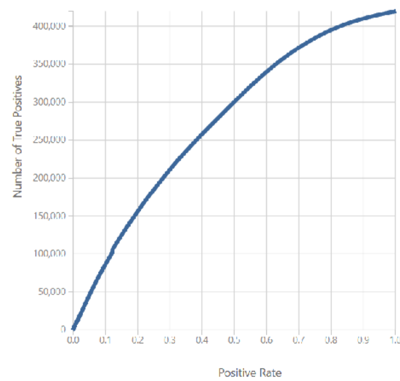
ROC



Precision/Recall



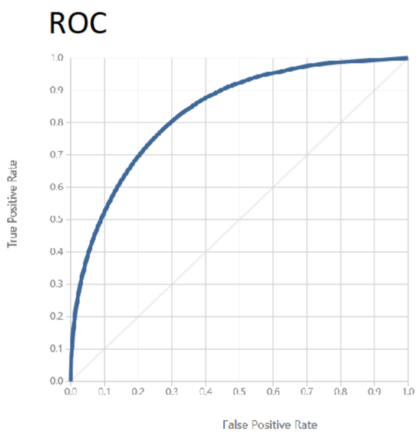
Lift

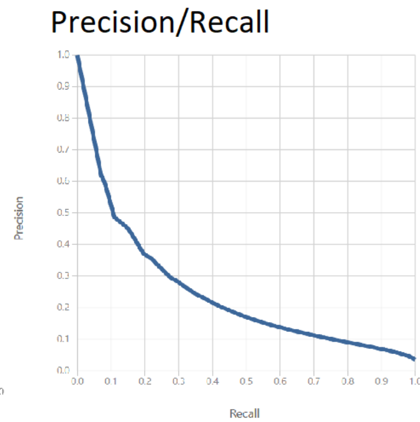


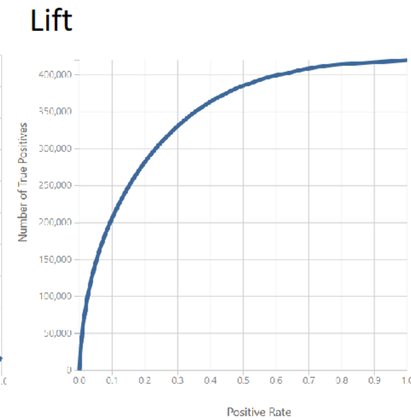
True Positive	False Negative	Accuracy	Precision	Threshold	AUC
101542	318247	0.860	0.071	0.5	0.660
False Positive	**True Negative**	**Recall**	**F1 Score**		
1334277	10086692	0.242	0.109		
Positive Label	**Negative Label**				
1	0				

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	241	2225	0.000	0.964	0.001	0.098	0.001	0.965	1.000	0.000
(0.800,0.900]	3758	38140	0.004	0.961	0.017	0.090	0.010	0.965	0.996	0.000
(0.700,0.800]	13572	172057	0.019	0.948	0.054	0.076	0.042	0.965	0.981	0.000
(0.600,0.700]	30091	363985	0.053	0.920	0.091	0.076	0.114	0.967	0.950	0.003
(0.500,0.600]	57540	757870	0.122	0.861	0.113	0.073	0.251	0.970	0.883	0.015
(0.400,0.500]	83004	1518563	0.257	0.740	0.109	0.062	0.448	0.974	0.750	0.062
(0.300,0.400]	111328	2745372	0.498	0.517	0.095	0.051	0.714	0.980	0.510	0.203
(0.200,0.300]	110647	4670803	0.902	0.132	0.074	0.038	0.977	0.992	0.101	0.560
(0.100,0.200]	9607	1151885	1.000	0.035	0.068	0.035	1.000	0.986	0.000	0.660
(0.000,0.100]	1	69	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.660

Two Class Decision Forest





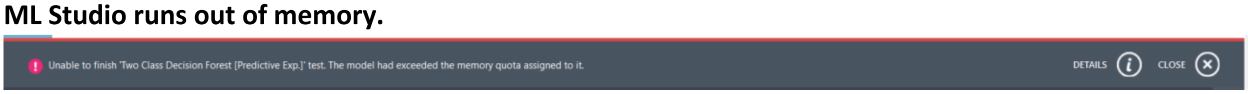


True Positive	False Negative	Accuracy	Precision	Threshold	AUC
263269	156520	0.838	0.130	0.5	0.835
False Positive	**True Negative**	**Recall**	**F1 Score**		
1756663	9664306	0.627	0.216		
Positive Label	**Negative Label**				
1	0				

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	75407	113262	0.016	0.961	0.248	0.400	0.180	0.970	0.990	0.001
(0.800,0.900]	53062	224312	0.039	0.947	0.290	0.276	0.306	0.974	0.970	0.006
(0.700,0.800]	46280	330606	0.071	0.923	0.277	0.207	0.416	0.978	0.941	0.017
(0.600,0.700]	47114	494898	0.117	0.885	0.246	0.160	0.529	0.981	0.898	0.037
(0.500,0.600]	43965	637274	0.174	0.835	0.214	0.129	0.633	0.984	0.842	0.070
(0.400,0.500]	37500	723317	0.239	0.777	0.187	0.107	0.723	0.987	0.779	0.113
(0.300,0.400]	34881	923805	0.320	0.702	0.161	0.089	0.806	0.990	0.698	0.175
(0.200,0.300]	28956	1079686	0.413	0.613	0.138	0.075	0.875	0.992	0.604	0.254
(0.100,0.200]	25108	1527366	0.544	0.486	0.114	0.061	0.934	0.995	0.470	0.376
(0.000,0.100]	27516	5366443	1.000	0.035	0.068	0.035	1.000	1.000	0.000	0.835

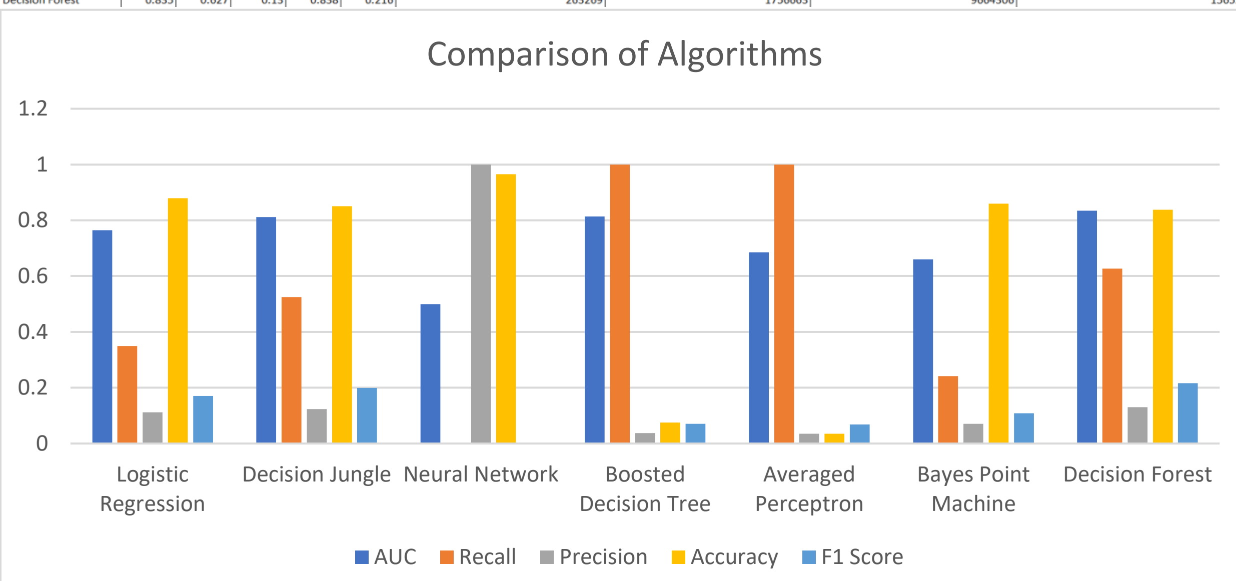
Comparative Analysis of the 7 Classification algorithms

- All the classifier characteristics have been shown at a threshold of 0.5. These values differ when the threshold changes, but we wanted to keep the threshold as standard 0.5 for the purpose of comparison of different models.
- From the above comparisons of different classifier characteristics, we can conclude that the **Two Class Decision Forest** was the best classifier among all the other algorithms. It had the highest AUC value, the highest Recall as well as Precision but we run into a technical difficulty when we try to deploy the algorithm. **When this model is deployed as a Web Service, Microsoft Azure ML Studio runs out of memory.**



- **The second best model** that we can see is the **Decision Jungle**. This has good scores for the AUC, Recall (0.525) and Precision(0.123) and a pretty good accuracy of 0.85. This model tries to increase the True Positives while not increasing too many false positives.
- **Logistic Regression** seems to be the **third best model** that we could apply on this dataset.
- The fourth best algorithm which we can apply for this dataset for classification is Bayes Point Machine.
- We will not recommend using the below algorithms at all
- Although Boosted Decision tree gives very good results for True Positives, it classifies too many Non-Delinquents as Delinquent. This model would end up becoming a costly affair for Freddie Mac because they would need to perform more scrutiny of loans for which it's not required. So although a Recall of 1 seems very high, the precision is very less (0.037). We can observe the same pattern of results for Averaged Perceptron. Also, the Neural Network performs the worst among all and it ends up classifying all the Loans as Non-Delinquent.

						Classified Delinquents		Classified Non Delinquents	
Algorithm	AUC	Recall	Precision	Accuracy	F1 Score	True Positive (Properly classified)	False Positive (Improperly classified)	True Negative (Properly classified)	False Negative (Improperly Classified)
Logistic Regression	0.765	0.349	0.112	0.879	0.17	146629	1159202	10261767	273160
Decision Jungle	0.811	0.525	0.123	0.85	0.199	220576	1578654	9842315	199213
Neural Network	0.5	0	1	0.965	0	0	0	11420969	419789
Boosted Decision Tree	0.814	1	0.037	0.075	0.071	419743	10950532	470437	46
Averaged Perceptron	0.685	1	0.035	0.035	0.068	419789	11420369	0	0
Bayes Point Machine	0.66	0.242	0.071	0.86	0.109	101542	1334277	10086692	318247
Decision Forest	0.835	0.627	0.13	0.838	0.216	263269	1756663	9664306	156520



Web Application

The web application has been developed using Flask and has been deployed using the Cloud Foundry CLI on IBM Bluemix. The application is hosted on the URL :

http://assignment3-precocious-wristband.mybluemix.net

To replicate this setup

Download the application files related to Flask available on github to a folder on local system. You will need to have Cloud Foundry CLI installed in your system. Please refer to the following link for more details.

After this, you will have to login using the command -- cf login

Once this is done, you can cd into this folder and enter the command -- cf push

```
C:\Users\visha\Desktop\MSIS\Advanced Data Science\Assignments\Assignment3\Flask\FlaskApplication>cf push
Using manifest file C:\Users\visha\Desktop\MSIS\Advanced Data Science\Assignments\Assignment3\Flask\FlaskApplication\manifest.yml

Updating app Assignment3 in org satam.v@husky.neu.edu / space DataSciX as satam.v@husky.neu.edu...
OK

Uploading Assignment3...
Uploading app files from: C:\Users\visha\Desktop\MSIS\Advanced Data Science\Assignments\Assignment3\Flask\FlaskApplication
Uploading 78.7K, 14 files
Done uploading
OK

Starting app Assignment3 in org satam.v@husky.neu.edu / space DataSciX as satam.v@husky.neu.edu...
Downloading dotnet-core_v1_0_20-20170620-1449...
Downloading php_buildpack...
Downloading xpages_buildpack...
Downloading noop-buildpack...
Downloading java_buildpack...
Downloaded php_buildpack
Downloading ruby_buildpack...
```

```
App started


OK

App Assignment3 was started using this command `python webApp.py`

Showing health and status for app Assignment3 in org satam.v@husky.neu.edu / space DataSciX as satam.v@husky.neu.edu...
OK

requested state: started
instances: 1/1
usage: 128M x 1 instances
urls: assignment3-precocious-wristband.mybluemix.net
last uploaded: Thu Aug 3 12:57:24 UTC 2017
stack: cflinuxfs2
buildpack: python 1.5.15

     state     since                    cpu    memory       disk       details
#0   running   2017-08-03 08:58:38 AM   0.0%   0 of 128M    0 of 1G
```

The api keys for the REST services have been deliberately removed from the apikey.json file. Please contact the owner of the repository for access to the REST api's which exist on Microsoft Azure Machine Learning Studio. This has been done for security reasons.

You can manually edit and add the api keys in the apikeys.json file located in the application files before pushing the application on IBM Bluemix.

The URL is also sourced from this configuration file so that no API parameters are hardcoded in the application files.

static	8/3/2017 5:41 PM	File folder	
templates	7/28/2017 12:43 PM	File folder	
.cfignore	6/19/2017 10:41 A...	CFIGNORE File	1 KB
.gitignore	6/19/2017 10:41 A...	Text Document	1 KB
apikeys.json	8/3/2017 5:39 PM	JSON File	1 KB
LICENSE	6/19/2017 10:41 A...	File	12 KB
manifest.yml	7/31/2017 8:47 PM	YML File	1 KB
Procfile	7/31/2017 8:46 PM	File	1 KB
requirements.txt	6/19/2017 10:41 A...	Text Document	1 KB
setup.py	6/19/2017 10:41 A...	Python File	1 KB
webApp.py	8/3/2017 5:59 PM	Python File	11 KB


```
{
    "classification":{
        "decisionjungle":{
            "apikey":"MyFUNy[redacted]x8tiuvF+4w==",
            "url":"
            https://ussouthcentral.services.azureml.net/workspaces/d7b83ef94e7c4da5ab1a77ebdd7253bd/services/f42e205489df40f5a468cba9113648f5/execute?api-version=2.0&details=true"
        },
        "logisticregression":{
            "apikey":"Hw8UN8W[redacted]6maxvh5g==",
            "url":"
            https://ussouthcentral.services.azureml.net/workspaces/d7b83ef94e7c4da5ab1a77ebdd7253bd/services/1d0bdee381b14b7d899a4b728836cbd7/execute?api-version=2.0&details=true"
        },
        "bayestwopoint":{
            "apikey":"KGMcXnj[redacted]6ccCcfw==",
            "url":"
            https://ussouthcentral.services.azureml.net/workspaces/d7b83ef94e7c4da5ab1a77ebdd7253bd/services/c86d4108d8c541abbb35b23bc341ee19/execute?api-version=2.0&details=true"
        }
    },
    "prediction":{
        "boosteddecisiontree":{
            "apikey":"4/1AxmSv[redacted]/Tkt3g==",
            "url":"
            https://ussouthcentral.services.azureml.net/workspaces/43247f6706b64e68a8b7d22ff16a2a4f/services/89403771d86f46e887c4d27b6eb80908/execute?api-version=2.0&details=true"
        },
        "linearregression":{
            "apikey":"cz3t27xi[redacted]XfX2Q==",
            "url":"
            https://ussouthcentral.services.azureml.net/workspaces/43247f6706b64e68a8b7d22ff16a2a4f/services/1f553c369b314b9dad2097ff5c012200/execute?api-version=2.0&details=true"
        },
        "neuralnetwork":{
            "apikey":"A2fylieT[redacted]j/8g==",
            "url":
            "https://ussouthcentral.services.azureml.net/workspaces/43247f6706b64e68a8b7d22ff16a2a4f/services/964b03c9c6fa480ca1156bd093eb9ff4/execute?api-version=2.0&details=true"
        }
    }
}
```


This application is active and running and supports the following resource identifiers.

GET method URLs

/prediction

/classification

The controller for this resource doesn't accept any parameters.

POST method URLs (The URLs to get values from the Machine Learning REST API from Microsoft Azure)

/prediction/getPrediction

/classification/getClassification

The application accepts parameters in the JSON format. These should be passed from the UI using Javascript. The returned value for the Prediction/Classification is sent as a JSON string.

Parameters to be passed for POST requests –

Prediction

```
{
    "algoType":<algo_type>,
    "credit_score": <credit score (from 301 - 850), numeric>,
    "og_upb":<orig upb numeric>,
    "og_first_time_home_buyer":<first time home buyer Y,N,X>,
    "og_loan_term":<orig loan term, numeric>,
    "og_quarter_year":<quarter followed by year, string example: Q12004>,
    "og_seller_name":<seller_name, string>,
    "og_servicer_name":<servicer name, string>
}
```

The algo_type for Prediction takes the following values

- pred_lr – Linear Regression
- pred_df – Boosted Decision Tree
- pred_nn – Neural Network

JSON returned by the POST URL :

{"predicted_interest_rate":predicted_interest_rate}

In any error occurs,

{"predicted_interest_rate":"Some error occured"}

Classification

```
{
    "algoType":<algo_type>,
    "curr_act_upb": <current actual upb, numeric>,
    "loan_age":<loan age, numeric>,
    "months_to_legal_maturity":<months to legal maturity, numeric>,
    "crr_interest_rate":<current interest rate, numeric>,
    "curr_deferred_upb":<current deferred upb, numeric>
}
```

The algo_type for Classification takes the following values

- pred_lr – Logistic Regression
- pred_df – Decision Jungle
- pred_nn – Bayes Point Classification

JSON returned by the POST URL :

{"classified_as":classified_as,"scored_probability":<probability>}

In any error occurs,

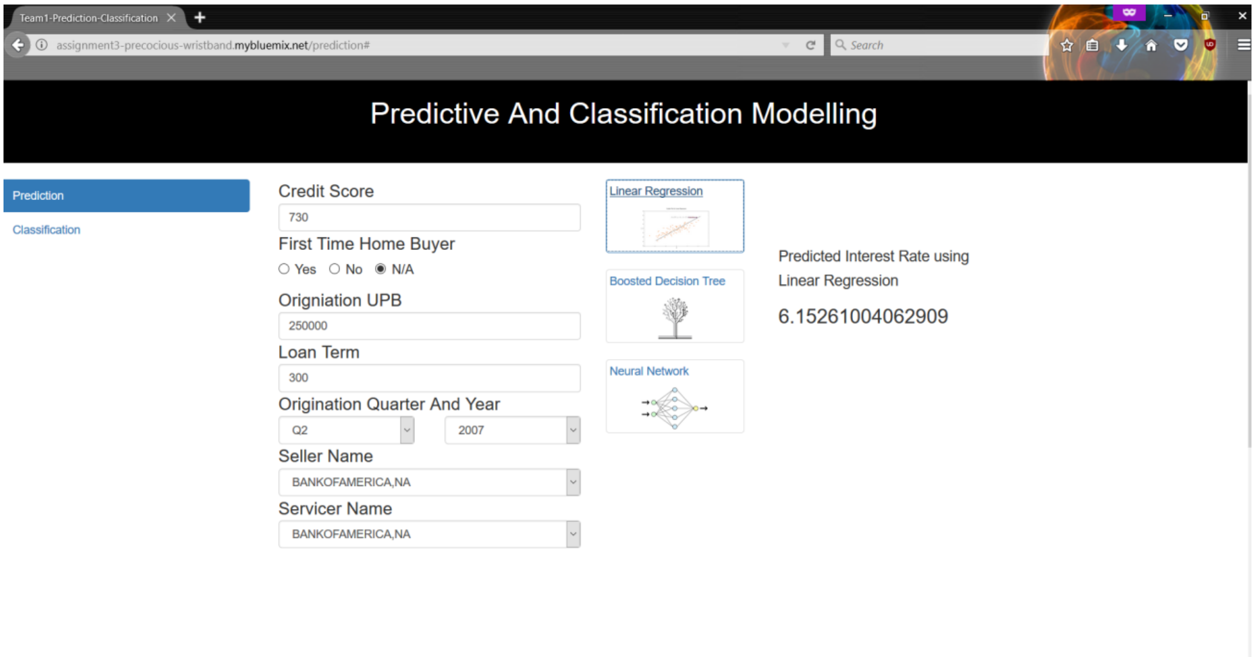
{"classified_as":"Some Error occured in Classification","scored_probability":""}

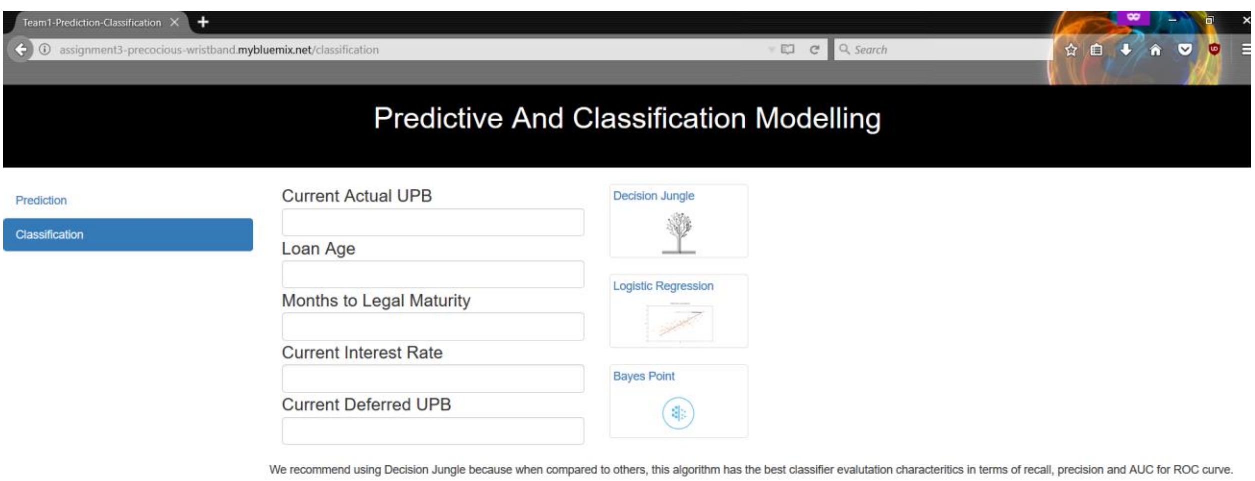
User Interface

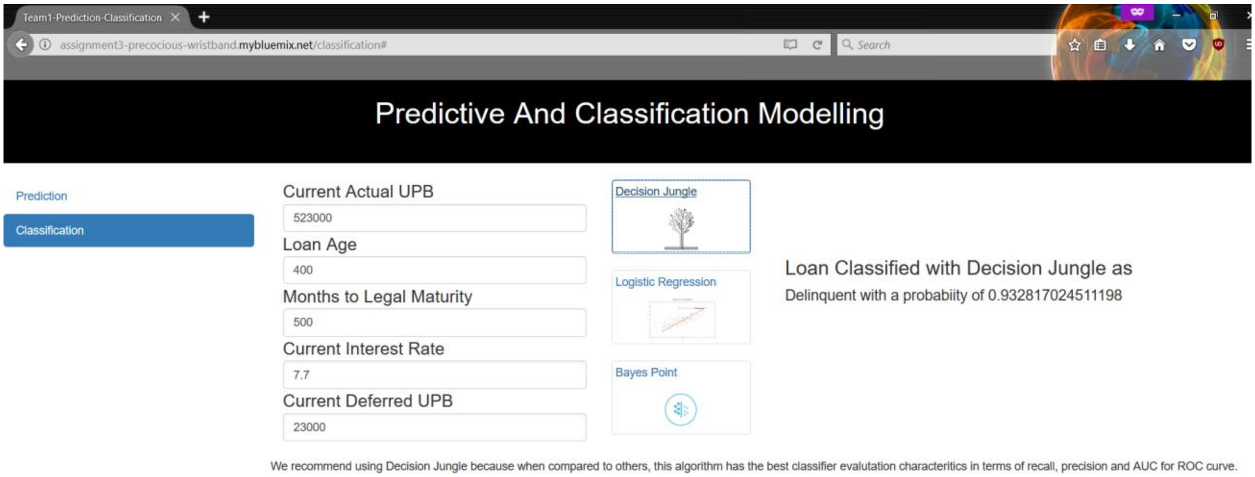
The user interface for the web application has been built using HTML, Bootstrap, JQuery, Javascript and CSS. Here are the screenshots of the UI.

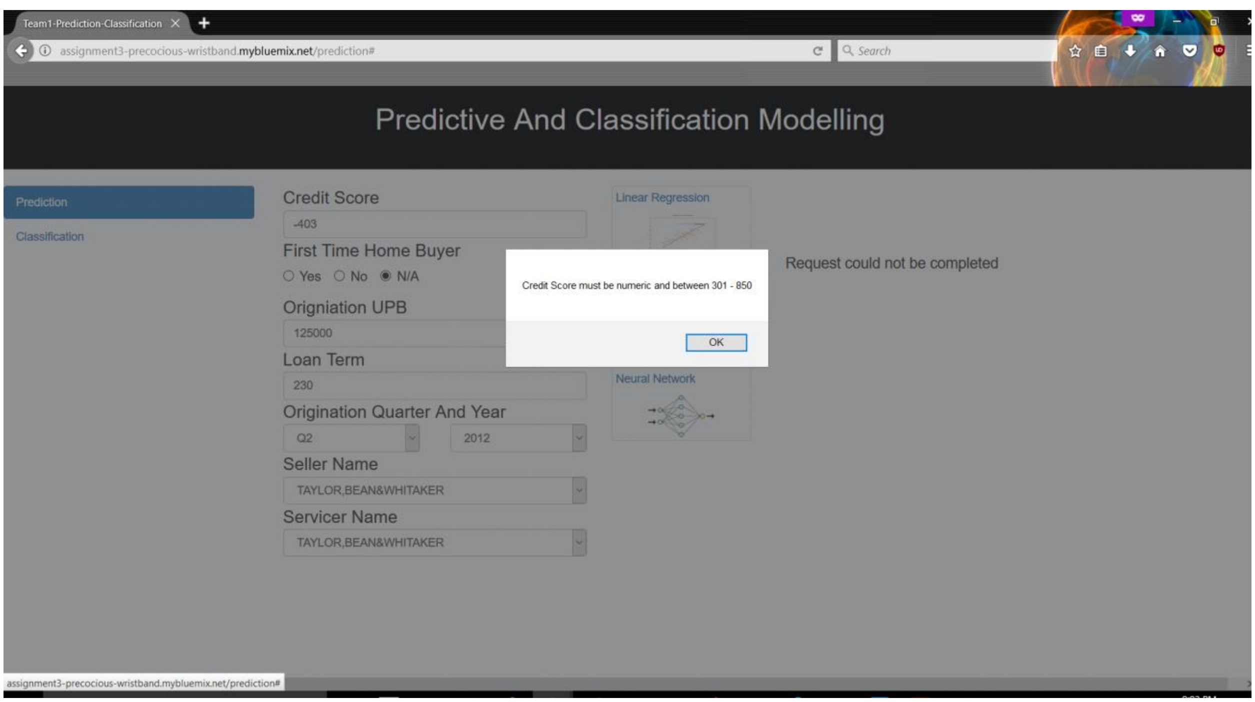
Javascript validations have been added to ensure that the user sends valid data. characters are not allowed in the numeric fields and the Credit Score has to be in the range of 301 – 850.

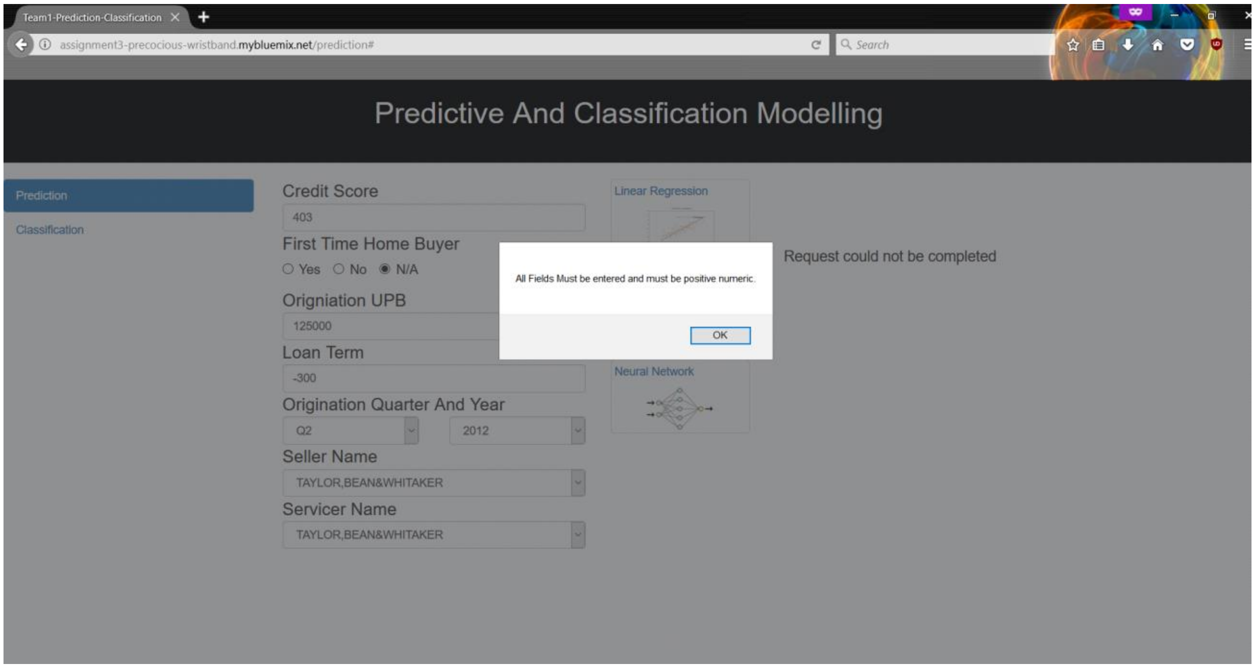
The controller URLs are called using AJAX and the results are updated on the UI.






Team1-Prediction-Classification
assignment3-precocious-wristband.mybluemix.net/classification#
Predictive And Classification Modelling
Prediction
Classification
Current Actual UPB
523000
Loan Age
400
Months to Legal Maturity
500
Current Interest Rate
7.7
Current Deferred UPB
23000
Decision Jungle
Logistic Regression
Bayes Point
Loan Classified with Decision Jungle as
Delinquent with a probabiity of 0.932817024511198
We recommend using Decision Jungle because when compared to others, this algorithm has the best classifier evalutation characteritics in terms of recall, precision and AUC for ROC curve.


Team1-Prediction-Classification
assignment3-precocious-wristband.mybluemix.net/prediction#
Predictive And Classification Modelling
Prediction
Classification
Credit Score
-403
First Time Home Buyer
Yes
No
N/A
Orignation UPB
125000
Loan Term
230
Origination Quarter And Year
Q2
2012
Seller Name
TAYLOR,BEAN&WHITAKER
Servicer Name
TAYLOR,BEAN&WHITAKER
Linear Regression
Neural Network
Credit Score must be numeric and between 301 - 850
OK
Request could not be completed



Microsoft Azure Machine Learning Studio to build REST Services

Microsoft Azure Machine Learning Studio has been used as the tool for creating the 6 REST services. We have created 3 services for Prediction of Interest Rates in the Freddie Mac's dataset and 3 services for Classification. These are available via the api key and the url provided by Microsoft Azure. These are the underlying backend and the heart of the system. The machine learning models have been trained and are available on Microsoft Azure. We can pass certain input parameters to these services and get the prediction and/or classification results.

Prediction

Parameters to be passed

JSON dictionary as shown below for the body

```
data =  {

        "Inputs": {

                "input1":
                {
                    "ColumnNames": ["CREDIT_SCORE", "FIRST_HOME_BUYER_FLAG", "OG_UPB", "OG_LOANTERM", "SELLER_NAME", "SERVICE_NAME", "OG_QUARTERYEAR"],
                    "Values": [ [credit_score,og_first_time_home_buyer,og_upb,og_loan_term,og_seller_name,og_servicer_name,og_quarter_year]]
                },
            },          },
            "GlobalParameters": {
}
    }
headers = {'Content-Type':'application/json', 'Authorization':('Bearer '+ api_key)}
```

The predicted interest rate is returned by the services and can be accessed as

```
predicted_interest_rate=response_json['Results']['output1']['value']['Values'][0][7]
```

Classification

Parameters to be passed

JSON dictionary as shown below for the body

```
data =  {

        "Inputs": {

                "input1":
                {
                    "ColumnNames": ["CUR_ACT_UPB", "LOAN_AGE", "MONTHS_LEGAL_MATURITY", "CURR_INTERESTRATE", "CURR_DEF_UPB"],
                    "Values": [[curr_act_upb, loan_age, months_to_legal_maturity, curr_interest_rate, curr_deferred_upb]]
                },         },
            "GlobalParameters": {
}
    }
```

The classifcation results are obtained as follows

Classification : response_json['Results']['output1']['value']['Values'][0][5]

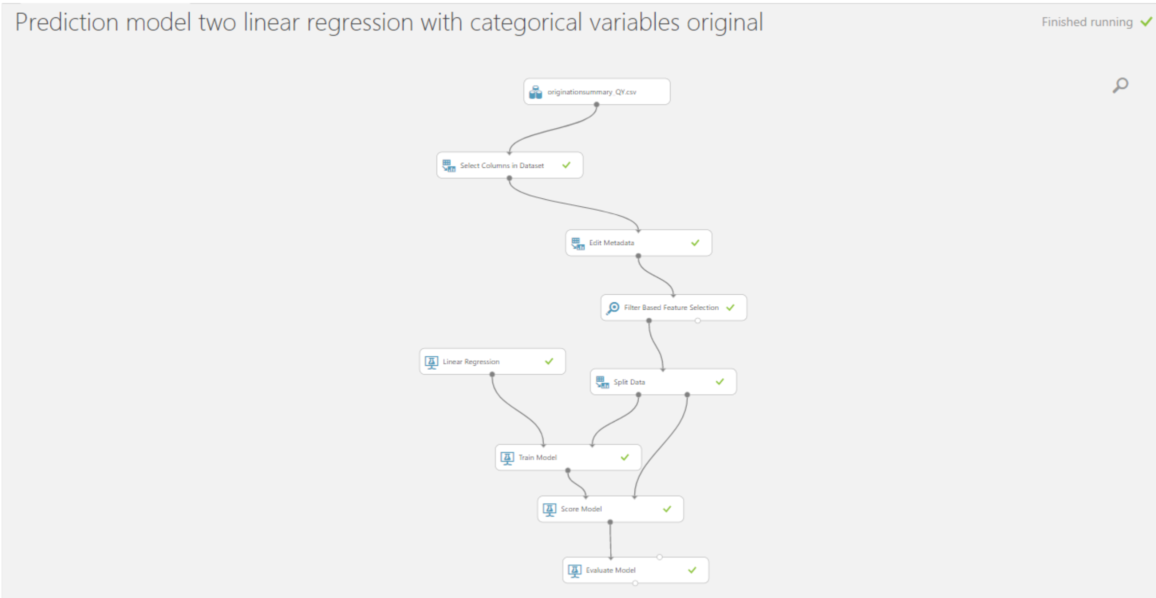
0 = Non-Delinquent   1 = Delinquent

```
scored_probability=response_json['Results']['output1']['value']['Values'][0][6]
```

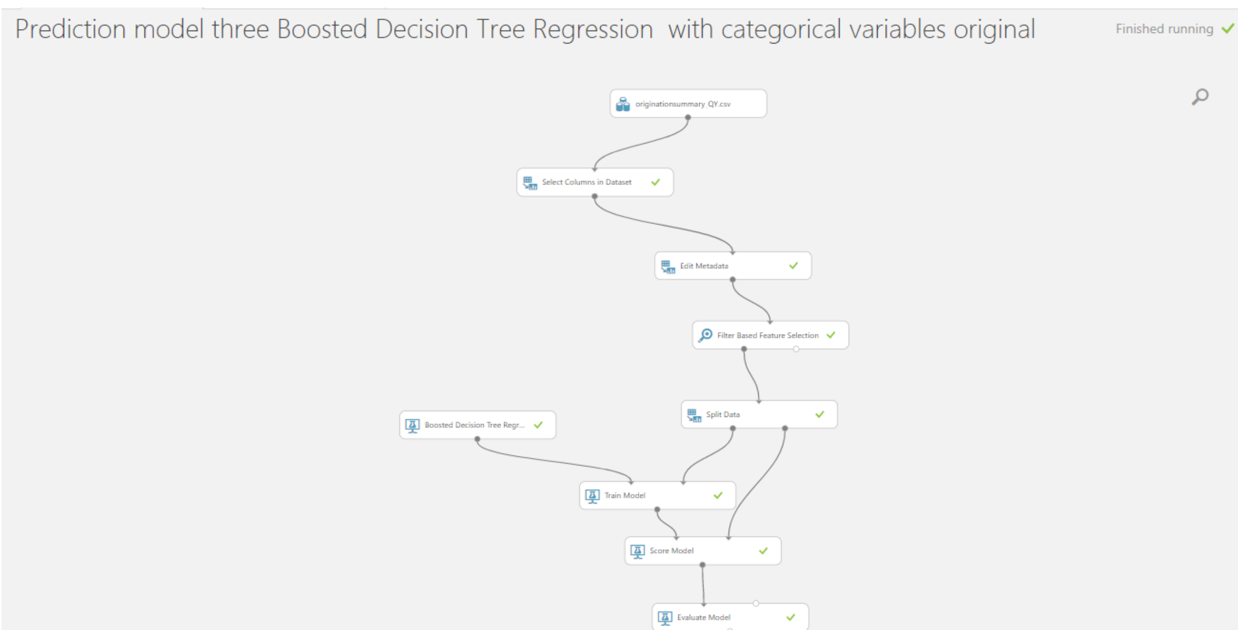
Web Services deployed

Prediction

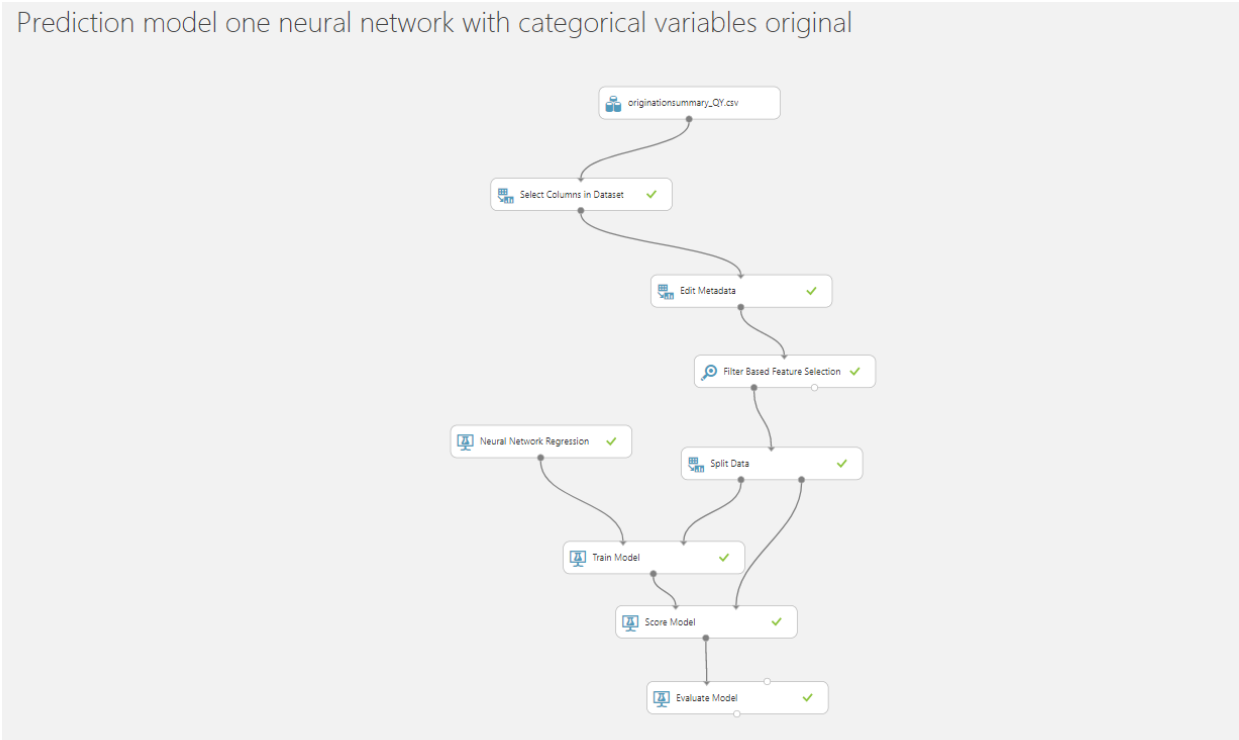
Linear Regression:



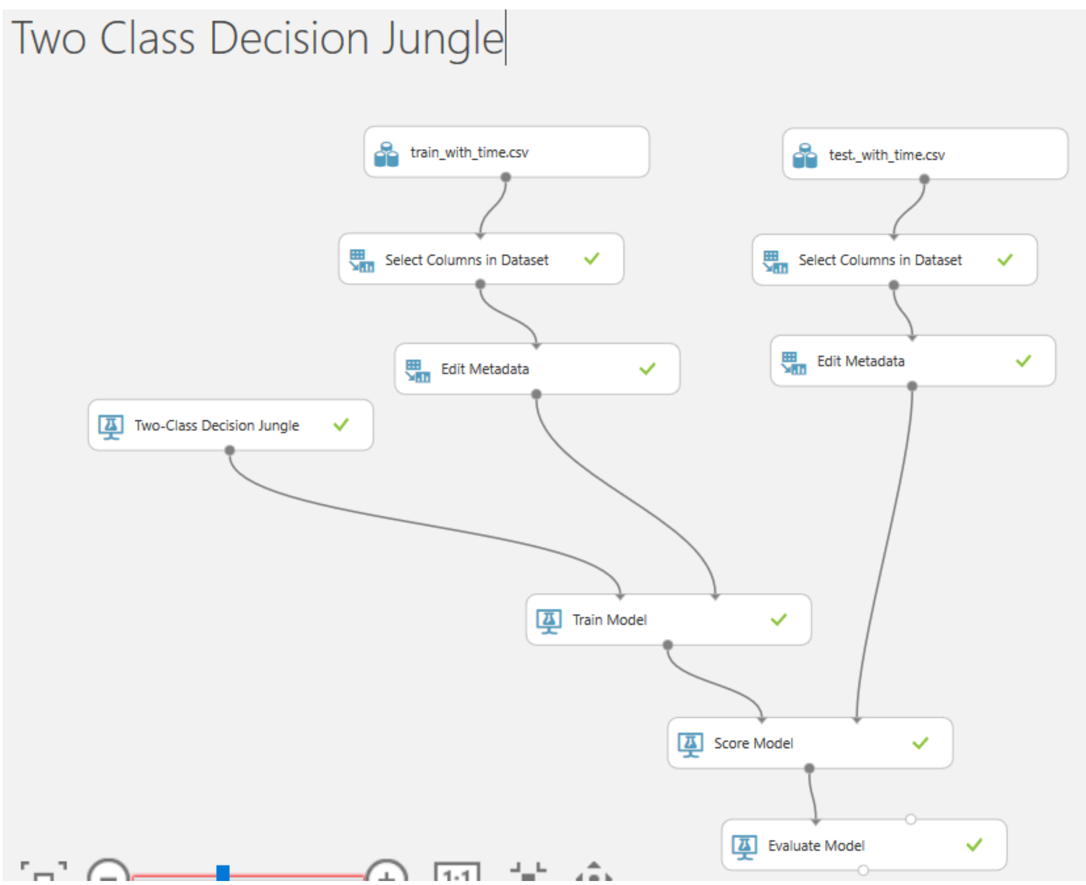
Boosted Decision Tree Regression :



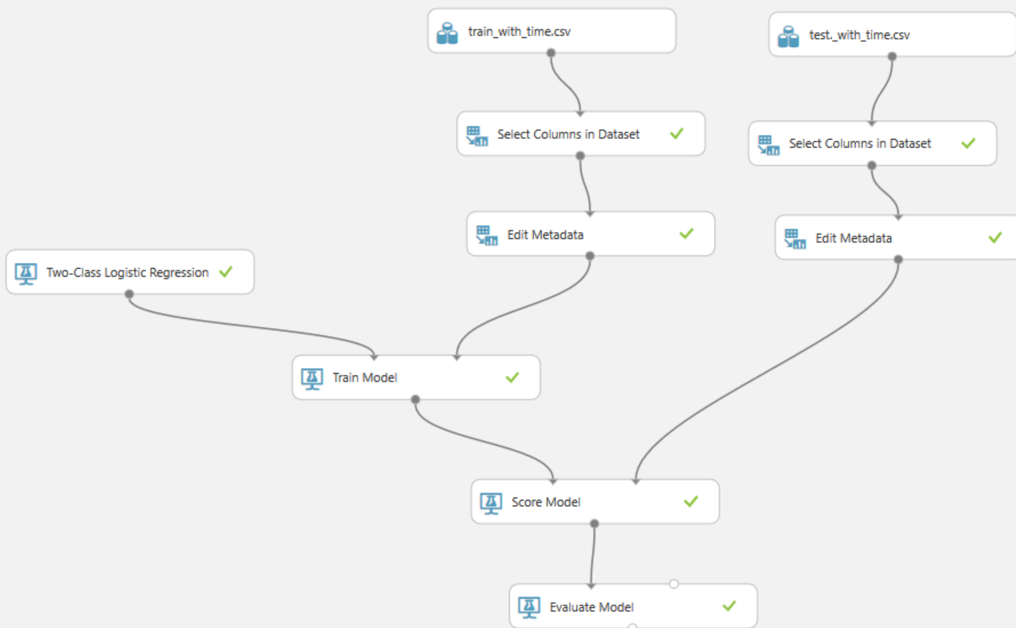
Neural Network:



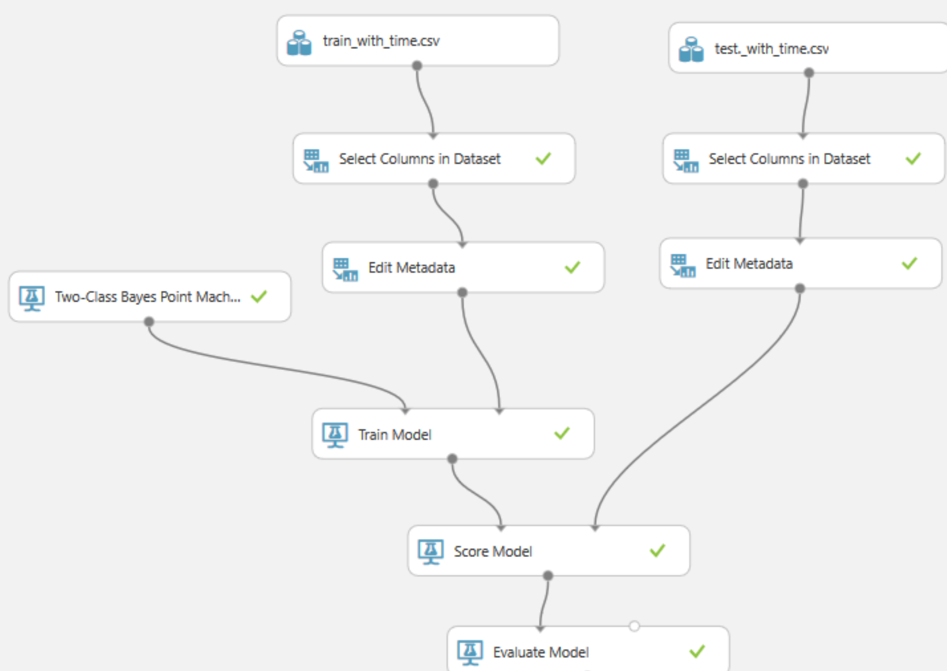
Classification



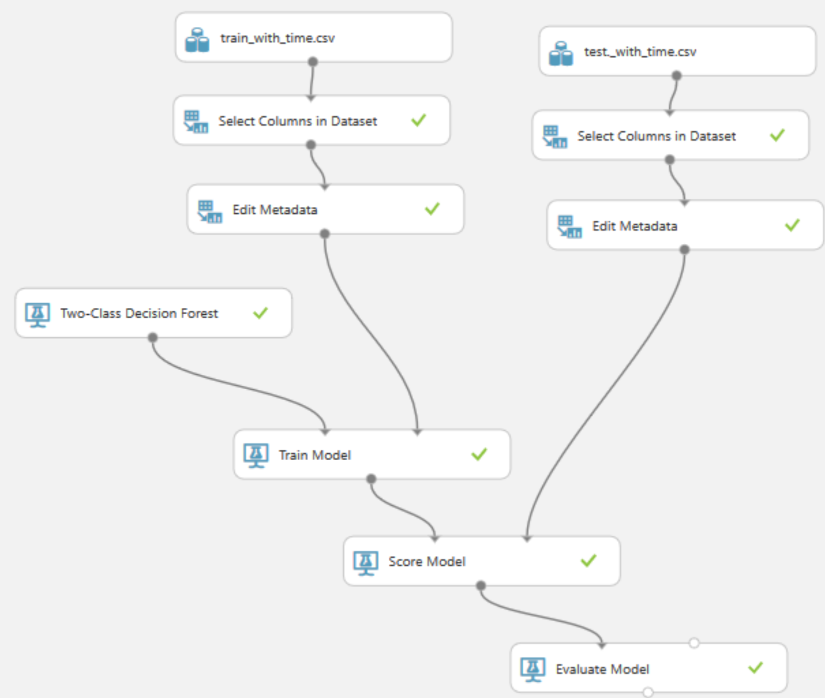
Two Class Logistic Regression



Two Class Bayes Point Machine



Two Class Decision Forest



Testing

Prediction Models

All three models, Linear Regression, Neural Network, Boosted Decision Tree were tested using the deployed WebApp to build a comparative score sheet. Models were tested during the economic boom of years 1999 /2000/2013 , financial crisis o 2007/2008, recovering period of 2009. Refer the following image, the comparison sheet is available on the GitHub repository.

The entire process for computing the predicted values for each model was automated, by calling the api from a function in a Jupyter notebook and outputting the results to a csv file. The jupyter notebook can be found in the github repository.

OG_QUATER_YEAR	CREDITSCORE	FIRST_HOME_BUYER_FLAG	OG_UPB	OG_LOANTERM	SELLER_NAME	SERVICE_NAME	ORIGINAL_INTERESTRATE	PREDICTED_INTERESTRATE		
								LINEAR	NEURAL NETWOF	BOOSTED DECISION
Q1 1999	615	Y	48000	360	Other sellers	Other servicers	5.375	5.788805937	5.945946217	6.164354801
Q1 1999	791	N	284000	312	Other sellers	Other servicers	5.75	5.292569069	5.326989651	5.430260658
Q2 1999	574	N	162000	325	Other sellers	Other servicers	6.875	5.627209263	5.727964878	5.935657501
Q2 1999	716	Y	70000	360	NORWEST MORTGAGE, IN	WELLSFARGOHOMEMORTGA	7.125	5.584598575	6.10819006	6.433392525
Q3 1999	642	N	178000	329	Other sellers	Other servicers	7.125	5.564460412	5.637985706	5.859110355
Q3 1999	630	N	190000	360	CROSSLAND MORTGAGE C	CHASEMTGECO	7.75	5.76729373	5.940724373	6.132706165
Q4 1999	637	X	105000	360	NATLCITYMTGECO	NATLCITYMTGECO	8.625	5.898117836	5.963931561	6.104550362
Q1 2000	734	N	151000	360	Other sellers	CHASEMTGECO	8	5.586177361	5.821092606	5.970065117
Q1 2000	698	N	124000	330	Other sellers	Other servicers	6.25	5.546749613	5.605181694	5.706794739
Q2 2000	768	N	96000	339	Other sellers	Other servicers	7.375	5.527014685	5.551906586	5.656756401
Q2 2000	635	X	78000	360	OLDKENTMTGECO	Other servicers	9.75	5.915954568	6.047593117	6.074911118
Q3 2000	791	N	175000	340	Other sellers	Other servicers	5.875	5.458566044	5.429929733	5.517225742
Q3 2000	676	Y	63000	360	Other sellers	Other servicers	8.75	5.719569409	5.817633152	5.972254753
Q4 2000	787	N	58000	358	BANKOFAMERICA,NA	BANKOFAMERICA,NA	7.25	5.603402578	5.671905518	6.023420334
Q4 2000	786	Y	156000	360	Other sellers	BANKOFAMERICA,NA	7.75	5.54676698	5.380783558	5.467279434
Q1 2005	699	N	253000	360	Other sellers	Other servicers	5.625	5.570011417	5.599218369	5.907714367
Q2 2005	743	N	340000	360	Other sellers	PNCMTGESERVICES,INC	5.875	5.482381573	5.444556236	5.675059795
Q3 2005	708	N	114000	360	WELLSFARGOBANK,NA	WELLSFARGOBANK,NA	6.875	5.668731445	5.716757774	5.781254768
Q4 2005	717	N	120000	360	Other sellers	Other servicers	6.625	5.635469387	5.654280663	5.829120159
Q1 2007	813	N	133000	360	Other sellers	USBANKNA	6.125	5.482502187	5.448886395	5.584337711
Q1 2007	681	Y	271000	360	Other sellers	Other servicers	7.125	5.584667199	5.669298649	5.609924316
Q2 2007	800	N	185000	360	Other sellers	Other servicers	6	5.513394421	5.453591824	5.589690685
Q2 2007	755	N	60000	360	Other sellers	Other servicers	6.125	5.6356771	5.699081898	5.971964836
Q3 2007	700	N	248000	360	USBANKNA	USBANKNA	6.375	5.57390599	5.599033356	5.766508102
Q3 2007	633	N	175000	360	Other sellers	Other servicers	6.125	5.683528291	5.733392715	5.939779758
Q4 2007	781	N	403000	360	Other sellers	USBANKNA	6.375	5.345160966	5.464688301	5.578310013
Q4 2007	696	N	225000	360	Other sellers	Other servicers	6.5	5.590454837	5.612054348	5.823934078
Q1 2008	755	Y	81000	360	Other sellers	Other servicers	5.875	5.630793793	5.648148537	5.691758156
Q1 2008	740	Y	55000	360	Other sellers	Other servicers	5.625	5.661763359	5.7210145	5.793377876
Q2 2008	672	X	190000	360	Other sellers	Other servicers	5.75	5.686066332	5.686931133	5.747159481
Q2 2008	813	N	94000	360	Other sellers	USBANKNA	6	5.506876343	5.495649338	5.695680618
Q3 2008	792	N	88000	360	Other sellers	Other servicers	6.875	5.581868057	5.572535515	5.81924057
Q3 2008	776	N	245000	360	Other sellers	Other servicers	5.875	5.499447923	5.473341465	5.704794884
Q4 2008	733	N	250000	360	Other sellers	Other servicers	6.125	5.53852073	5.545017242	5.775118351
Q4 2008	747	N	150000	360	Other sellers	Other servicers	6.25	5.587279781	5.571796894	5.694217205
Q1 2009	771	N	144000	360	Other sellers	Other servicers	4.875	5.567477447	5.53553772	5.717024326
Q1 2009	732	N	384000	360	Other sellers	CENTRALMTGECO	5.25	5.556713051	5.675244808	5.675155163
Q2 2009	784	N	154000	360	Other sellers	Other servicers	5	5.54847022	5.503624916	5.63553524
Q2 2009	792	N	78000	360	Other sellers	USBANKNA	5.125	5.537484175	5.552968979	5.853200912
Q3 2009	773	N	69000	360	Other sellers	Other servicers	5.5	5.612388141	5.645580769	5.915802956
Q3 2009	723	N	188000	360	Other sellers	Other servicers	5.625	5.587082807	5.585133553	5.775118351
Q4 2009	812	N	99000	360	Other sellers	Other servicers	4.75	5.555366458	5.517090321	5.739845276
Q4 2009	741	N	151000	360	Other sellers	Other servicers	5.5	5.592542854	5.581163406	5.778348446
Q1 2013	723	X	149000	360	Other sellers	Other servicers	3.6	5.661642012	5.632866859	5.464830875
Q1 2013	681	Y	86000	360	Other sellers	Other servicers	3.5	5.700288198	5.772026062	5.836648464
Q2 2013	801	X	102000	360	Other sellers	Other servicers	3.625	5.614471332	5.55034256	5.271491528
Q2 2013	758	X	156000	360	Other sellers	Other servicers	3.5	5.622920199	5.564457417	5.330955029
Q3 2013	793	X	91000	180	BANKOFAMERICA,NA	BANKOFAMERICA,NA	2.875	5.004874178	5.215662003	4.267882347
Q3 2013	751	X	256000	180	BANKOFAMERICA,NA	BANKOFAMERICA,NA	2.875	4.942969104	5.055450916	4.319439888
Q4 2013	720	X	143000	240	BANKOFAMERICA,NA	BANKOFAMERICA,NA	4.625	5.253743968	5.405078411	5.12420845
Q4 2013	715	X	120000	120	BANKOFAMERICA,NA	BANKOFAMERICA,NA	3.25	4.853563757	4.926253796	4.353067398

Edge Case testing

Testing for invalid values, negative values, and range values.

Edge Cases							
Credit Score range	301 - 850						
OG_QUATER_YEAR	CREDITSCORE	FIRST_HOME_BUYER_FL	OG_UPB	OG_LOANTERM	SELLER_NAME	SERVICE_NAME	Output
Q1 2013	256	N	8900000	78	Other sellers	Other servicers	Invalid range for credit score
Q1 2013	956	N	8900000	78	Other sellers	Other servicers	Invalid range for credit score
Negetive Values							
Q1 2013	-567	N	8900000	78	Other sellers	Other servicers	Negetive value error
Q1 2014	566	N	-90000	78	Other sellers	Other servicers	Negetive value error
Q1 2015	-565	N	780000	-90	Other sellers	Other servicers	Negetive value error
Seller Name/Service Name							
The webapp provides a drop down list for all the valid values of Seller names and Service Names							

Testing for Max Length		
	max length	Result
Credit Score	3	UI doesn't allow User to enter
Origination UPB	9	UI doesn't allow User to enter
Loan Term	3	UI doesn't allow User to enter

Classification Models:

Models were tested on actual values from the dataset. The excel sheet for classification testing can be found in the GitHub repository.

Testing for actual values taken from the dataset						Classification					
current upb	loan age	months to legal maturity	current interestrate	current def upb	Actual current loan del sts	Decision Jungle		Logistic Regression		Bayes Point	
						Deliquncy Status	Probability	Deliquncy Status	Probability	Deliquncy Status	Probability
42011.81	43	316	6.875	0	0	0	0.4296	0	0.3543	0	0.3514
130213.4	37	323	6.75	0	1	0	0.4991	0	0.358	0	0.3096
73637.1	20	340	6.75	0	0	0	0.334	0	0.26299	0	0.2622
107339.07	83	277	7.5	0	1	1	0.6752	1	0.638	0	0.4907
124000	2	358	6.75	0	0	0	0.1131	0	0.2177	0	0.1971
116272.52	46	314	6	0	0	0	0.3597	0	0.325	0	0.350
87000	4	356	8.375	0	0	0	0.17	0	0.3375	0	0.202
132000	2	358	7.85	0	1	0	0.1084	0	0.304	0	0.191

Testing Edge Cases and invalid values:

Tested for negative values as none of the features can be negative. Also zero is a valid value for all the fields. Character values are invalid and alerts are provided in the WebApp.

Testing for negetive values					
current upb	loan age	months to legal maturity	current interestrate	current def upb	Actual current loan del sts
-300	9	56	9.8	0	Negetive value error
300	-9	56	9.8	0	Negetive value error
300	9	-56	9.8	0	Negetive value error
300	9	56	-9.8	0	Negetive value error
300	9	56	9.8	-9	Negetive value error
Testing for invalid values					
abc	90	9	0	9	Invalid value.
89999	abc	9	0	9	Invalid value.
89999	90	abc	0	9	Invalid value.
89999	90	9	abc	9	Invalid value.
89999	90	9	0	abc0	Invalid value.

Testing for Max Length		
	max leng	Result
Current UPB	9	UI doesn't allow User to enter
Loan Age maxlength	3	UI doesn't allow User to enter
Months to Legal Maturity	3	UI doesn't allow User to enter
Current Interest Rate	5	UI doesn't allow User to enter
Current Deferred UPB	9	UI doesn't allow User to enter

Summary

This project focused on building machine learning algorithms for prediction and classification using Microsoft Azure and hosting them as a service. The data used is from the Freddie Mac Single Loan Dataset containing Single Family Loan data for years 1999 to 2016 . The aim of this project was to apply prediction algorithms to predict the interest rate based on the given inputs and classification algorithms for classifying a loan as delinquent or non-delinquent. We have used Linear Regression, Boosted Decision Tree and Neural network for prediction and Logistic Regression, Decision Jungle and Bayes Point for Classification. The built models were then hosted using IBM BlueMix Cloud Platform and a Web application was created using flask for users to access the service. We have tested the models for varying data, invalid data, and out of range data using the web application User interface. The test case results are stored in excel sheets and uploaded to git hub. Docker image was build for data sourcing and preprocessing.