2016

DETECTION OF MALICIOUS TWITTER FEEDS

TEAM NAUTILUS

Table of Contents

Detection of Malicious Twitter Feeds

1. Team Nautilus

Team members:

NAME	NUID	SECTION
JAHNVI GANDHI	**001665157**	**2**
KENNETH PEREIRA	**001686410**	**2**
MARIANO GARCIA CLAVERIA	**001222174**	**1**
NUHIYA RAFEEQ	**0016133212**	**1**
PRIYANKA MISHRA	**001666716**	**1**

2. Motivation

Phishing scams are typically fraudulent messages appearing to come from legitimate enterprises (e.g., your university, your Internet service provider, your bank). These messages usually direct you to a spoofed website or otherwise get you to divulge private information (e.g., passphrase, credit card, or other account updates). The perpetrators then use this private information to commit identity theft.

One type of phishing attempt is a tweet message stating you to secure your bank accounts or social profile details, and asking you to "click here" to verify your information. According to Kaspersky Lab, phishing remains a major threat in Russia and the EU as the number of attacks has increased in the region, up 18% to 36.3 million attacks in Q3 2015 compared with the same time period last year.Social networking sites are now a prime target of phishing; they are, in fact, a fantastic depository of information for hackers preparing spear-phishing or whaling campaigns. Detailed job descriptions, connections, friends, and previous employment info are normally widely available and can be used to craft specific and realistic baits. Phishing on social networks is as easy as gathering information from profiles, or it can be done by luring victims into revealing personal information or credentials. Scams like the Facebook Lottery Winning one (April, 2016) have already caused many victims. Phishers might use fake "social network" web site URL addresses with the intention to steal login and password details (or other personal information) and gain access to accounts.

To counterfeit these kind of attacks on social profiles a robust system is required which will run through the data on a real time basis, detect the malicious URL and neutralize them to protect the users from potential scam.

3. Objective

The objective of the project is to build a robust system that would pull the data from the Twitter and scrutinize the URL in the tweets to detect if the URL is malicious or safe. The system possesses the capability to categorize the URL by breaking it down into numerous parameters and uses Machine Learning techniques to classify them.

The main highlight of the project is the dataset fetched from Twitter is large and possesses multiple values along with the URL. A MapReduce program running on a Hadoop Environment will process this entire data and limit the data to URL and its corresponding values. The output of the MapReduce program is then passed through a series of Machine Learning Algorithms to generate if the URL on twitter feeds are legitimate or phishing.

4. Approach

- **a.** Develop a **program in Java** which will extract r**eal time data from Twitter** account.
- **b.** The data extracted from the Twitter Scrapper program will provide as an input to the Hadoop MapReduce program
- **c.** The Hadoop MapReduce program will break the large output extracted into key value pairs of URL and parameter values of the URL
- **d.** The dataset generated by the Hadoop program will serve as input to Machine Learning algorithms
- **e.** A UCI dataset of phishing websites with the similar kind values as the Twitter dataset generated by Hadoop program is used to train and validate the Machine Learning algorithm
- **f.** The datasets were cleaned with the help of a **python** program to remove redundant data from the dataset

g. The datasets are run through Machine Learning algorithms like **Winnow and Genetic Algorithm.** This helps in training the algorithm and calculating the accuracy of the UCI datasets. Once the accuracy of UCI dataset is achieved the program is run through the Twitter dataset extracted and the corresponding target values are generated for individual URL.

h. Azure Machine Learning was used to extensively to test the dataset against multiple models and check the accuracy for the datasets. The Algorithms used in Azure ML are

   i. Two-Class Averaged Perception
   ii. Two-Class Bayes Point Machine
   iii. Two-Class Boosted Decision Tree
   iv. Two-Class Decision Forest
   v. Two-Class Decision Jungle
   vi. Two-Class Locally-Deep Support Vector Machine
   vii. Two-Class Logistic Regression
   viii. Two Class Neural Network
   ix. Two-Class Support Vector Machine

i. The datasets were tested against **Deep Learning algorithms** using ND4J an open-sourced scientific computing library to test and validate UCI datasets and generate the target values for the Twitter datasets

j. A **Multilayer perceptron (MLP)** is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training the network. Our Dataset were tested against Multilayer Perceptron algorithm which resulted out our Url's into the category of Phishing or Legitimate.

5. Tasks Performed

Individuals	Tasks Allotted
1. Jahnvi Gandhi	Machine Learning Algorithms
2. Kenneth Pereira	Machine Learning Algorithm
3. Nuhiya Rafeeq	Machine Learning Algorithms
4. Mariano Garcia Claveria	Twitter Scrapper and MapReduce
5. Priyanka Mishra	MapReduce and ML using Azure

6. Research and Findings

- Motivation : A research conducted in University of Huddersfield, United Kingdom by Rami M Mohammad, Fadi Thabtah and Lee McCluskey. The study involved putting together a set of URL rules which are based on patterns generally observed in phishing URLs. The study discusses about these rules, but does not have instructions on the implementation. Therefore the implementation of the rules is our own approach. Apart from putting together the rules, the team additionally collected data in their own tool as a dataset for Phishing URLs was not published before.
- While trying to fetch the twitter feeds from Twitter Scraper, some information from the feeds were missing as Twitter restricts some its data due to its User Accounts settings
- While working with many algorithms with our dataset and UCI dataset, these were the findings:

❖ Training Dataset.csv (Has four tabs; one for winnow, Logistic Classifier, and GA each; 1 with original dataset). Each tab has the data in the right most tab to suit the formula in professors code

❖ About the dataset : UCI Phishing Dataset:
The dataset can be found using the url below. This dataset has 11055 records. This data has been used to train the model.
https://archive.ics.uci.edu/ml/datasets/Phishing+Websites

❖ USe of Weka to represent graphical views (Original file format is .arff, which is a WEKA file. I've converted it to CSV in R and kept a copy of the .csv file in this same folder)

R script for converting file,

```
library(foreign)
data <- read.arff("Training Dataset.arff")
write.csv(data, "Training Dataset.csv")
```

Total attributes: 31
Total samples/ instances: 11055

Current relation

Relation: phishing
Instances: 11055
Attributes: 31
Sum of weights: 11055

Target variable: 1 and -1 (as seen in below screenshot).
Attributes have values:

1 - Legitimate
0 - Suspicious
-1 - Phishy/Phishing website

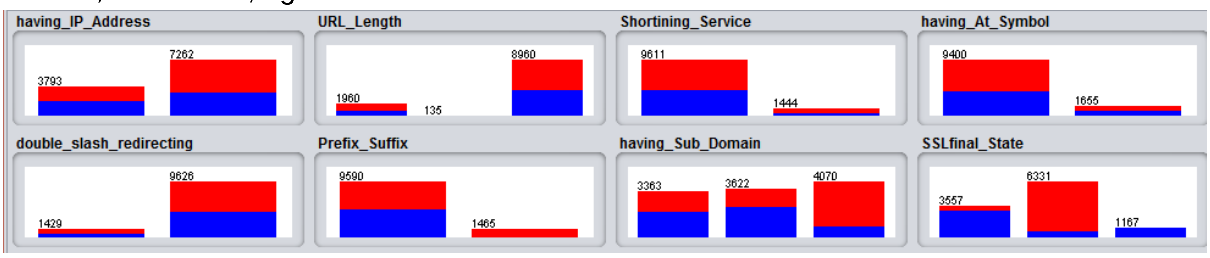
Selected attribute

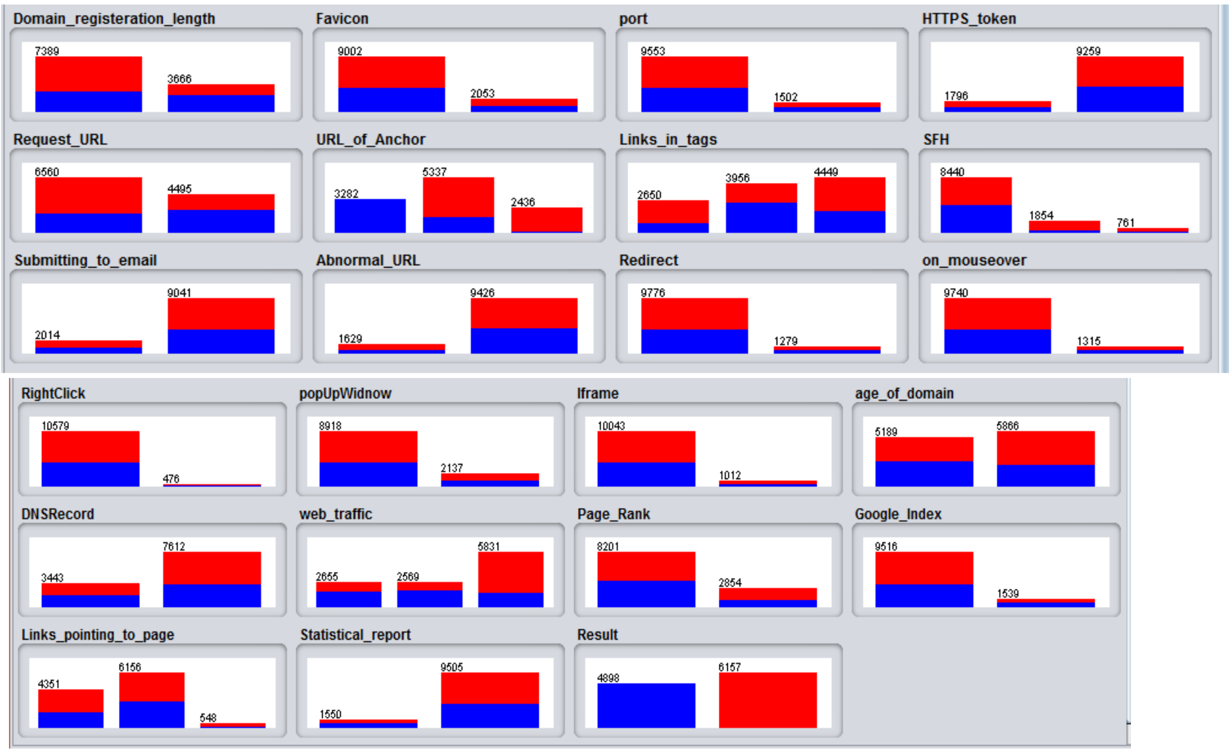
Name: Result
Missing: 0 (0%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

No.	Label	Count	Weight
1	-1	4898	4898.0
2	1	6157	6157.0

Please note Label -1 was changed to Label 0 since some algorithms take positive values

Some visualization to understand the different attributes and their distribution among the samples:
Left is -1, center is 0, right is 1.





7. Obstacles Faced

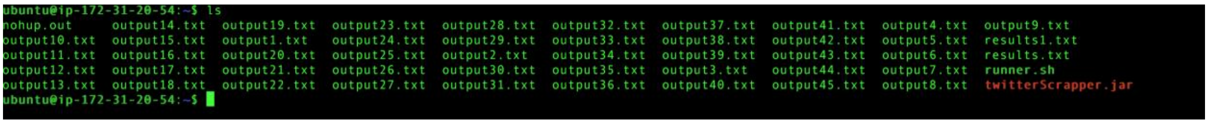
a. Deciding the Machine Learning Algorithms

b. Fetching twitter feeds from **Twitter**, building the **API**

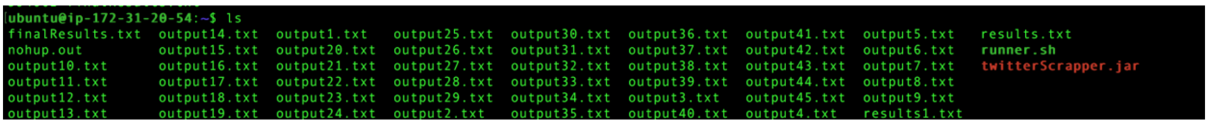
c. Using **Restricted Boltzmann Machines (RBM)** with our dataset

8. Result

a. **Results from Twitter Scraper:**



```
ubuntu@ip-172-31-20-54:~$ ls
nohup.out     output14.txt  output19.txt  output23.txt  output28.txt  output32.txt  output37.txt  output41.txt  output4.txt  output9.txt
output10.txt  output15.txt  output1.txt   output24.txt  output29.txt  output33.txt  output38.txt  output42.txt  output5.txt  results1.txt
output11.txt  output16.txt  output20.txt  output25.txt  output2.txt   output34.txt  output39.txt  output43.txt  output6.txt  results.txt
output12.txt  output17.txt  output21.txt  output26.txt  output30.txt  output35.txt  output3.txt   output44.txt  output7.txt  runner.sh
output13.txt  output18.txt  output22.txt  output27.txt  output31.txt  output36.txt  output40.txt  output45.txt  output8.txt  twitterScrapper.jar
ubuntu@ip-172-31-20-54:~$
```



```
ubuntu@ip-172-31-20-54:~$ ls
finalResults.txt  output14.txt  output1.txt   output25.txt  output30.txt  output36.txt  output41.txt  output5.txt   results.txt
nohup.out         output15.txt  output20.txt  output26.txt  output31.txt  output37.txt  output42.txt  output6.txt   runner.sh
output10.txt      output16.txt  output21.txt  output27.txt  output32.txt  output38.txt  output43.txt  output7.txt   twitterScrapper.jar
output11.txt      output17.txt  output22.txt  output28.txt  output33.txt  output39.txt  output44.txt  output8.txt
output12.txt      output18.txt  output23.txt  output29.txt  output34.txt  output3.txt   output45.txt  output9.txt
output13.txt      output19.txt  output24.txt  output2.txt   output35.txt  output40.txt  output4.txt   results1.txt
```

```
Output URLS: 165122
Waiting 15 minutes to continue
Tweets read: 310968
Output URLS: 165122
ubuntu@ip-172-31-20-54:~$
```

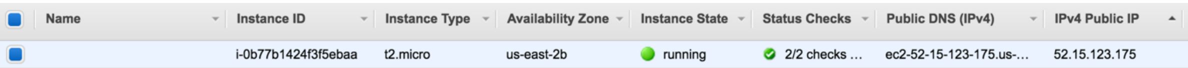
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Public DNS (IPv4)	IPv4 Public IP
	i-0b77b1424f3f5ebaa	t2.micro	us-east-2b	running	2/2 checks ...	ec2-52-15-123-175.us-...	52.15.123.175

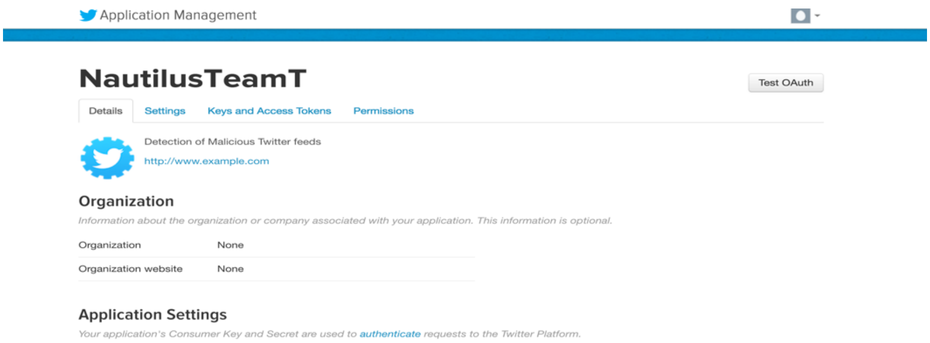
```
url added
ubuntu@ip-172-31-20-54:~$ head -n30 nohup.out
Connected to Twitter Successfully
Getting all tweets with query: boston
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
```

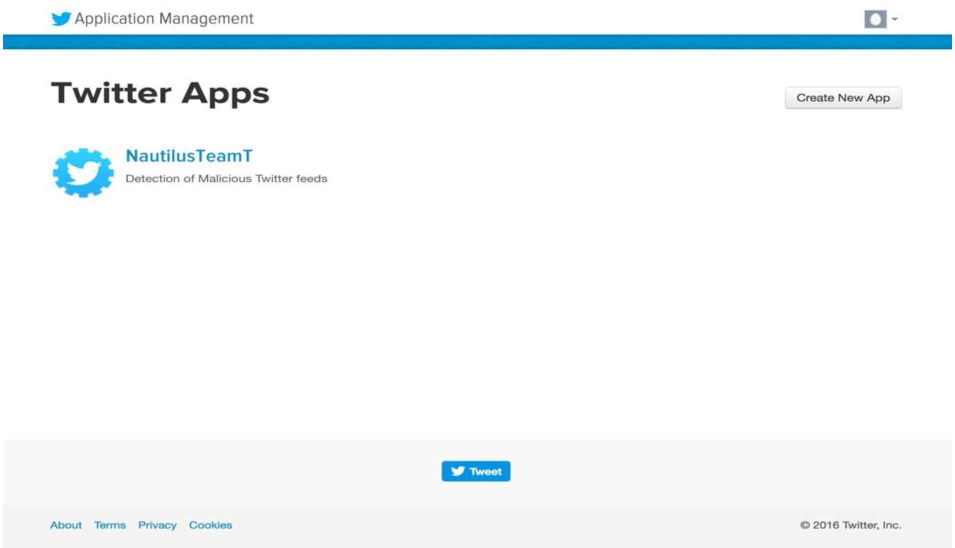
```
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
url added
Waiting 15 minutes to continue
Tweets read: 16761
Output URLS: 7555
ubuntu@ip-172-31-20-54:~$
```

```
ubuntu@ip-172-31-20-54:~$ tail -f nohup.out
url added
url added
url added
url added
url added
url added
url added
Waiting 15 minutes to continue
Tweets read: 33413
Output URLS: 16009
```

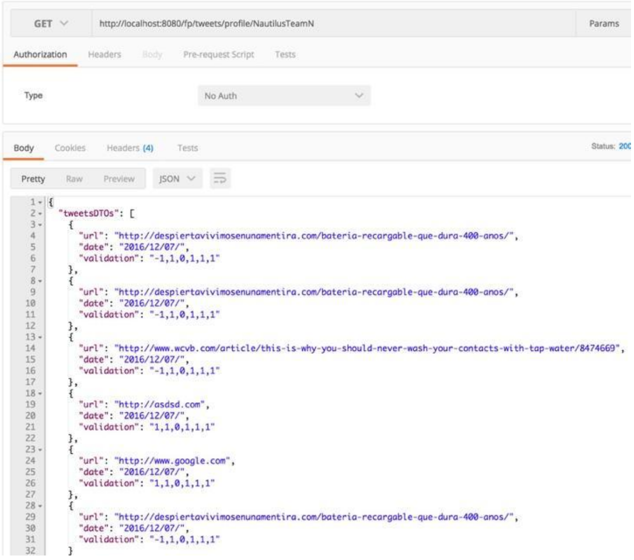
```
ubuntu@ip-172-31-20-54:~$ ps -aux | grep java
ubuntu    3530  2.0  8.3 991980 85212 ?        Sl   08:10   0:21 java -jar twitterScrapper.jar boston
ubuntu    3692  0.0  0.0  12944   928 pts/1    S+   08:28   0:00 grep --color=auto java
ubuntu@ip-172-31-20-54:~$ ls
hola  nohup.out  output1.txt  output2.txt  runner.sh  twitterScrapper.jar
ubuntu@ip-172-31-20-54:~$
```







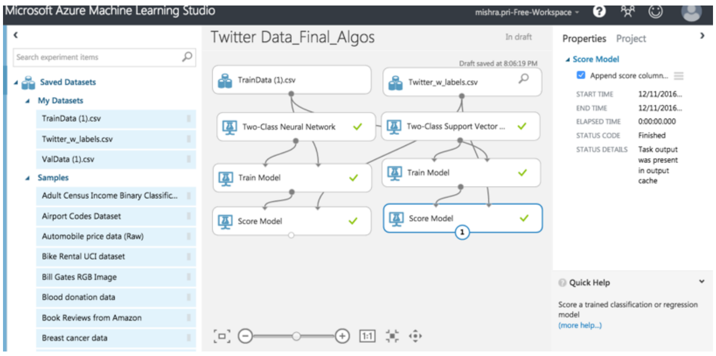
b. <u>Results from Map Reduce</u>




```
http://1.usa.gov/1Xd5hV1       ,1,1,0,1,1,1,10
http://10.Weather      ,1,1,0,1,1,1,56
http://100-singalong-songs-for-kids.i9-news.com/post/153880171592?utm_source=dlvr.it&utm_medium=twitter   ,-1,1,0,-1,1,1,2
http://1000oldies.de   ,1,1,0,1,1,1,8
http://1000phones.com/phone/1-617-830-7579    ,1,1,0,1,1,1,2
http://1000rockhits.de ,1,1,0,1,1,1,18
http://1001.tl/122813  ,1,1,0,1,1,1,2
http://100resilientcities.org/blog/entry/boston-resilience-blueprint-leads-with-discussions-of-race-equity        ,-1,1,0,1,1,1,6
http://1033ampradio.cbslocal.com/2016/11/21/listener-taylor-barfed-on-the-t/       ,-1,1,0,1,1,1,2
http://1033ampradio.cbslocal.com/2016/11/30/lorens-lodown-113016/   ,0,1,0,1,1,1,4
http://1063.mobi       ,1,1,0,1,1,1,18
http://13thstreetsports.com/2016/11/28/so-lets-talk-about-that-bogut-to-boston-rumors/     ,-1,1,0,1,1,1,4
http://1510wmex.com    ,1,1,0,1,1,1,48
http://1999.One,1,1,0,1,1,1,48
http://1bb.camsvip.pw  ,1,1,0,1,1,1,2
http://1gv.pussyxcam.pw       ,1,1,0,1,1,1,4
http://1rql.merfgtesfn.xyz    ,1,1,0,1,1,1,2
http://2.London,1,1,0,1,1,1,2
http://2.Team  ,1,1,0,1,1,1,4
http://2009taxes.org/2013/05/29/after-the-events-in-boston-and-texas-charity-scams-are-common/    ,-1,1,0,1,1,1,2
http://2016mama.com    ,1,1,0,1,1,1,2
http://2044.info/DAa   ,1,1,0,1,1,1,6
http://24.ONE  ,1,1,0,1,1,1,8
http://247sports.com/Bolt/DeAaron-Fox-Lonzo-Ball-would-make-sense-for-Boston-Celtics-49378510     ,-1,1,0,1,1,1,2
http://24prognoz.ru/novosti/baffalo-boston-prognoz-na-match-natsionalnoi-hokkeinoi-ligi-10643     ,-1,1,0,1,1,1,2
http://24prognoz.ru/novosti/huston-boston-prognoz-na-match-nba-6-dekabrya-2016-10900       ,-1,1,0,1,1,1,4
http://2dopeboyz.com/contact-us/Chicago       ,1,1,0,1,1,1,2
http://2fo7.com/2016/12/01/atlas-goes-off-road-and-this-time-doesnt-fall-over-watch-boston-dynamics-humanoid-robot-learn-to-trot-across-the-
trickiest-of-terrain/  ,-1,1,0,1,1,1,2
http://2fo7.com/2016/12/01/who-will-replace-big-papi-in-boston/     ,0,1,0,1,1,1,2
http://2id.camsolo.pw  ,1,1,0,1,1,1,2
http://2med.me/2fHauF5 ,1,1,0,1,1,1,2
http://2rd.feadat.pw   ,1,1,0,1,1,1,2
http://2yc.camspussy.pw       ,1,1,0,1,1,1,2
http://305Fitness-Boston.com  ,1,1,0,-1,1,1,2
http://31l.merfgtesfn.xyz     ,1,1,0,1,1,1,2
http://31nights.nticentral.org/       ,1,1,0,1,1,1,2
http://360.io/K2FNEw   ,1,1,0,1,1,1,2
http://3dotnews.com    ,1,1,0,1,1,1,4
http://3ds.one/W4t3xr  ,1,1,0,1,1,1,2
http://3gm.zpussy.pw   ,1,1,0,1,1,1,2
http://3k.teffko.pw    ,1,1,0,1,1,1,2
http://3ng.io/sl/KdcK65       ,1,1,0,1,1,1,2
http://3p0.ru/science/smotret-boston-dinamika-atlas-hodbyi-po-peresechennoy-mestnosti.70676.html?
utm_source=twitter&utm_medium=username&utm_campaign=blog    ,-1,1,0,1,1,1,2
http://4.nbcny.com/9D9P8cZ    ,1,1,0,1,1,1,2
http://4.srl/2fC1M7j   ,1,1,0,1,1,1,2
http://44d.io/bukx     ,1,1,0,1,1,1,2
```


c. Result from Machine Learning using Microsoft Azure

Two Class Neural Network and Two Class Support Vector



Twitter Data_Final_Algos > Score Model > Scored dataset

rows	columns
98046	10

double_slash_redirecting	Prefix_Suffix	HTTPS_token	Count	Scored Labels	Scored Probabilities
1	1	1	10	1	0.990307
1	1	1	56	1	0.990307
-1	1	1	2	1	0.999087
1	1	1	8	1	0.990307
1	1	1	2	1	0.990307
1	1	1	18	1	0.990307

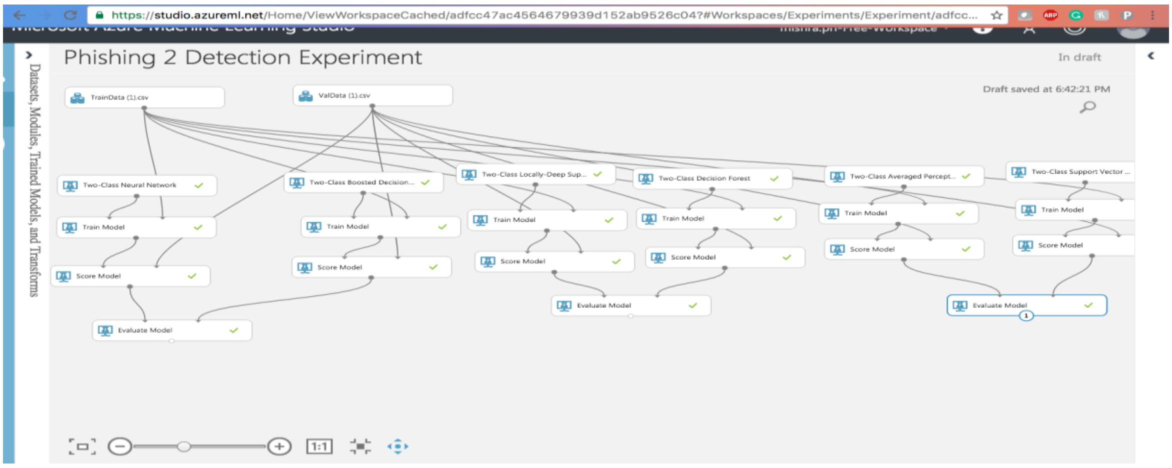
Statis

Visua

Twitter Data_Final_Algos > Score Model > Scored dataset

rows	columns
98046	10

double_slash_redirecting	Prefix_Suffix	HTTPS_token	Count	Scored Labels	Scored Probabilities
1	1	1	10	1	0.958993
1	1	1	56	1	0.958993
-1	1	1	2	1	0.815083
1	1	1	8	1	0.958993
1	1	1	2	1	0.958993
1	1	1	18	1	0.958993



Two Class average perceptron

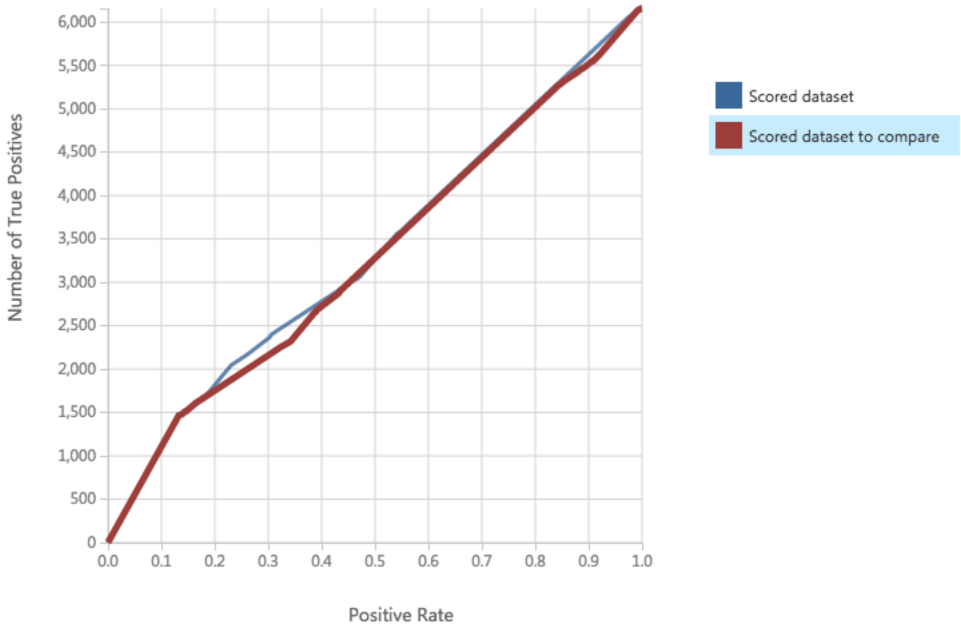
Final part 3 UCI › Evaluate Model › Evaluation results

True Positive	False Negative	Accuracy	Precision	Threshold	AUC
1606	4551	0.570	0.885	0.5	0.580

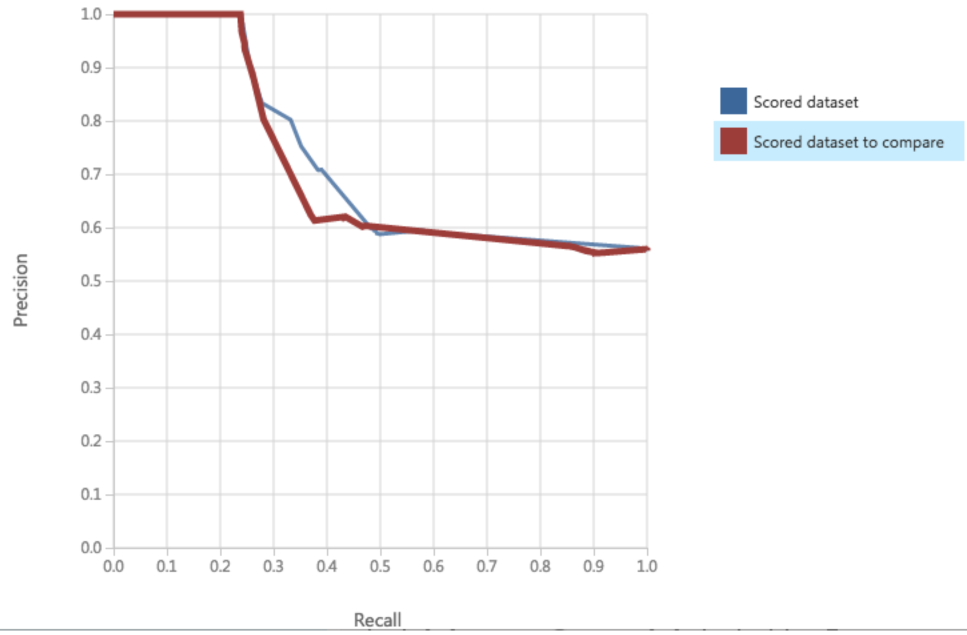
False Positive	True Negative	Recall	F1 Score
208	4690	0.261	0.403

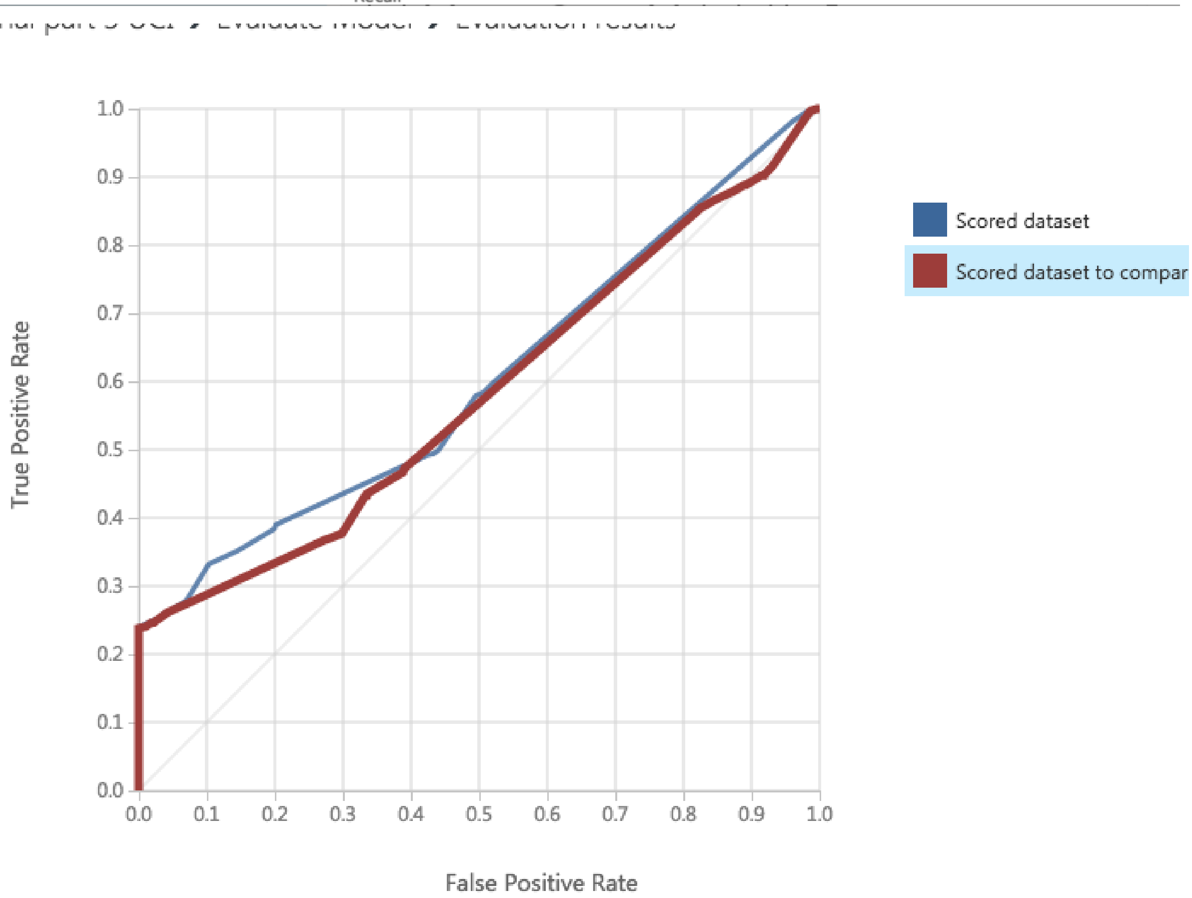
Positive Label	Negative Label
1	0

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	1461	0	0.132	0.575	0.384	1.000	0.237	0.511	1.000	0.000
(0.800,0.900]	4	0	0.133	0.576	0.384	1.000	0.238	0.511	1.000	0.000
(0.700,0.800]	0	0	0.133	0.576	0.384	1.000	0.238	0.511	1.000	0.000
(0.600,0.700]	48	89	0.145	0.572	0.390	0.944	0.246	0.509	0.982	0.004
(0.500,0.600]	93	119	0.164	0.570	0.403	0.885	0.261	0.508	0.958	0.011
(0.400,0.500]	1054	1432	0.389	0.535	0.509	0.619	0.432	0.482	0.665	0.107
(0.300,0.400]	2782	2679	0.883	0.545	0.684	0.558	0.884	0.447	0.118	0.470
(0.200,0.300]	715	561	0.998	0.559	0.716	0.558	1.000	1.000	0.004	0.577
(0.100,0.200]	0	18	1.000	0.557	0.715	0.557	1.000	1.000	0.000	0.580
(0.000,0.100]	0	0	1.000	0.557	0.715	0.557	1.000	1.000	0.000	0.580



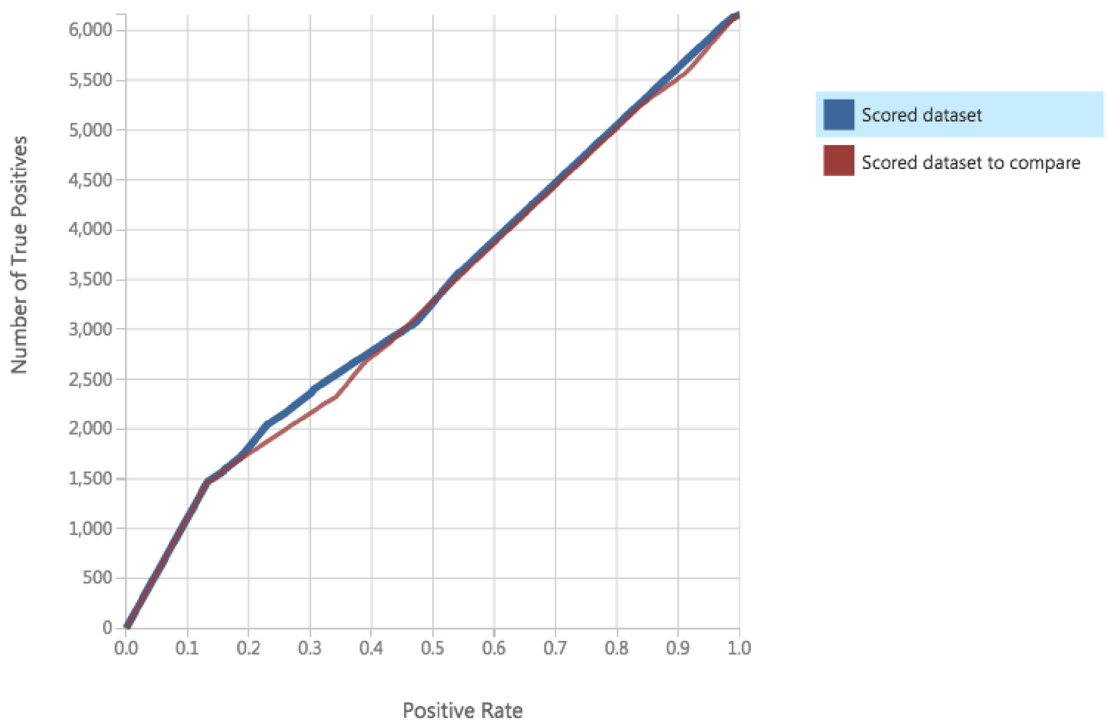
Final part 3 UCI › Evaluate Model › Evaluation results



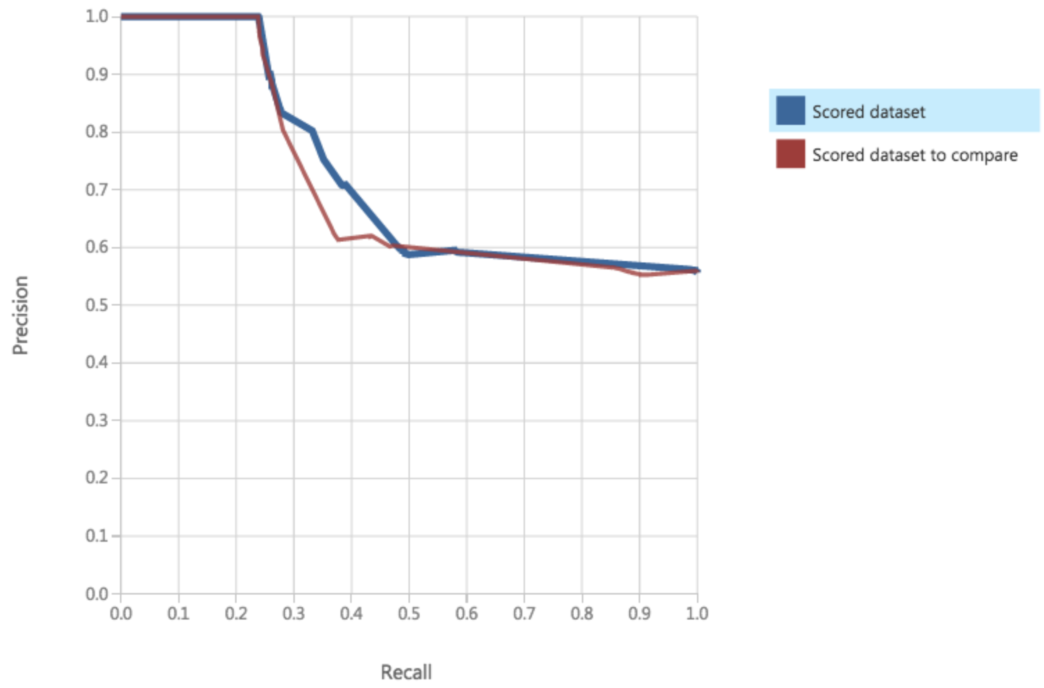


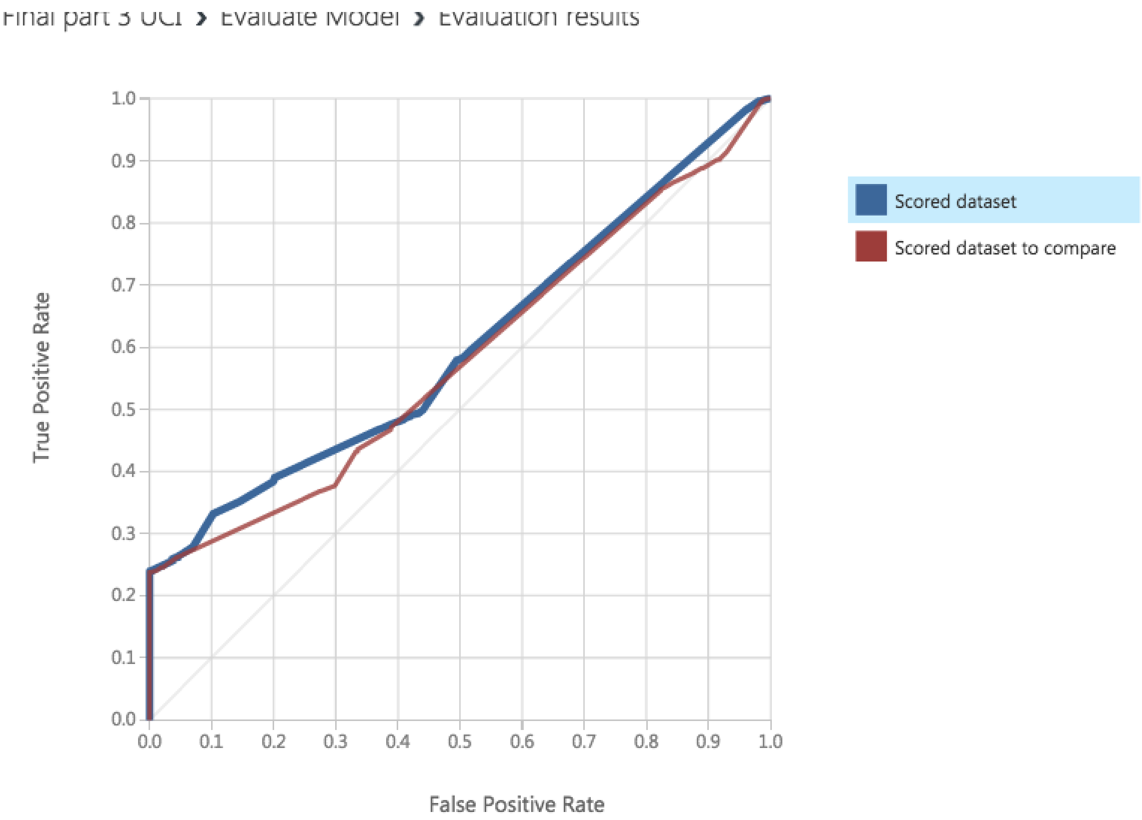
Two Class Boosted Decision

Final part 3 UCI › Evaluate Model › Evaluation results



Final part 3 UCI › Evaluate Model › Evaluation results





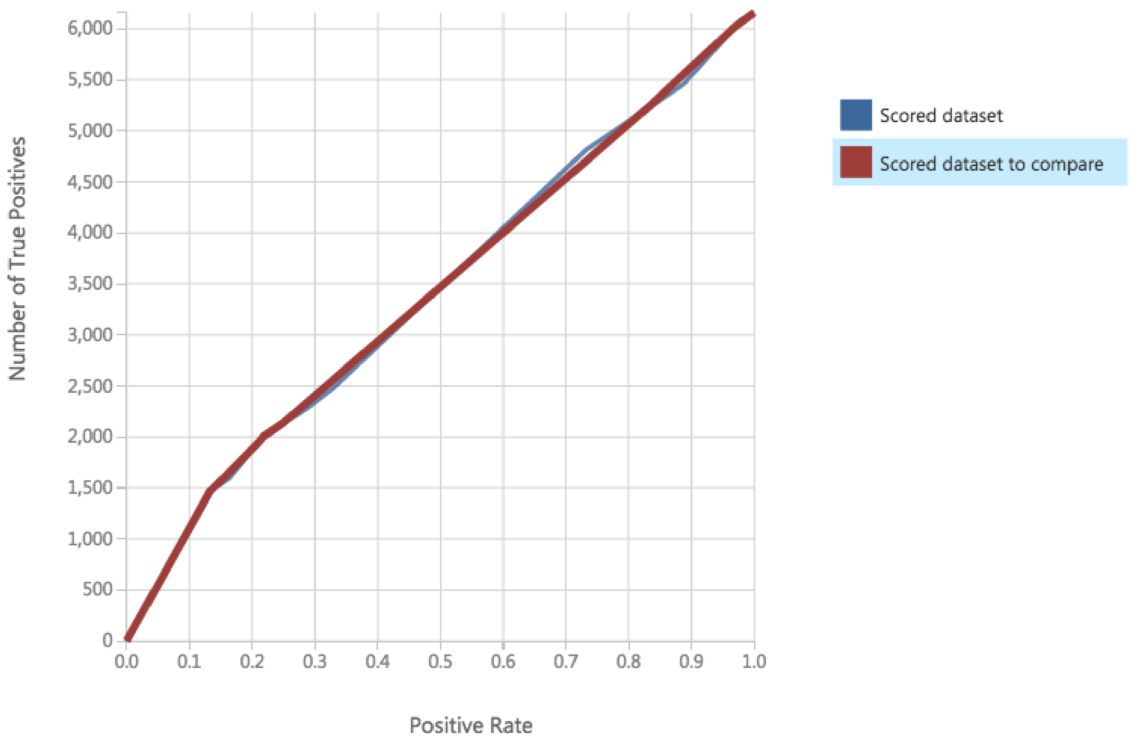
Two Class Decision Forest

Final_Part2 on UCI › Evaluate Model › Evaluation results

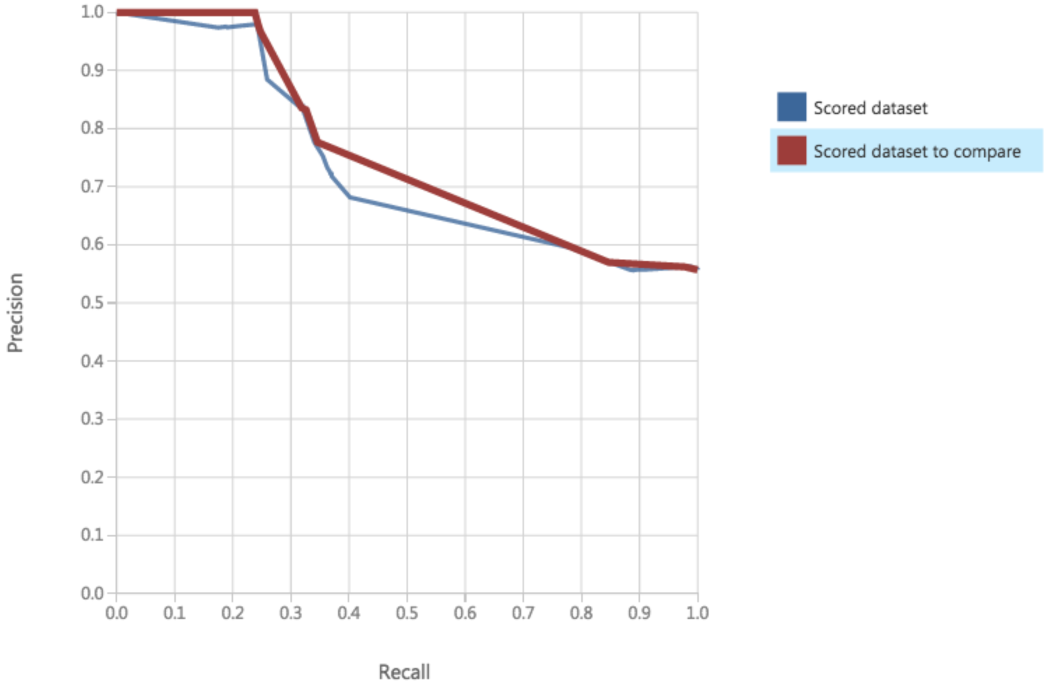
True Positive	False Negative	Accuracy	Precision	Threshold	AUC
1960	4197	0.586	0.836	0.5	0.624
False Positive	**True Negative**	**Recall**	**F1 Score**		
385	4513	0.318	0.461		
Positive Label	**Negative Label**				
1	0				

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	1465	0	0.133	0.576	0.384	1.000	0.238	0.511	1.000	0.000
(0.800,0.900]	40	37	0.139	0.576	0.391	0.976	0.244	0.511	0.992	0.002
(0.700,0.800]	19	15	0.143	0.576	0.394	0.967	0.248	0.511	0.989	0.003
(0.600,0.700]	436	333	0.212	0.586	0.461	0.836	0.318	0.518	0.921	0.022
(0.500,0.600]	47	19	0.218	0.588	0.468	0.832	0.326	0.520	0.918	0.023
(0.400,0.500]	0	0	0.218	0.588	0.468	0.832	0.326	0.520	0.918	0.023
(0.300,0.400]	121	208	0.248	0.580	0.478	0.777	0.346	0.515	0.875	0.037
(0.200,0.300]	3087	3326	0.828	0.559	0.681	0.570	0.847	0.505	0.196	0.442
(0.100,0.200]	805	752	0.969	0.563	0.714	0.562	0.978	0.603	0.042	0.582
(0.000,0.100]	137	208	1.000	0.557	0.715	0.557	1.000	1.000	0.000	0.624

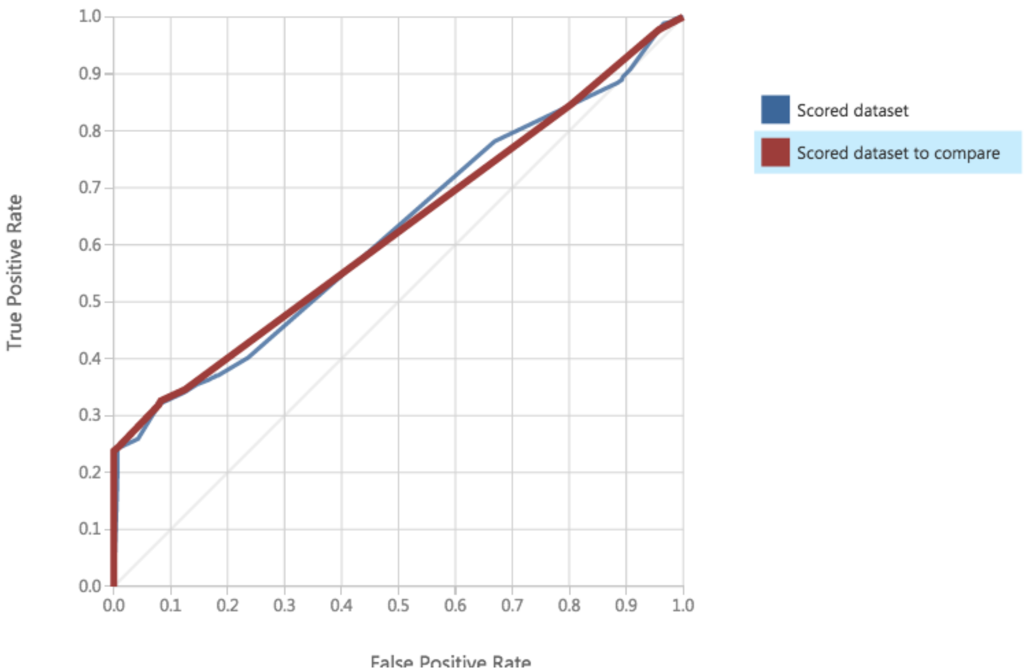
Final_Part2 on UCI › Evaluate Model › Evaluation results



Final_Part2 on UCI › Evaluate Model › Evaluation results



Final_Part2 on UCI › Evaluate Model › Evaluation results



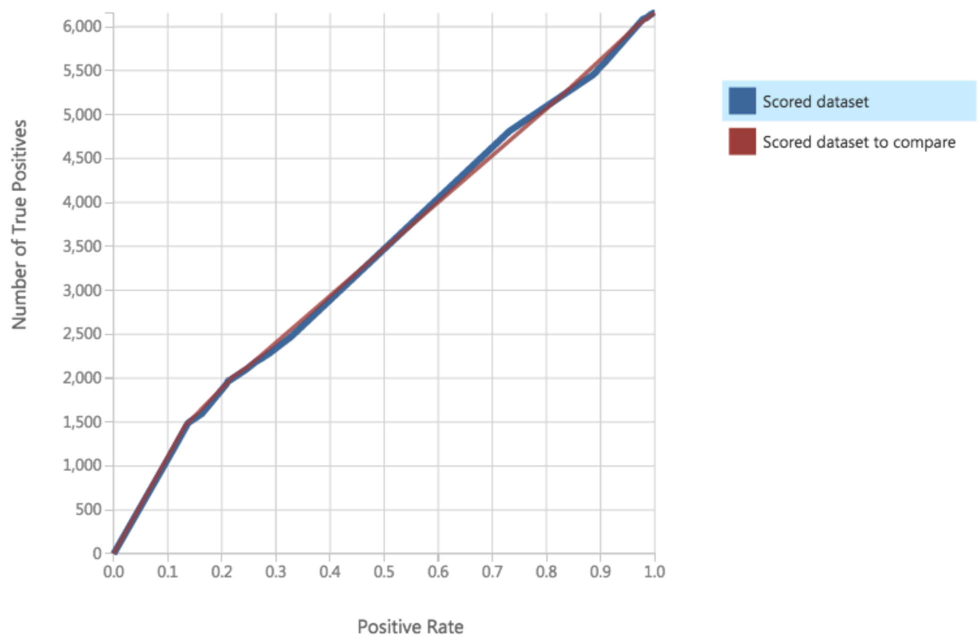
Two Class Locally Deep Support Vector Machine

Final_Part2 on UCI › Evaluate Model › Evaluation results ×

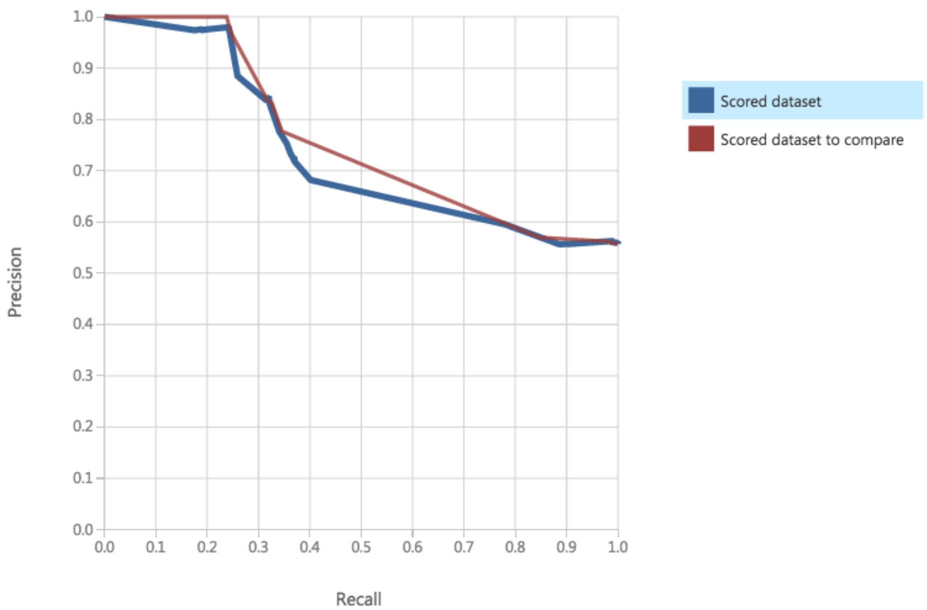
True Positive	False Negative	Accuracy	Precision	Threshold	AUC
2096	4061	0.578	0.776	0.5	0.623
False Positive	**True Negative**	**Recall**	**F1 Score**		
604	4294	0.340	0.473		
Positive Label	**Negative Label**				
1	0				

Score Bin	Positive Examples	Negative Examples	Fraction Above Threshold	Accuracy	F1 Score	Precision	Recall	Negative Precision	Negative Recall	Cumulative AUC
(0.900,1.000]	1929	371	0.208	0.584	0.456	0.839	0.313	0.517	0.924	0.020
(0.800,0.900]	167	233	0.244	0.578	0.473	0.776	0.340	0.514	0.877	0.035
(0.700,0.800]	0	0	0.244	0.578	0.473	0.776	0.340	0.514	0.877	0.035
(0.600,0.700]	0	0	0.244	0.578	0.473	0.776	0.340	0.514	0.877	0.035
(0.500,0.600]	0	0	0.244	0.578	0.473	0.776	0.340	0.514	0.877	0.035
(0.400,0.500]	0	0	0.244	0.578	0.473	0.776	0.340	0.514	0.877	0.035
(0.300,0.400]	0	0	0.244	0.578	0.473	0.776	0.340	0.514	0.877	0.035
(0.200,0.300]	0	0	0.244	0.578	0.473	0.776	0.340	0.514	0.877	0.035
(0.100,0.200]	170	264	0.283	0.570	0.488	0.723	0.368	0.509	0.823	0.055
(0.000,0.100]	3891	4030	1.000	0.557	0.715	0.557	1.000	1.000	0.000	0.623

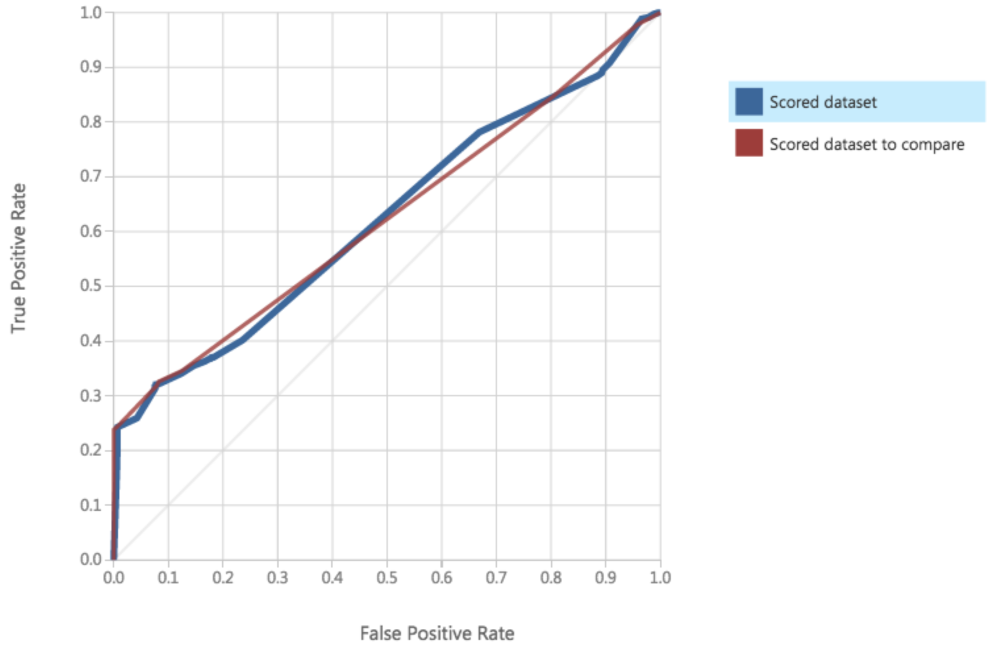
Final_Part2 on UCI › Evaluate Model › Evaluation results



Final_Part2 on UCI › Evaluate Model › Evaluation results



inal_Part2 on UCI › Evaluate Model › Evaluation results



Two Class Neural Network

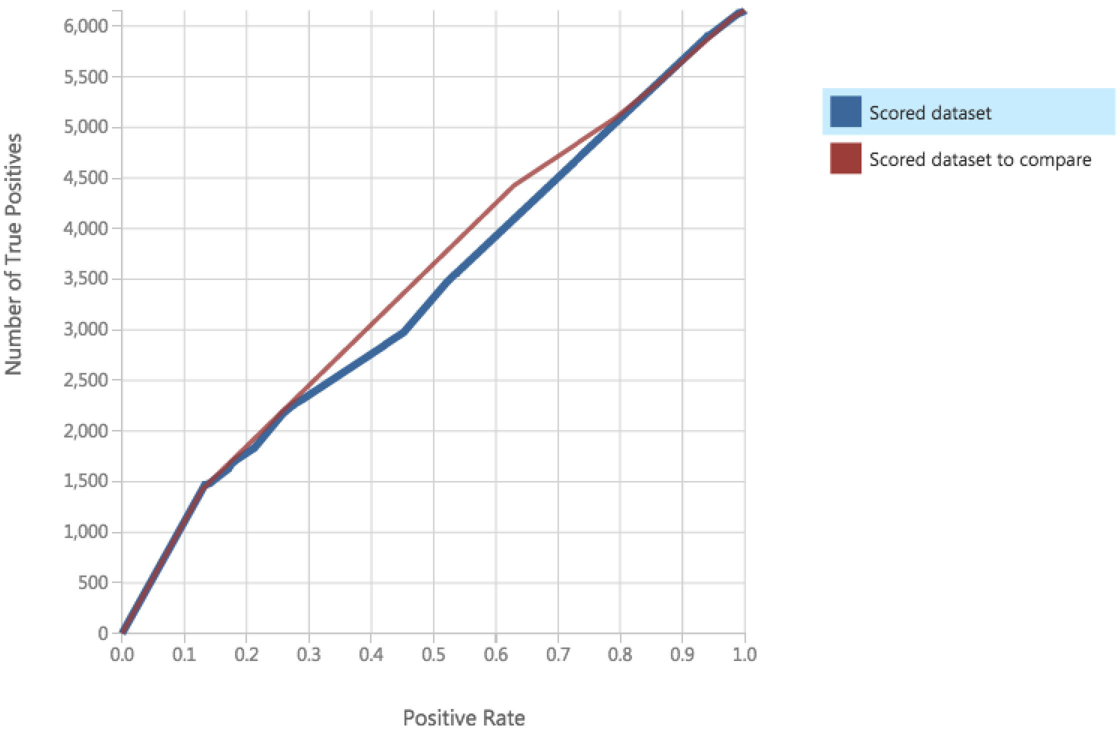
Final Experiment on UCI with Neural N/w ... › Evaluate Model › Evaluation results

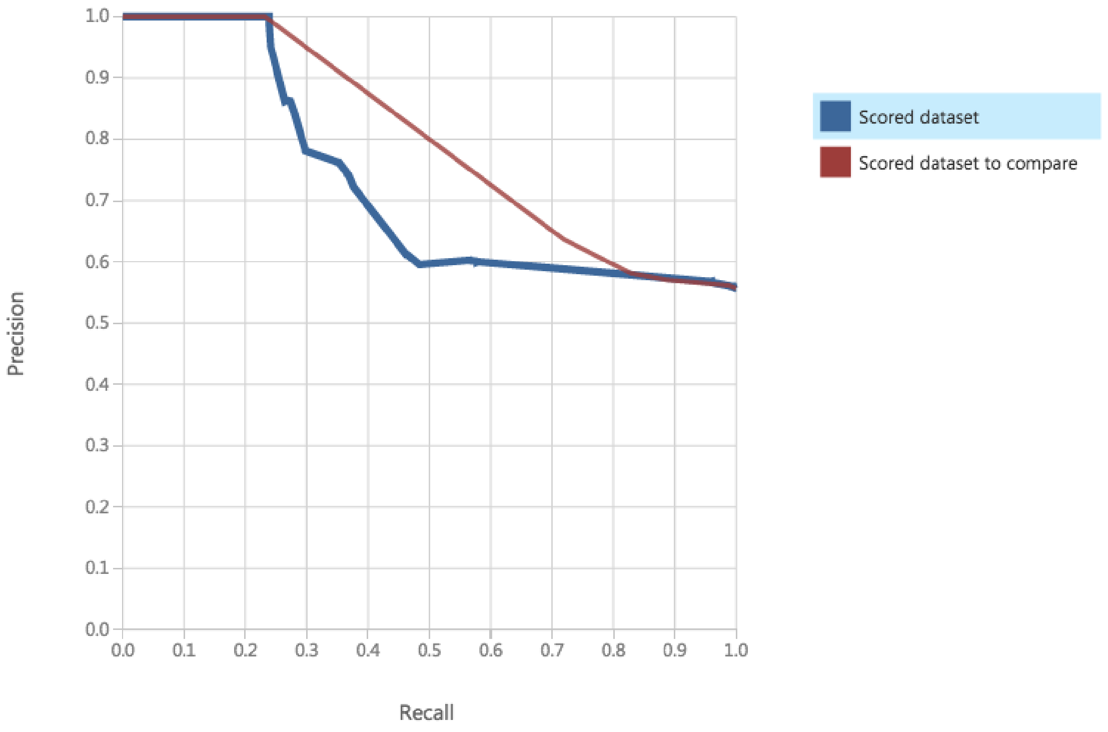
True Positive	False Negative	Accuracy	Precision	Threshold	AUC
3468	2689	0.550	0.602	0.5	0.606

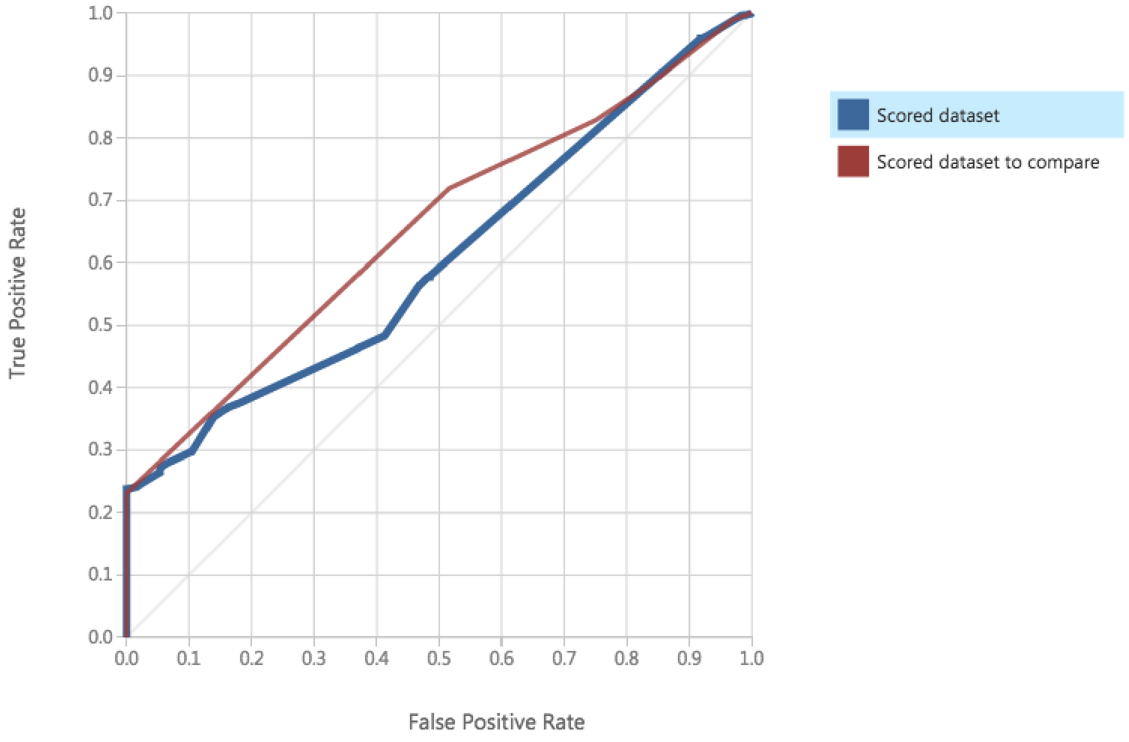
False Positive	True Negative	Recall	F1 Score
2289	2609	0.563	0.582

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]	1465	0	0.133	0.576	0.384	1.000	0.238	0.511	1.000	0.000
(0.800,0.900]	0	0	0.133	0.576	0.384	1.000	0.238	0.511	1.000	0.000
(0.700,0.800]	162	261	0.171	0.567	0.404	0.862	0.264	0.506	0.947	0.013
(0.600,0.700]	203	252	0.212	0.562	0.431	0.781	0.297	0.503	0.895	0.028
(0.500,0.600]	1638	1776	0.521	0.550	0.582	0.602	0.563	0.492	0.533	0.182
(0.400,0.500]	2435	2217	0.942	0.569	0.713	0.567	0.959	0.607	0.080	0.527
(0.300,0.400]	230	333	0.992	0.560	0.716	0.559	0.996	0.711	0.012	0.594
(0.200,0.300]	0	13	0.994	0.559	0.716	0.558	0.996	0.657	0.009	0.596
(0.100,0.200]	24	46	1.000	0.557	0.715	0.557	1.000	1.000	0.000	0.606
(0.000,0.100]	0	0	1.000	0.557	0.715	0.557	1.000	1.000	0.000	0.606





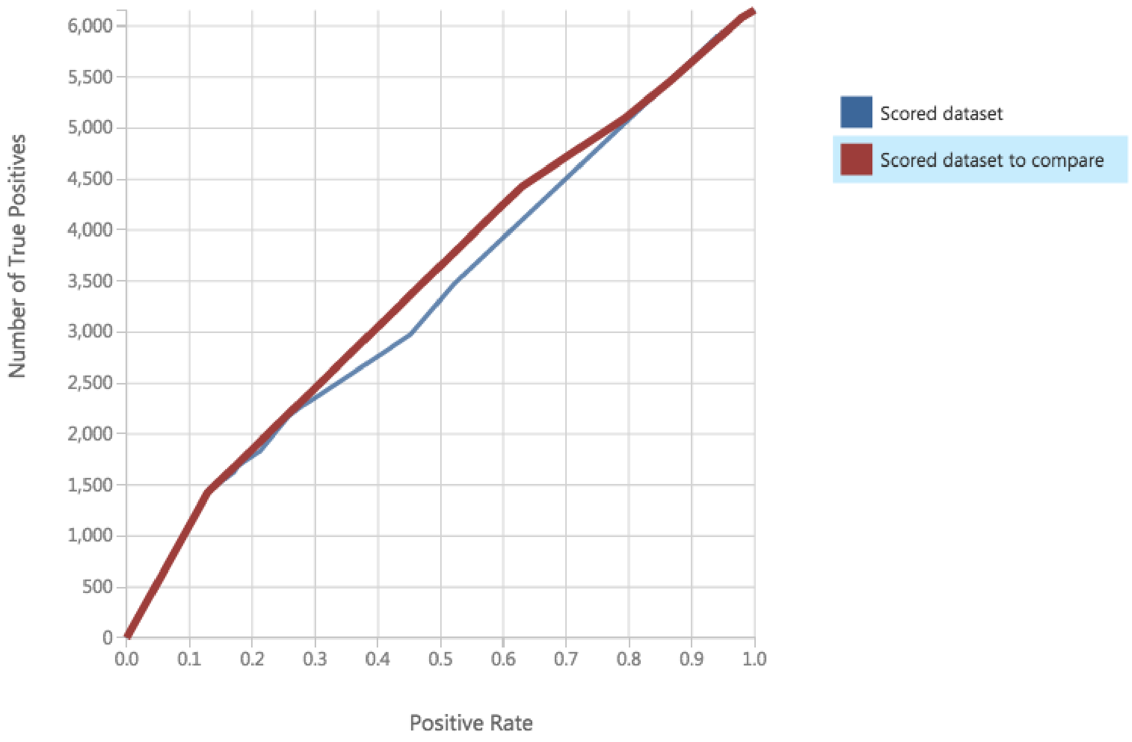


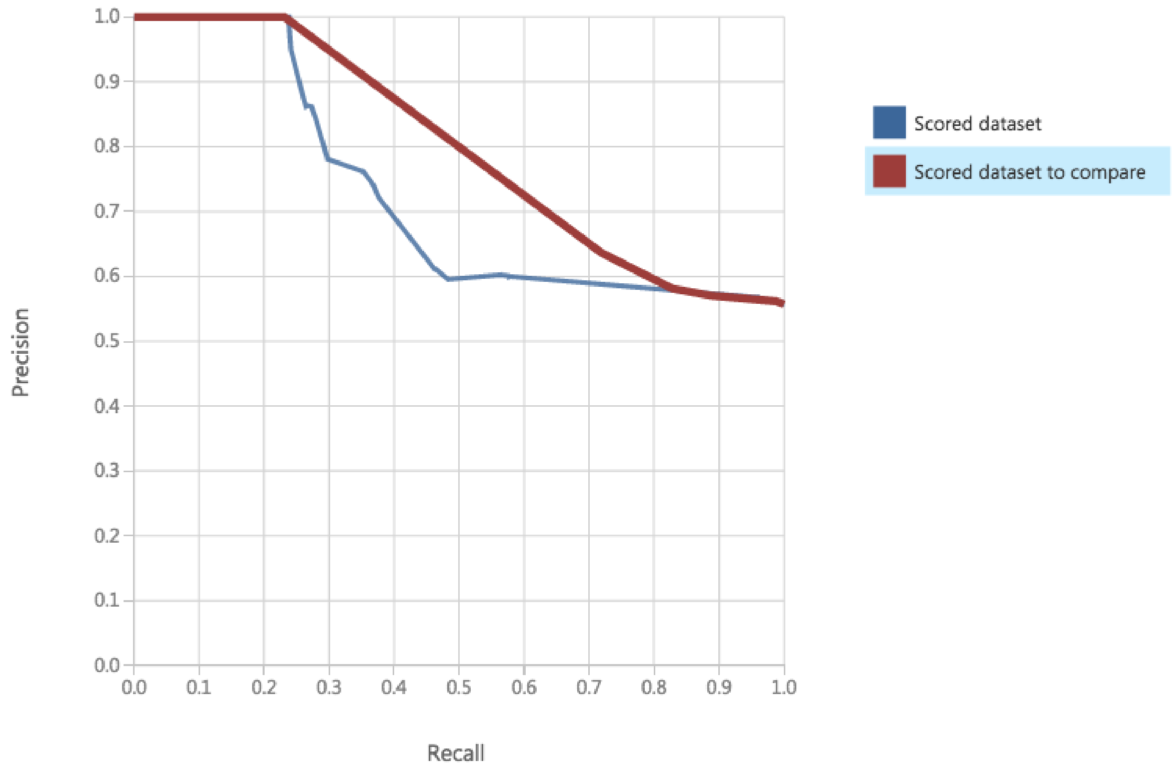
Two Class Support Vector

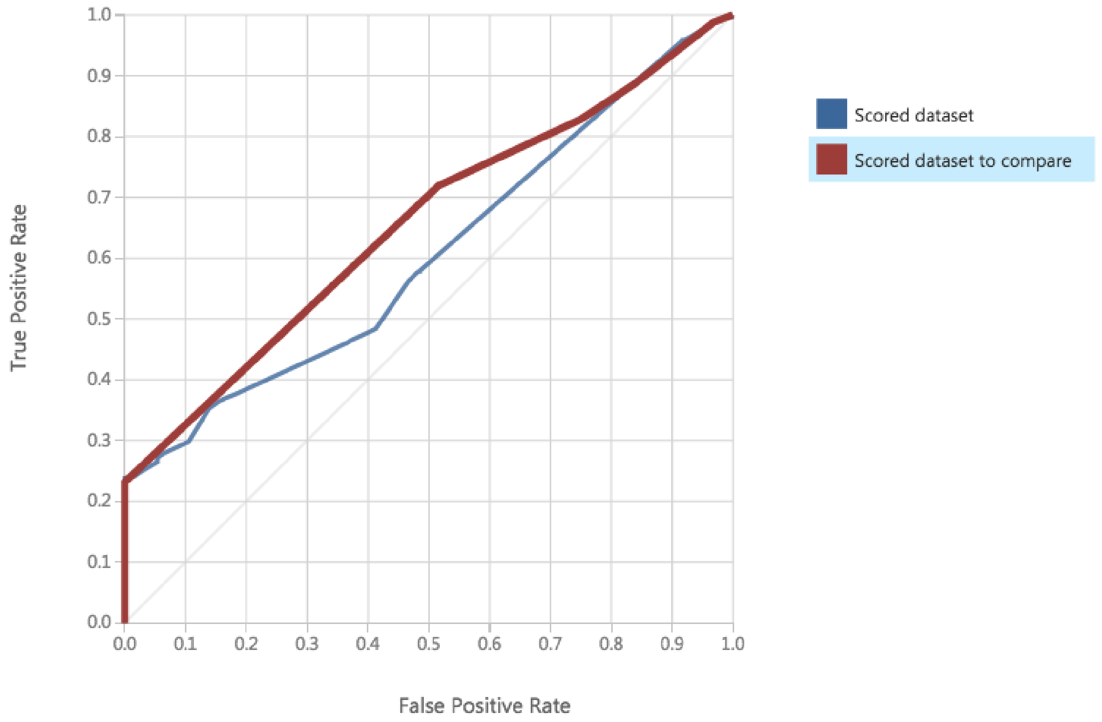
Final Experiment on UCI with Neural N/w ... › Evaluate Model › Evaluation results

True Positive	False Negative	Accuracy	Precision	Threshold	AUC
5099	1058	0.572	0.581	0.5	0.655
False Positive	**True Negative**	**Recall**	**F1 Score**		
3673	1225	0.828	0.683		
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]	1130	0	0.102	0.545	0.310	1.000	0.184	0.494	1.000	0.000
(0.800,0.900]	295	0	0.129	0.572	0.376	1.000	0.231	0.509	1.000	0.000
(0.700,0.800]	3001	2527	0.629	0.615	0.675	0.637	0.719	0.578	0.484	0.245
(0.600,0.700]	0	0	0.629	0.615	0.675	0.637	0.719	0.578	0.484	0.245
(0.500,0.600]	673	1146	0.793	0.572	0.683	0.581	0.828	0.537	0.250	0.426
(0.400,0.500]	358	432	0.865	0.565	0.694	0.571	0.886	0.531	0.162	0.502
(0.300,0.400]	605	619	0.976	0.564	0.716	0.562	0.985	0.647	0.036	0.620
(0.200,0.300]	18	13	0.978	0.565	0.716	0.562	0.987	0.676	0.033	0.623
(0.100,0.200]	77	161	1.000	0.557	0.715	0.557	1.000	1.000	0.000	0.655
(0.000,0.100]	0	0	1.000	0.557	0.715	0.557	1.000	1.000	0.000	0.655




Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Scored dataset
Scored dataset to compare


True Positive Rate
False Positive Rate
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
Scored dataset
Scored dataset to compare

d. Data Cleaning Using Python

File Edit View Insert Cell Kernel Help

Code CellToolbar

```
In [1]: def remove_duplicate_lines(inp,out):
            """Function to remove duplicate lines from a text file and save output to different file
            inp = name of input file as a string with the extension e.g. 'inp.txt'
            out = name of output file as a string with the extension e.g. 'out.txt'
            """
            unique_lines = []

            with open(inp) as f:
                for line in f:
                    if line not in unique_lines:
                        unique_lines.append(line)

            with open(out,'w') as f:
                for line in unique_lines:
                    f.write(line)

In [5]: remove_duplicate_lines('TrainData1.txt','TrainData2.txt')

In [ ]:
```

e. Result of Other Algorithms of Machine Learning (Winnow, Genetic Algorithm, Multilayer Perceptron

Winnow Algorithm

```
[98020]   1 1 0 1 1 1 1
[98021]   1 1 0 1 1 1 1
[98022]   1 1 0 1 1 1 1
[98023]   1 1 0 1 1 1 1
[98024]   0 1 0 1 1 1 1
[98025]   0 1 0 1 1 1 1
[98045]   -1 -1 0 -1 -1 -1 0

----------------------------------------------------------

Prediction accuracy on training data = 0.5694
Prediction accuracy on test data = 0.5803

Predicting URL with all 'legitmate' attriutes
Prediction is 'legitimate'

Predicting URL with all 'phishing' attriutes
Prediction is 'phishing'

End Winnow demo
```

Logistic Regression

```
Starting training using no regularization..

Best weights found:
-3.382 -0.024 -0.003  0.024  0.062 -0.012  10.000
Prediction accuracy on training data = 0.4722
Prediction accuracy on test data = 1.0000

Seeking good L1 weight
Good L1 weight = 0.000

Seeking good L2 weight
Good L2 weight = 0.000

Starting training using L1 regularization, alpha1 = 0.000

Best weights found:
-3.382 -0.024 -0.003  0.024  0.062 -0.012  10.000
Prediction accuracy on training data = 0.4722
Prediction accuracy on test data = 1.0000

Starting training using L2 regularization, alpha2 = 0.000

Best weights found:
-3.382 -0.024 -0.003  0.024  0.062 -0.012  10.000
Prediction accuracy on training data = 0.4722
Prediction accuracy on test data = 1.0000

End Regularization demo



C:\NortheasternUniversity\Fall 2016\Project\Logistic_Phishing>
```

Multilayer Perceptron

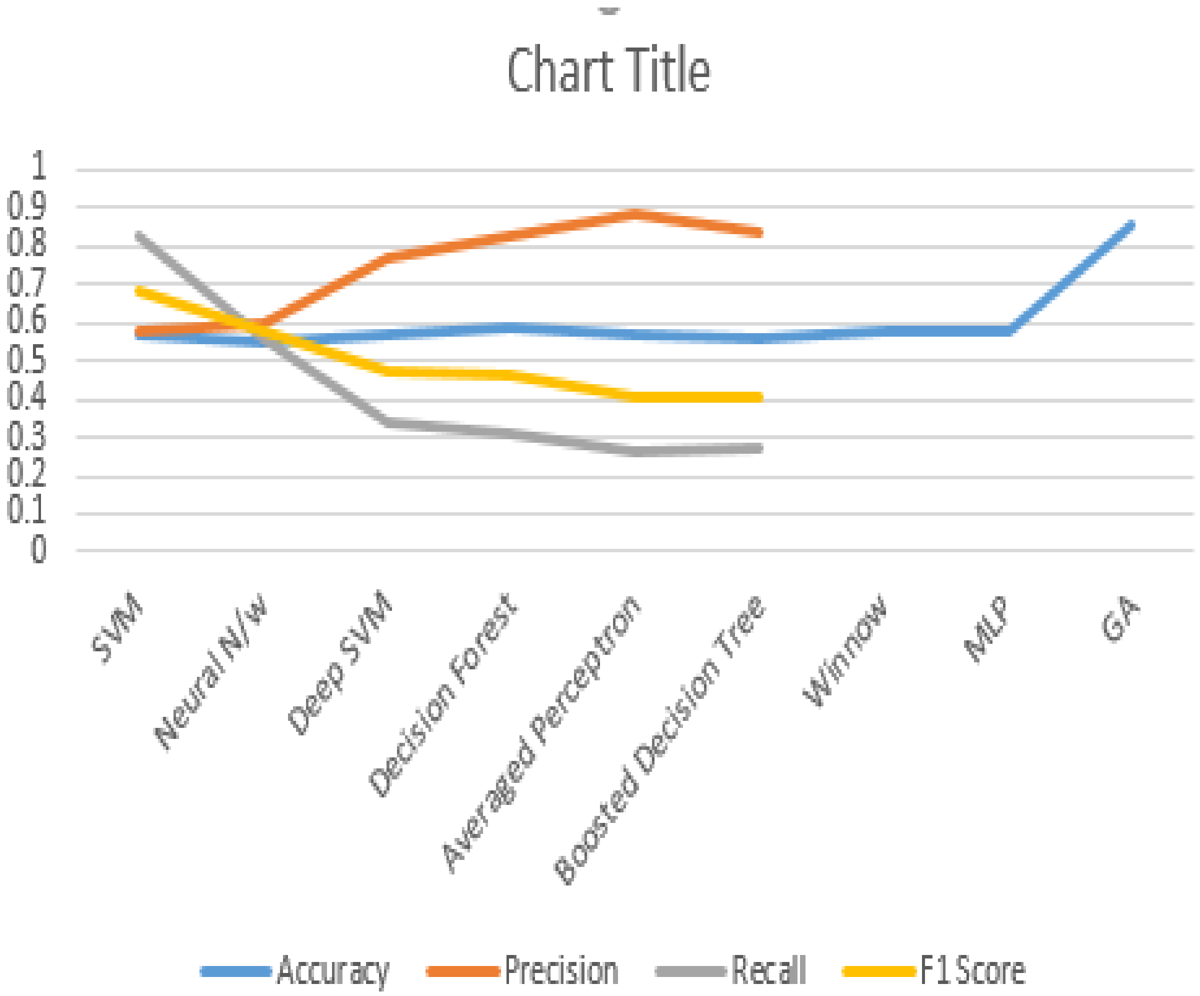
```
@ Javadoc  Declaration  Console
<terminated> DL4J [Java Application] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (Dec 11, 2016, 11:22:57 PM)
o.d.o.l.ScoreIterationListener - Score at iteration 410 is 0.16069400310516357
o.d.o.l.ScoreIterationListener - Score at iteration 420 is 0.1576046413845486
o.d.o.l.ScoreIterationListener - Score at iteration 430 is 0.15457604991065133
o.d.o.l.ScoreIterationListener - Score at iteration 440 is 0.1515563726425171
o.d.o.l.ScoreIterationListener - Score at iteration 450 is 0.14854816595713297
o.d.o.l.ScoreIterationListener - Score at iteration 460 is 0.1455861594941881
o.d.o.l.ScoreIterationListener - Score at iteration 470 is 0.1426843802134196
o.d.o.l.ScoreIterationListener - Score at iteration 480 is 0.13981754249996609
o.d.o.l.ScoreIterationListener - Score at iteration 490 is 0.13700495825873482
Evaluate model....
Final Target ....


Examples labeled as 0 classified by model as 0: 4305 times
Examples labeled as 0 classified by model as 1: 593 times
Examples labeled as 1 classified by model as 0: 4074 times
Examples labeled as 1 classified by model as 1: 2083 times


==========================Scores========================================
 Accuracy:        0.5778
 Precision:       0.6461
 Recall:          0.6086
 F1 Score:        0.6268
========================================================================
************************************
```

```
Javadoc  Declaration  Console
<terminated> DL4J [Java Application] C:\Program Files\Java\jre1.8.0_111\bin\javaw.exe (Dec 11, 2016, 11:24:34 PM)
https://youtu.be/zRCYEkA0_q8,1,1,0,1,1,1,1
https://youtu.be/zbsjSuJyCgU,1,1,0,1,1,1,1
https://youtu.be/zcIXRrhimi0,1,1,0,1,1,1,1
https://youtu.be/zeWIgAWMk-c,1,1,0,1,1,1,1
https://youtu.be/zthQPe41w24,1,1,0,1,1,1,1
https://youtube.com//watch?v=SSR6ZzjDZ94,1,1,0,1,1,1,1
https://youtube.com//watch?v=zthQPe41w24,1,1,0,1,1,1,1
https://youtube.com/watch?v=EGPDhEz9FkI,1,1,0,1,1,1,1
https://youtube.com/watch?v=sgTpIyl7qx4,1,1,0,1,1,1,1
https://zonalizer.makeinstall.se?0g3s7c1PQu6mNSLcOUJymQ,0,1,0,1,1,1,1
https://zonalizer.makeinstall.se?5mqnHG4cREKWsAp-CxX_lw,0,1,0,1,1,1,1
http://125.98.3.123/fake.html,-1,-1,0,-1,-1,1,1
http://whatsfungoing@inSanAntaino-checkOutLatest//TouristPage.com,-1,-1,0,-1,-1,1,1
https://httpHotStarNews-LatestUpdates@ABcedrf.com,-1,-1,0,-1,-1,1,1
http://0x58.0xCC.0xCA.0x62/2/paypal.ca/index.html,-1,-1,0,-1,-1,-1,1
http://federmacedoadv.com.br/3f/aze/ab51e2e319e51502f416dbe46b773a5e/?cmd=_home&amp;dispatch=11004d58f5b74f8dc1e7c2e8dd4105e811004d5
http://124.78.3.110/fakeHot.com,-1,-1,0,-1,-1,-1,1
https://http@colaradoinsightScene-TouristPage.html,-1,-1,0,-1,-1,1,0
www.teslatestingApplication.br@3f@rf/aze/cmd=_home&amp;dispatch=11004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8dc1e7c2e8dd4105e8@phi
httpab//@bitlinkClick@.com,-1,-1,0,-1,-1,-1,0
https://http@-CheckflipkartMcKenzee-LatestUpdates@NewDress@zopalo.website.html,-1,-1,0,-1,-1,-1,0
https://http@CheckibmBigData-@HotNewsPostinCalifornia09iuhjy76yt5564rew34*hjgyrfde@phishingSpam.com,-1,-1,0,-1,-1,1,0
https://phishingSpam@latestUpdatesinTheWorldOfHackathon-Checkout/Abc.com,-1,-1,0,-1,-1,1,0
https://@DrinkCrazy-PartyHard@boulderColaradoCamp/seeyouSoon.html,-1,-1,0,-1,-1,1,0
https://DjZack@TuneintoMyMusic@Crazypitch123.45.67.110/djcrazy.html,-1,-1,0,-1,-1,-1,0
https://FriendsRachelJoeyMeet-Hub@LosAngles/0x58.0xCC.0xCA.0x62/2/checkoutWebpage.com,-1,-1,0,-1,-1,-1,0
https://httpsearchEngineBingParty.com,-1,-1,0,-1,-1,-1,0
https://yahoo12forbesHighestPaid@business/1004d58f5b74f8dc1e7c2e8dd4105e811004d58f5b74f8dc1e7c2e8dd4105e@forbes.com,-1,-1,0,-1,-1,-1
https://hotspot@Jeeyomilke-SlashEvent.html,-1,-1,0,-1,-1,-1,0
```

Genetic Algorithm

```
Setting popSize = 4
Setting maxGeneration = 500
Setting early exit MSE error = 0.000
Setting mutateRate = 0.090
Setting mutateChange = 0.050
Setting tau = 0.300
Creating a 4-6-3 neural network
Using tanh and softmax activations

Beginning training
Training complete

Final weights and bias values:
 5.68810 -1.71994 -1.82197
 1.42476  3.48330  1.86523
 2.23151 -4.98860 -5.42930
-9.08476 -9.30498 -7.13815
-6.69261  8.30220  3.03176
 3.61276 -0.94880  3.07613
-1.42342  1.65556  2.36028
 8.77754  9.34345  1.50777
 8.36861  9.06027 -9.18410
-6.52579  5.64520 -1.02509
-1.61524 -2.19731  7.99744
-1.28768  7.56420 -5.64823
-1.83016 -2.02348  9.71713
 9.73265  7.06506  7.97359
-5.86816 -9.37927

Accuracy on training data = 0.4583

Accuracy on Validation data = 0.8524
```

Machine Learning Algorithm Accuracy Chart



9. Future Scope

- Project can be incorporated with other sophisticated algorithms like Deep Learning techniques to predict the accuracy of the phishing website
- Project can be extended to build an URL Advisor which many organizations can incorporate in their systems
- Currently, the Twitter scrapper runs the scripts to download the twitter feeds whereas in future the scope can be extended for the automation of the process where in as soon as the new tweets are tweeted, the system is automated to classify the tweets and download the feeds thus avoiding manual intervention
- CHATBOT implementation- Scope can be extended to incorporate ChatBot as when the user Tweets, and as soon as the system start scraping out the feeds, a chatbot can be used to popup a warning message if the Url's used in Tweets are malicious

10.References

https://deeplearning4j.org/

http://eprints.hud.ac.uk/24330//

https://archive.ics.uci.edu/ml/datasets/Phishing+Websites
http://eprints.hud.ac.uk/24330/6/MohammadPhishing14July2015.pdf