

CLICK THROUGH RATE PREDICTION

PROJECT REPORT: CSYE 7245(Big Data Sys & Int Analytics)

Under the Guidance of Prof. NIK BROWN

DHRUV KANAKIA

001222427

TABLE of CONTENTS:

1.1 PROJECT PROGRESS SUMMARY	3
1.2 What is Click Through Rate?	4
1.3 Problem Statement	4
1.4 Introduction	4
1.5 Background and Research Work	4
1.6 Dataset and Features	5
2.1 Exploratory Data Analysis	5
2.2 Feature Selection	7
3.1 Overall Workflow	7
3.2 WEB SCRAPING	8
3.3 DATA CLEANING	9
3.4 Uploading Data On Cloud	9
3.5 MODELING	9
3.6 COMPARISION OF MODELS	16
3.7 Deploying Decision Forest on AZURE ML	17
3.8 FLASK APPLICATION:	18
4.1 Challenges	22
4.2 Conclusion	22
4.3 References	22

1.1 PROJECT PROGRESS SUMMARY:

1. Project Proposal

DATE: 10/05/2017

In the proposal, I described what exactly CTR is and how marketing campaigns can be evaluated based on it. Mentioned that I'll use Logistic Regression, Neural Network, XGBoost and SVM to for modeling. Couldn't implement XGBoost but implemented Naïve Bayes instead.

Data Source: <https://www.kaggle.com/c/outbrain-click-prediction/data>

2. Project Progress Report

DATE: 11/18/2017

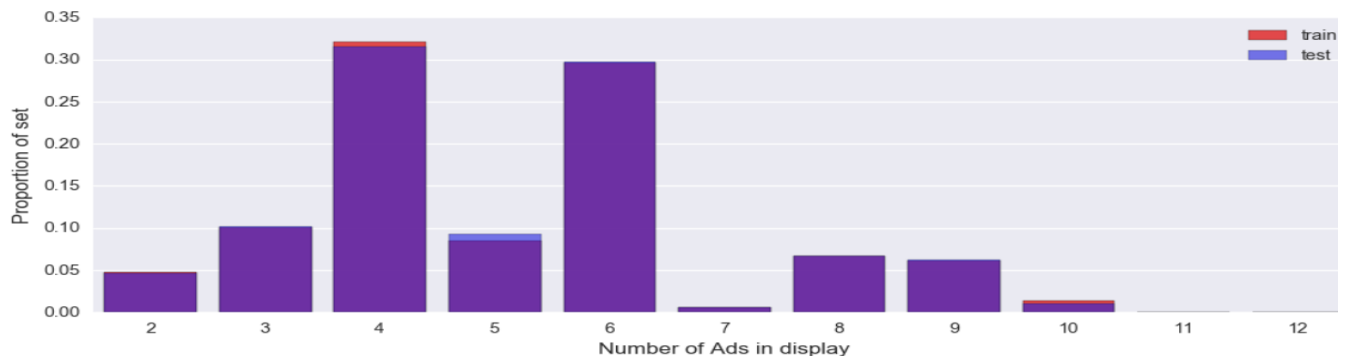
After looking at the related work that has been done in the past and what approaches have been taken I started with the first step i.e Exploratory Data analysis and feature selection.

Found various meaningful insights like : There are 300 unique topics, 97 unique categories and 1,326,009 unique entities, The number of ads clicked daily remained failry constant through out the 15 day window.53% of ads were clicked once or 47% ads were never clicked.

A sample Graph from the EDA part!!!!!!!!!!

```
plt.figure(figsize=(12,4))
sns.barplot(sizes_train.index, sizes_train.values, alpha=0.8, color='red', label='train')
sns.barplot(sizes_test.index, sizes_test.values, alpha=0.6, color='blue',label='test')
plt.legend()
plt.xlabel('Number of Ads in display', fontsize=12)
plt.ylabel('Proportion of set', fontsize=12)
```

<matplotlib.text.Text at 0x230061ee4e0>



3. Project Progress Report

DATE: 12/06/2017

After EDA and Feature Selection I started working on Metadata modeling and readying the data for models. The major parts covered here were: Web Scraping, Data Backup on Azure, Metadata modeling for prediction & Azure ML Logistic Regression.

4. FINAL PROJECT

DATE: 12/16/2017

Scroll through to know more!!!!!!!!!!!!!!

The final end product will be described in this report hence forth building upon the previous pit stops of this project. Go through the next major steps for understanding the entire flow in detail

1.2 What is Click Through Rate?

Click-through rate (CTR) is the ratio of users who click on a specific link to the number of total users who view a page, email, or advertisement. It is commonly used to measure the success of an online advertising campaign for a particular website as well as the effectiveness of email campaigns.

1.3 Problem Statement:

With millions of user's having access to the web these day's it becomes really important to provider personalization marketing campaigns to improve their efficiency. More the click through rate better the revenue generated and more successful the campaign. To tackle this problem using data science I planned to come up with an application that uses machine learning to identify whether a user will click on an ad or not so that we can come up with an estimate of how many users will actually click on the ad.

1.4 Introduction:

The web is fantastically rich in the data that the general population require on everyday premise. Consistently we get to many archives with respect to different themes like travel, news, formulas and other subjects. While the clients surf on their most loved locales, they are given part of substance which they may be occupied with. Outbrain is one such substance disclosure stage that conveys these minutes while the clients surf their most loved websites. Outbrain pairs relevant content with curious readers in about 250 billion personalized recommendations every month across many thousands of sites. In this project, we aim to predict which pieces of content its global base of users are likely to click on. Improving Outbrain's recommendation algorithm will mean more users uncover stories that satisfy their individual tastes.

1.5 Background and Research Work:

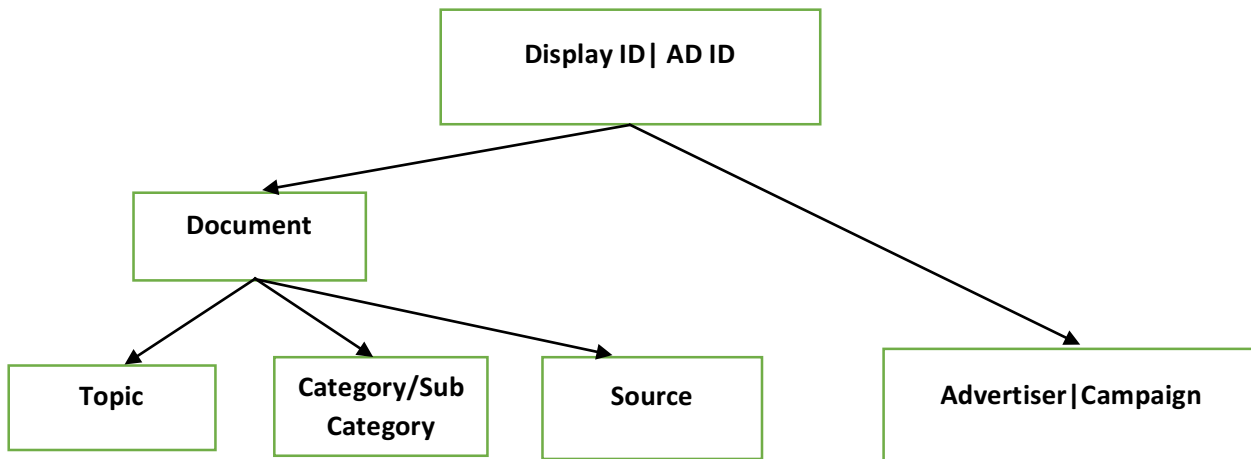
During research, I found several groups that sought to use machine learning techniques in order to determine whether an advertisement got clicked. A team from Google published an article describing how they present advertisements to users based on search queries. They used probabilistic methods to reduce features and they got best results for logistic regression with regularization with use of modified gradient descent algorithms. But, we do not have large dataset/information about user's behavior and we will have to do feature selection to obtain the features.

(Chang et al. 2010, Kudo et al. 2003) in their research paper used degree-n polynomial features and learn weights of each feature either linearly or using SVM. But, this can't be used here as our data is sparse and feature combination scenarios being rare provide relatively little value.

(Koren et al. 2009; Chen et al. 2012) suggested using factorization models that involved using users and ads by a fixed length vector followed by collaborative filtering.

1.6 Dataset and Features:

Below is the hierarchy of how the attributes are linked to each other.



User: It is an entity that browse pages and has unique identifier UUID

Document: A document is a page identified by document_id. It has following features

Source and Publisher identified by source_id and publisher_id

Publish time identified by publish_time

Category, Topic and Entity are represented by category_id, topic_id and entity_id

Advertisement: Advertisement is identified by which document it resides in, who is the advertiser and has unique identifier campaign_id and advertiser_id

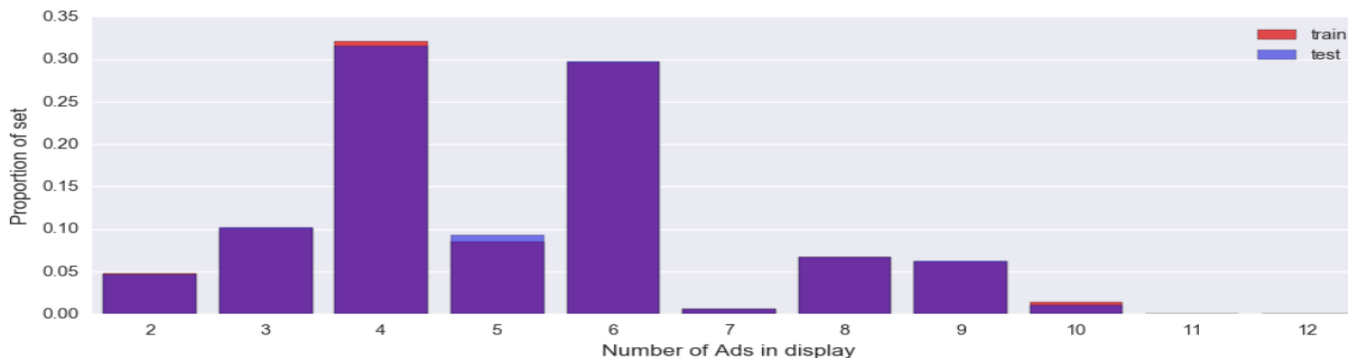
Display: It is the page/screen where the ad has been displayed. It is indicated by uuid and document_id.

2.1 Exploratory Data Analysis

The first step always is to look and explore the available data to understand how it is distributed and what is the relation with other attributes. So, I started with the EDA and found out few meaningful insights that are mentioned below:

- i. This graph shows that the split up between the train and test is properly done and we do not need any changes to be made in it.

```
plt.figure(figsize=(12,4))
sns.barplot(sizes_train.index, sizes_train.values, alpha=0.8, color='red', label='train')
sns.barplot(sizes_test.index, sizes_test.values, alpha=0.6, color='blue',label='test')
plt.legend()
plt.xlabel('Number of Ads in display', fontsize=12)
plt.ylabel('Proportion of set', fontsize=12)
<matplotlib.text.Text at 0x230061ee4e0>
```



- ii. I wanted to check how many ads in the train set are also in test set. The percentage I got was 88% which is quite a good number. So, the data again doesn't need any modifications here.
- iii. This analysis showed that there are only few user who are in the data set more than 2 times. This means we cannot come up with user based models.

```
uuid
ef7761dd22277c    38
45d23867dbe3b3    38
c0bd502c7a479f    42
2759b057797f02    46
b88553e3a2aa29    49
Name: uuid, dtype: int64
Users that appear less than 2 times: 88.42%
Users that appear less than 5 times: 99.51%
Users that appear less than 10 times: 99.96%
```

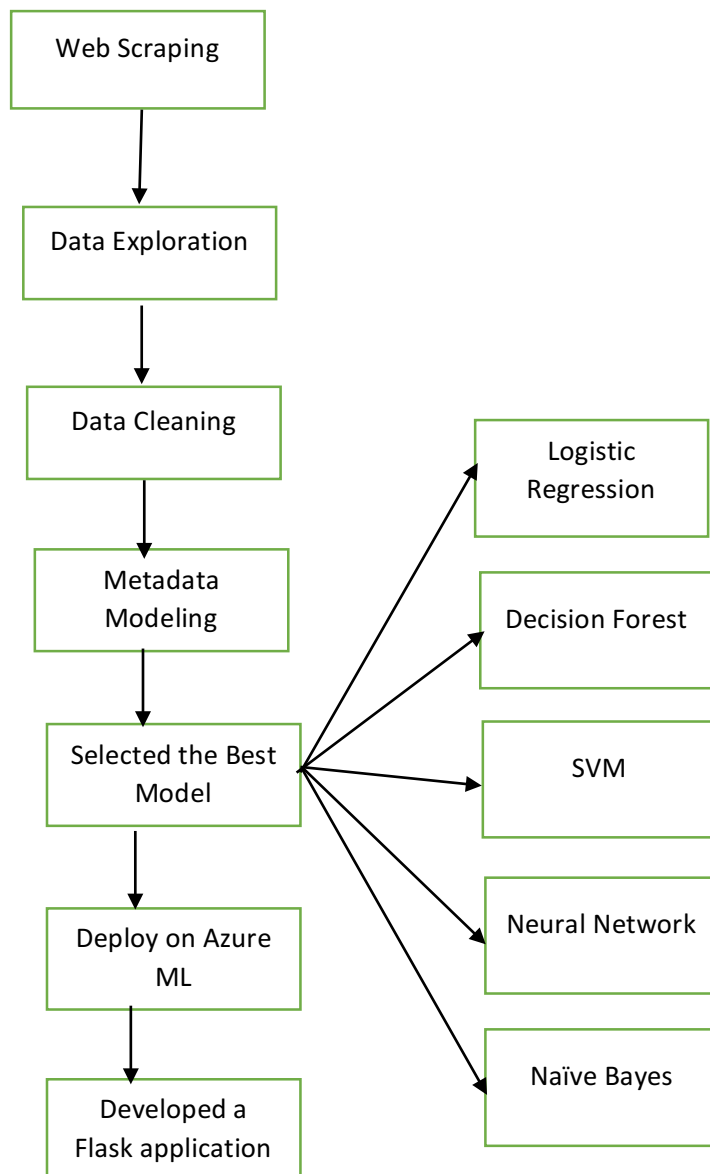
- iv. Few numerical insights: There are 300 unique topics, 97 unique categories and 1,326,009 unique entities, The number of ads clicked daily remained fairly constant through out the 15 day window. 53% of ads were clicked once or 47% ads were never clicked.
- v. Various other Analysis were performed like Click distributions(Daily), Click distribution(Hourly), Percentage of user's by Country, Percentage of user's by State, Source of traffic by Platform.
- vi. We observed that the click frequency is very high between 10-15 hrs. Around 10-11 a.m. time people are usually on their commute to work, school etc and pass their time surfing net. Around 1-3 p.m. is usually lunch time where people get time to surf internet and thats the frequency is high.
- vii. The frequency in the bucket 15-20 is again when people are traveling back home and around dinner time when they have time to surf internet.
- viii. Maximum percentage of clicks were made through mobile phones, followed by desktop and then tablets.

2.2 Feature Selection:

- Most of the documents have several associated Topics, Categories and Entities. The association between a document and a Topic or a Category or an Entity is quantified by the confidence level.

While analyzing I realized Topics and Categories, Entities as a feature will add many dummy variables but will be extremely sparse. Hence considering only the top 5 categories as features. Platform is pretty straight forward and can be considered as a feature.

3.1 OVERALL FLOW



3.2 WEB SCRAPING:

There are 9 csv files that needs to be scraped from the kaggle website. But, before scraping we need to login to Kaggle.com to get access to files in the same session. Used request library to login and gain access to the files. Here's a screenshot of the code snippet that show's how I extracted the files.

```

In [3]: logger.info('Reading Json')
        with open('config.json') as d: #Reading all the downloading Links
            data=json.load(d)

In [4]: links=data['links']

In [5]: links

Out[5]: ['https://www.kaggle.com/c/outbrain-click-prediction/download/clicks_test.csv.zip',
'https://www.kaggle.com/c/outbrain-click-prediction/download/clicks_train.csv.zip',
'https://www.kaggle.com/c/outbrain-click-prediction/download/documents_entities.csv.zip',
'https://www.kaggle.com/c/outbrain-click-prediction/download/documents_categories.csv.zip',
'https://www.kaggle.com/c/outbrain-click-prediction/download/documents_topics.csv.zip',
'https://www.kaggle.com/c/outbrain-click-prediction/download/documents_meta.csv.zip',
'https://www.kaggle.com/c/outbrain-click-prediction/download/page_views_sample.csv.zip',
'https://www.kaggle.com/c/outbrain-click-prediction/download/events.csv.zip',
'https://www.kaggle.com/c/outbrain-click-prediction/download/promoted_content.csv.zip']

In [12]: logger.info('Logging in and Downloading Zip ')
        with requests.session() as s:
            username = input("Enter email address:")
            paswd = input("paswd:")
            s = requests.session()
            payload = {'username': username , 'password': paswd} #Using ID Password to Login and then hit the links to download fi
            url = 'https://www.kaggle.com/account/login?ReturnUrl=%2faccount%2fset-username%3freturnUrl%3d%2fc%2foutbrain-click-prediction'
            a = s.post(url, data=payload)
            for x in links:
                file1 = s.get(x)
                m1 = z.ZipFile(io.BytesIO(file1.content))
                m1.extractall()

```

Enter email address:dhruvkanakia7@gmail.com
paswd:Dhruvkanakia7.

3.3 DATA CLEANING:

After extracting the data, I looked at all the files to see if it has any missing values. While looking at the data I found that there are no missing values in any columns of the 8 files. Here's a snippet that shows there are no missing values in it.

Checking if any of the above files need to be cleaned!!!!

```
In [8]: promoted_content.isnull().sum()
```

```
Out[8]: ad_id      0
document_id  0
campaign_id  0
advertiser_id 0
dtype: int64
```

```
In [9]: promoted_content.dtypes
```

```
Out[9]: ad_id      int64
document_id  int64
campaign_id  int64
advertiser_id int64
dtype: object
```

```
In [10]: document_topic.isnull().sum()
```

```
Out[10]: document_id  0
topic_id    0
confidence_level  0
dtype: int64
```

```
In [11]: document_category.isnull().sum()
```

```
Out[11]: document_id  0
category_id  0
confidence_level  0
dtype: int64
```

```
In [12]: document_entity.isnull().sum()
```

```
Out[12]: document_id  0
entity_id    0
confidence_level  0
dtype: int64
```

```
In [13]: events_data.isnull().sum()
```

```
Out[13]: display_id  0
uuid      0
document_id  0
timestamp  0
platform   0
geo_location 340
dtype: int64
```


3.4 Uploading Data On Cloud:

Once the data file is cleaned we upload the data on azure blobs to provide the data for modeling. Here's a code snippet screenshot and AZURE BLOB screenshot that shows file has been successfully uploaded.

Uploading data on Azure Blobs

```
In [3]: import glob

In [4]: from azure.storage.blob import BlockBlobService
from azure.storage.blob import ContentSettings

In [5]: from os import walk

f = []
for files in glob.glob('*.csv'):    #Retrieving all CSV FILES
    print(files)
    f.append(files.rsplit('\\', 1)[-1])

clicks_test.csv
clicks_train.csv
documents_categories.csv
documents_entities.csv
documents_meta.csv
documents_topics.csv
Document_Metadata.csv
events.csv
Merged_data_File.csv
merged_df_with_CL_Cat.csv
merged_df_with_CL_Ent.csv
page_views_sample.csv
Prediction_final_file.csv
Prediction_final_file2.csv
Prediction_final_file3.csv
promoted_content.csv

In [6]: f[14]
Out[6]: 'Prediction_final_file3.csv'

In [7]: block_blob_service = BlockBlobService(account_name='csyefinalproject', account_key='2dt200oxkQzVDhdoHKAwhPm09FCLeAstV9XJFrS+W0B8xK

#Updating account name and account key to gain access to azure blobs

In [8]: # Using Container name, File Location and file to upload csv onto Azure Blob
block_blob_service.create_blob_from_path(
    'first',
    f[14],
    f[14],
    content_settings=ContentSettings(content_type='csv')
)
```

SCREENSHOT THAT SHOWS THE FILE SUCCESSFULLY UPLOADED ON AZURE BLOBS:

The screenshot shows the Microsoft Azure portal interface. On the left is a navigation pane with options like Dashboard, All resources, Resource groups, App Services, Function Apps, SQL databases, Azure Cosmos DB, and Virtual machines. The main area is divided into two panes. The left pane shows the 'Blob service' for 'csyefinalproject' with a 'first' container. The right pane shows the 'first' container with a table of blobs. The table has columns for NAME, MODIFIED, ACCESS TIER, and BLOB TYPE. Two files are listed: 'Document_Metadata.csv' (modified 12/5/2017, 8:14:33 AM) and 'Prediction_final_file3.csv' (modified 12/17/2017, 5:14:23 AM). The 'Prediction_final_file3.csv' file is highlighted in yellow.

NAME	MODIFIED	ACCESS TIER	BLOB TYPE
Document_Metadata.csv	12/5/2017, 8:14:33 AM	Hot (Inferred)	Block blob
Prediction_final_file3.csv	12/17/2017, 5:14:23 AM	Hot (Inferred)	Block blob

3.5 HANDLING UNBALANCED DATASET:

The classification variable “CLICKED” is unbalanced in the cleaned dataset. To handle the imbalance problem I’ve used Smote which is one of the Over Sampling solution to the problem. Here’s the code snippet screenshot that shows how we handle the imbalanced dataset!

```
In [4]: train=train.sample(frac=0.2,random_state=200)
train.shape
LD = df_7.sample(frac=0.7)
LD.shape
LD = pd.DataFrame(train)
LD.shape

Out[4]: (25181752, 10)

In [11]: LD['state'] = pd.Categorical.from_array(LD.state).labels

C:\Users\dhruv\Anaconda3\lib\site-packages\ipykernel\__main__.py:1: FutureWarning: Categorical.from_array is deprecated, use Categorical instead
if __name__ == '__main__':
C:\Users\dhruv\Anaconda3\lib\site-packages\ipykernel\__main__.py:1: FutureWarning: 'labels' is deprecated. Use 'codes' instead
if __name__ == '__main__':

In [12]: LD['clicked'].value_counts()

Out[12]: 0    4845785
         1    990565
         Name: clicked, dtype: int64

In [14]: LD = LD.query("platform == platform")
y = LD['clicked']
cols = ['platform','state','document_id','traffic_source','mean_cf','ad_id']

X = LD[cols]
y = np.ravel(y)

In [16]: from collections import Counter
from sklearn.datasets import make_classification
from imblearn.over_sampling import SMOTE

smote = SMOTE(random_state=42,ratio=0.65)
X_res, y_res = smote.fit_sample(X, y)
print('Resampled dataset shape {}'.format(Counter(y_res)))

C:\Users\dhruv\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning: Function _ratio_float is deprecated; Use a float for 'ratio' is deprecated from version 0.2. The support will be removed in 0.4. Use a dict, str, or a callable instead.
warnings.warn(msg, category=DeprecationWarning)

Resampled dataset shape Counter({0: 4845785, 1: 2629760})

In [17]: FinalDataFrame = pd.DataFrame()
n = pd.DataFrame(X_res)
m = pd.DataFrame(y_res)
FinalDataFrame = pd.concat([n, m], axis=1)

FinalDataFrame.columns = ['platform','state','document_id','traffic_source','mean_cf','ad_id','clicked']
FinalDataFrame.head(2)

Out[17]:
```

	platform	state	document_id	traffic_source	mean_cf	ad_id	clicked
0	2.0	0.0	2245446.0	0.0	0.315267	182059.0	0
1	2.0	8.0	1899050.0	0.0	0.161156	116094.0	0

```


In [18]: FinalDataFrame['clicked'].value_counts()

Out[18]: 0    4845785
         1    2629760
         Name: clicked, dtype: int64

In [ ]: FinalDataFrame.to_csv('Prediction_final_file3.csv')
```

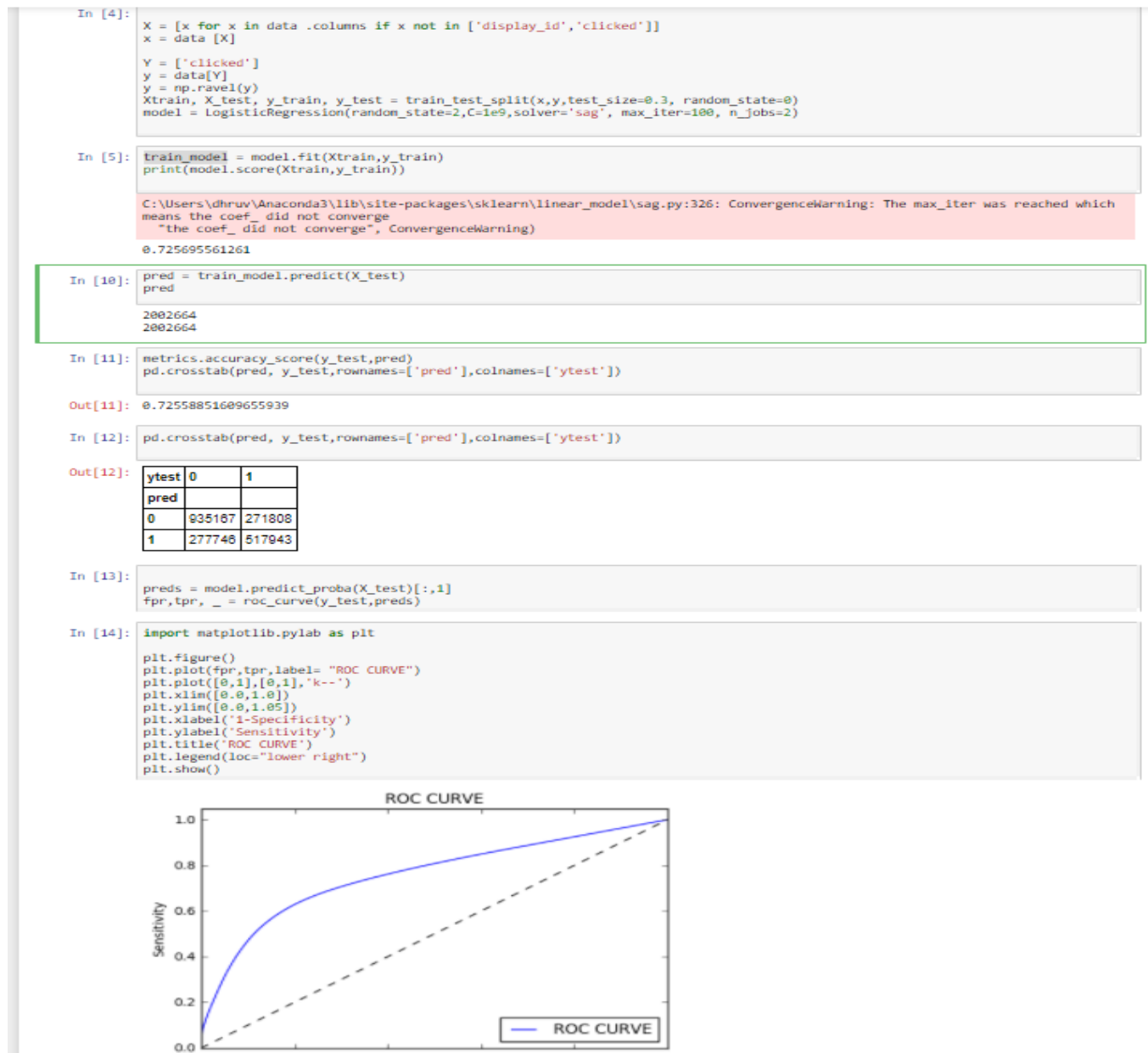
FROM the highlighters we can confirm that we have solved the minority class problem and now we class considerable number to start modeling.

3.5 MODELING:

I've used python, R and Azure Machine learning to select the best performing model. Logistic Regression has been implemented in Python, Neural Network in R , SVM and Naïve Bayes on Azure Machine learning.

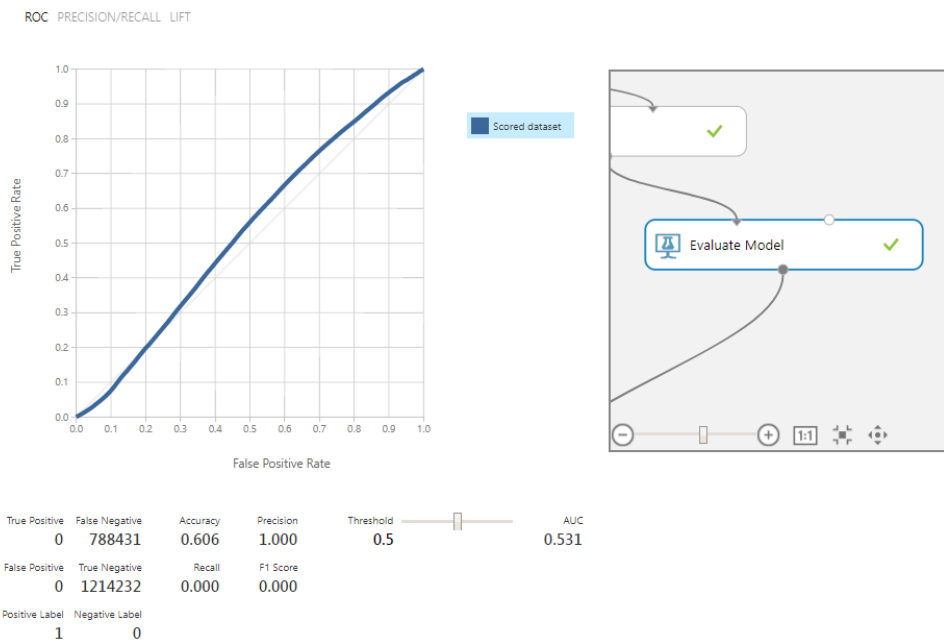
1. Logistic Regression:

After implementing logistic regression in python the accuracy we attained was around 72%. But, we didn't take accuracy as the only metric to select the best model so we also looked at Confusion matrix. Heres the screenshot that shows accuracy and Confusion matrix along with the model.



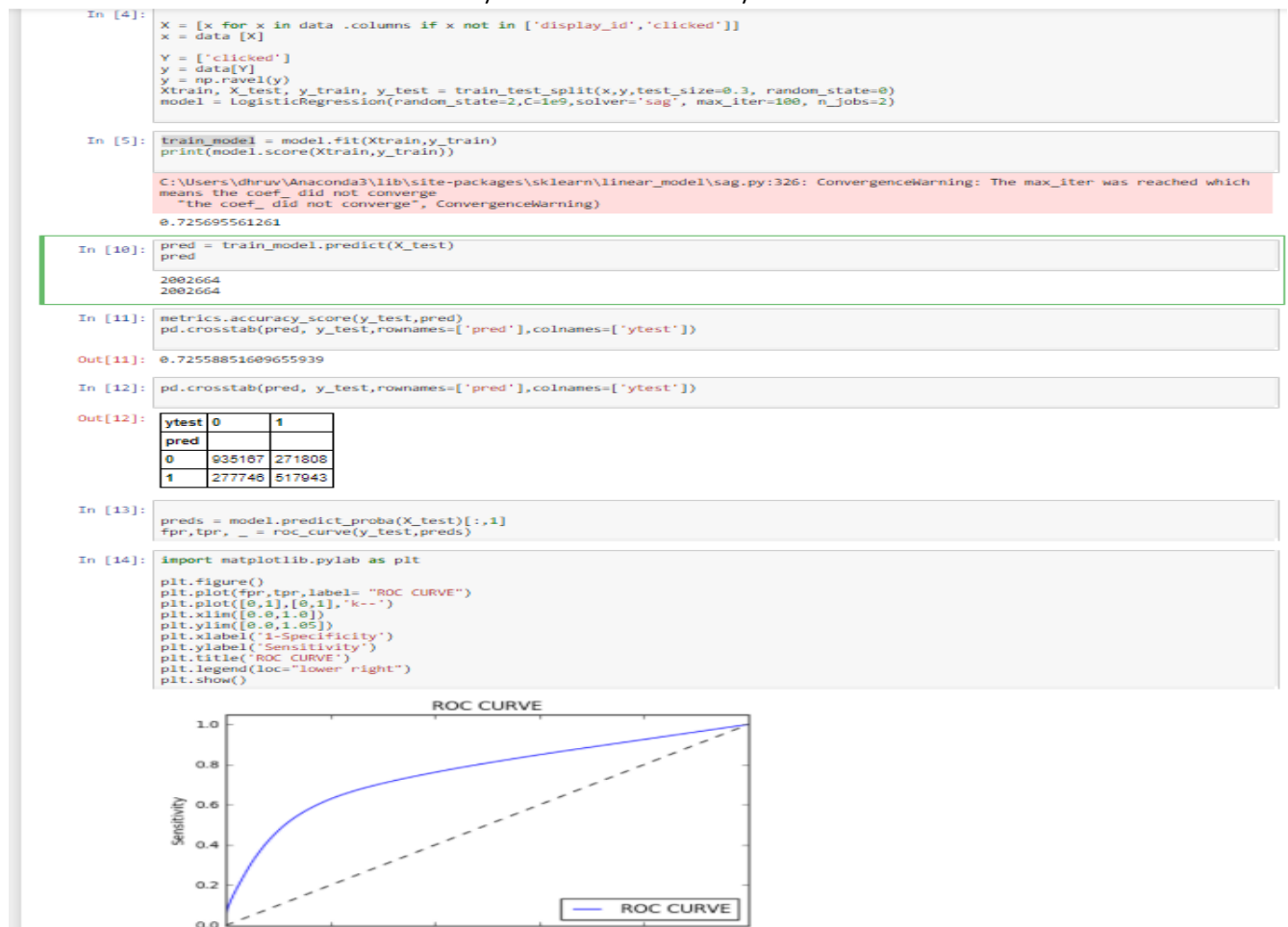
Here's a screenshot of performance metrics of Logistic regression in Azure ML

Experiment created on 12/16/2017 > Evaluate Model > Evaluation results



2. Random Forest:

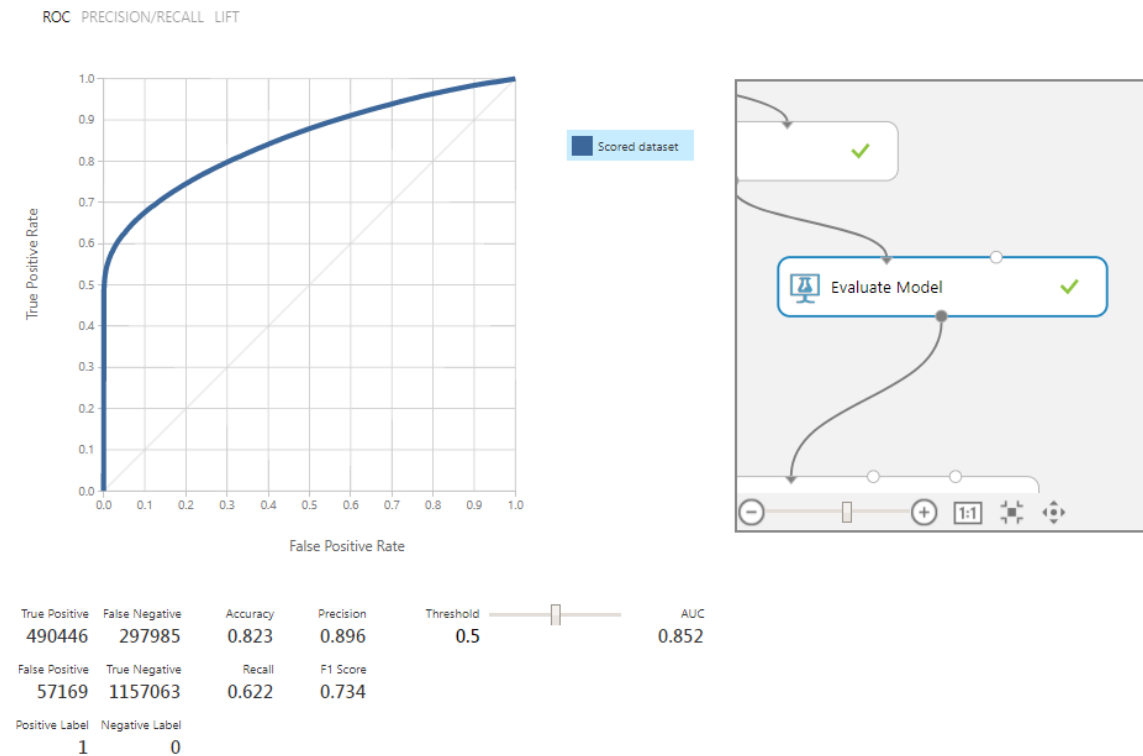
Here's the screenshot that shows the accuracy and confusion matrix by Random Forest.



IT DEFINITLY GIVES BETTER RESULT THAN LOGISTIC REGRESSION

The following screenshot shows performance metrics of Decision Forest in AZURE ML

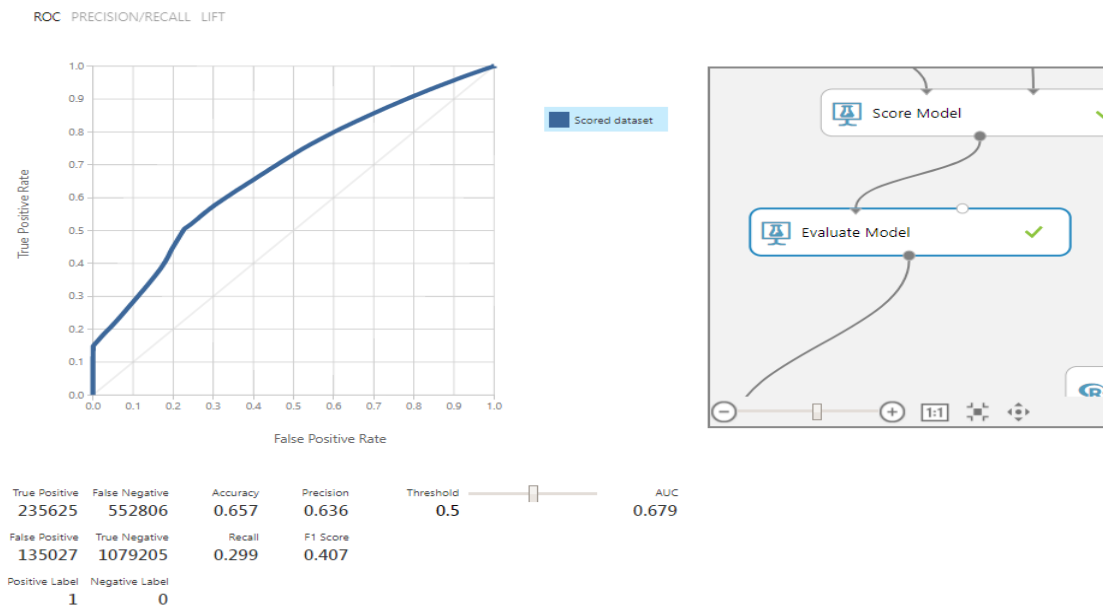
Experiment created on 12/16/2017 > Evaluate Model > Evaluation results



3. Neural Network:

I tried implementing nn in both R and Python but ran out of memory issues. So, then I implemented it on Azure Machine Learning.

Experiment created on 12/16/2017 > Evaluate Model > Evaluation results

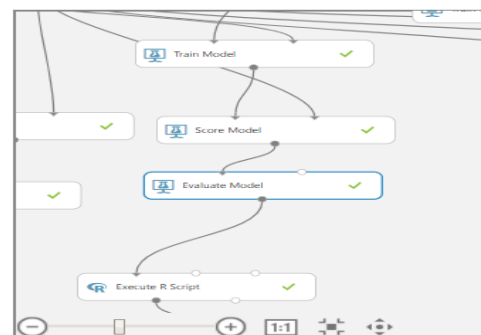
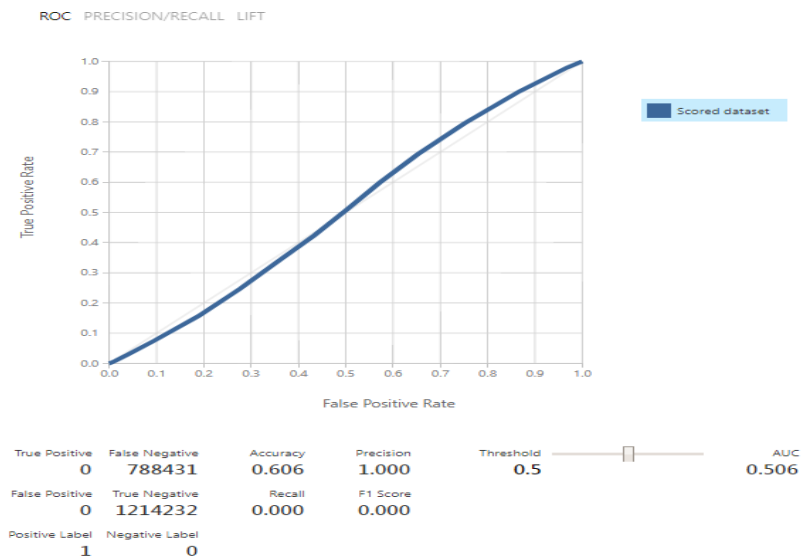


Here's the screenshot of NN metrics after implementing it in Azure ML.

4. SVM

Here's SVM implemented on Azure Machine Learning

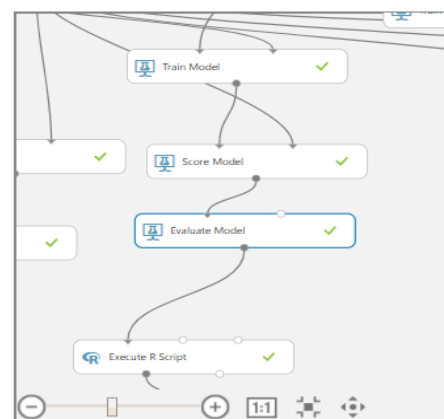
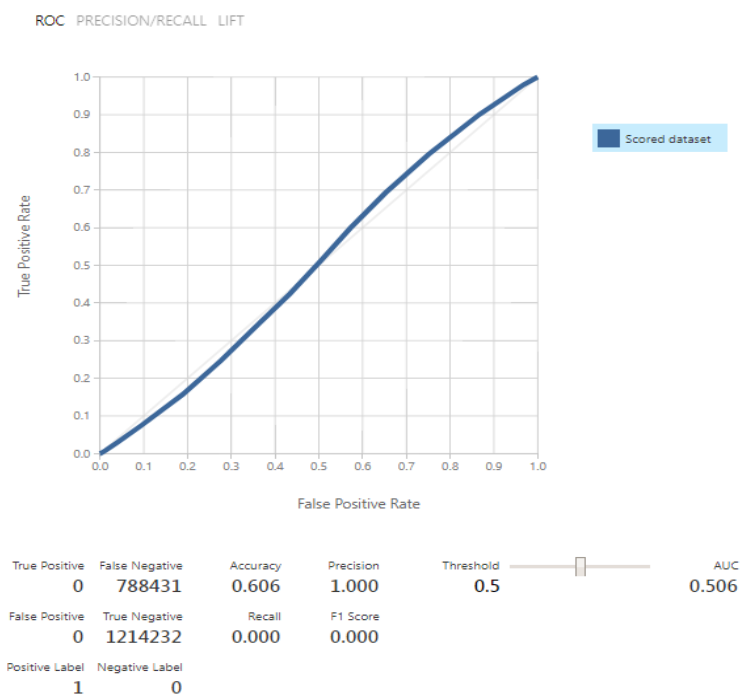
Experiment created on 12/16/2017 > Evaluate Model > Evaluation results



5. Naïve Bayes

Heres the screenshot of performance metrics of Naïve Bayes algorithms.

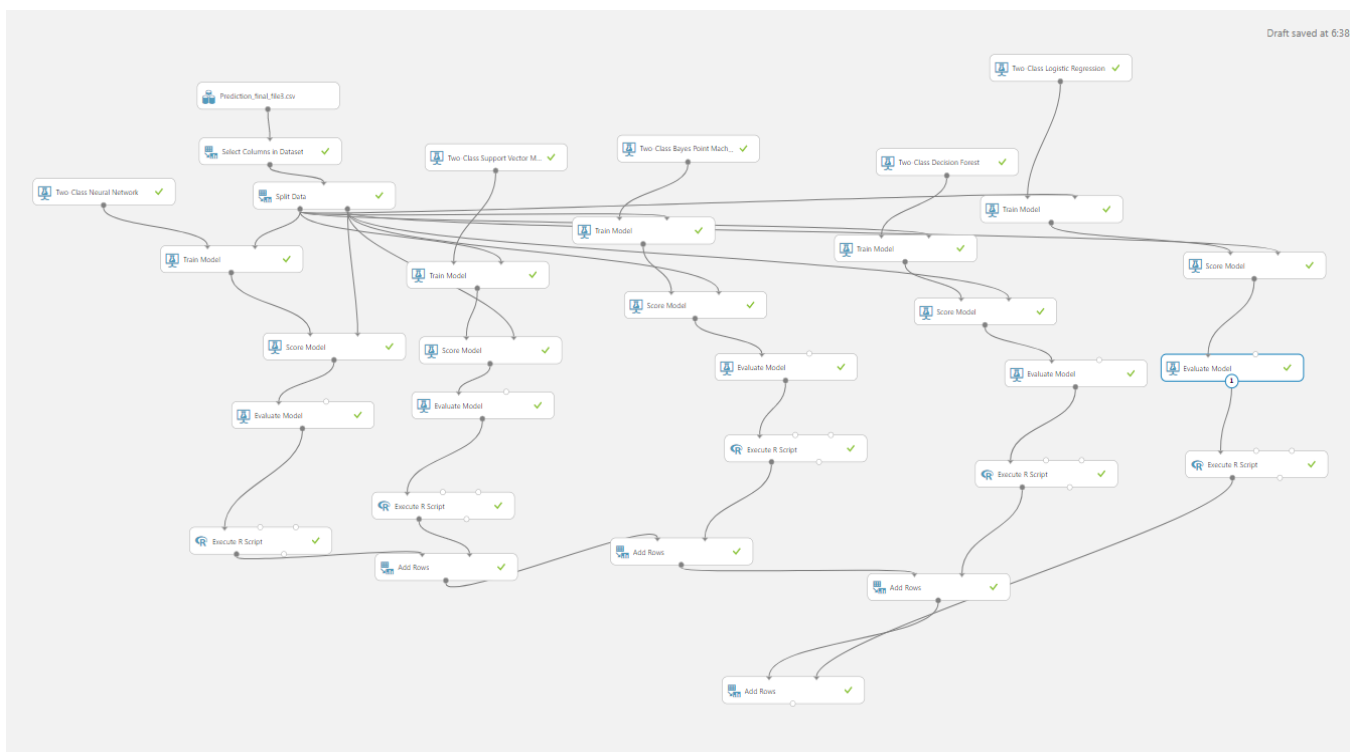
Experiment created on 12/16/2017 > Evaluate Model > Evaluation results



3.6 COMPARISON OF MODELS:

The following screenshot shows comparison of all the models mentioned above so that we can select the best performing model.

Here's a screenshot for it:



Now, Looking at Performance metric we select the best model out of all the mentioned algorithm/.

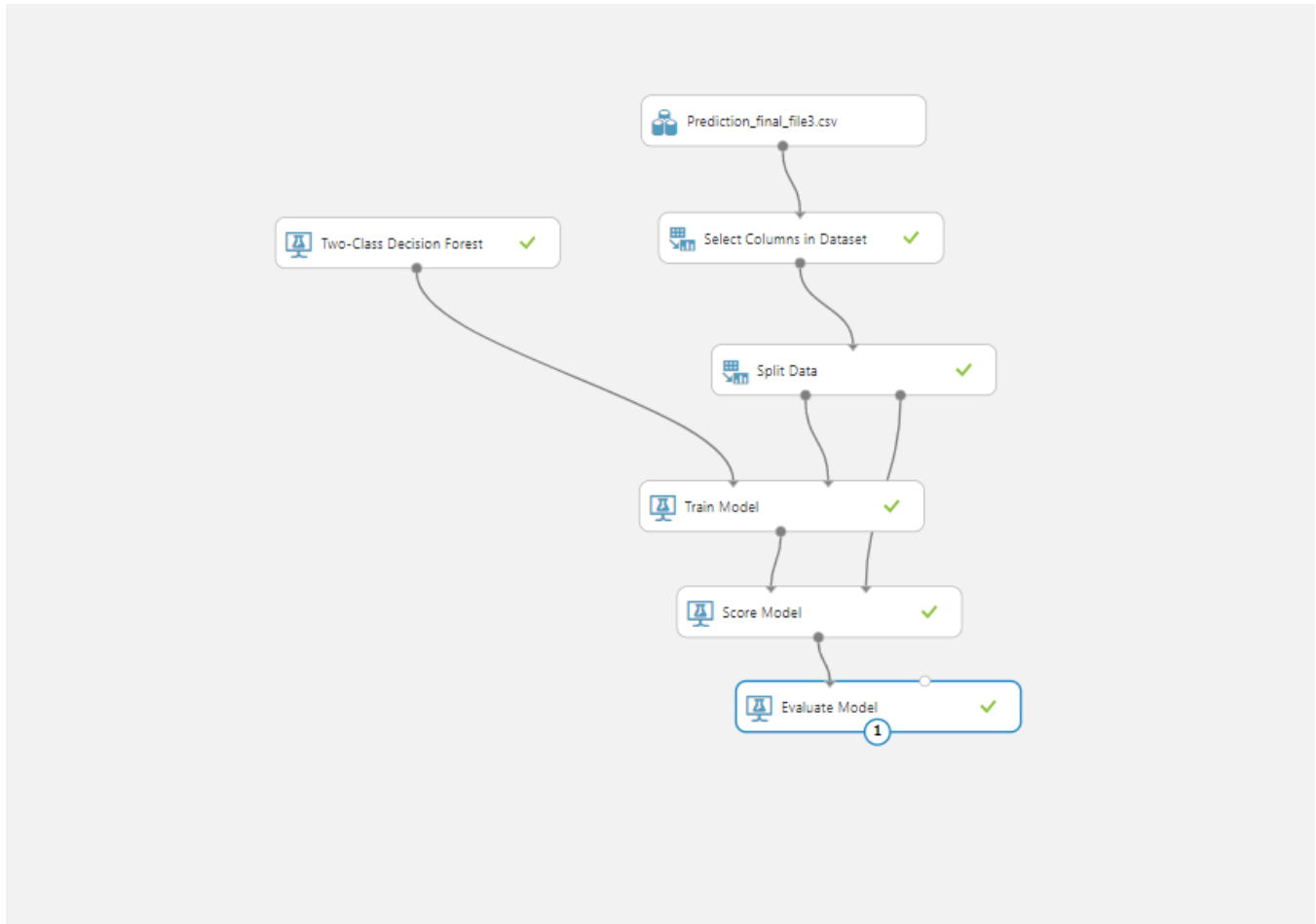
Experiment created on 12/16/2017 > Add Rows > Results dataset

rows	columns								
5	8								
	Accuracy	Precision	Recall	F-Score	AUC	Average Log Loss	Training Log Loss	Algorithm	
view as									
	0.656541	0.635704	0.298853	0.406571	0.679498	0.608814	9.182434	NeuralNetwork	
	0.606309	0	0	0	0.506093	0.670354	0.002428	SVM	
	0.606309	0	0	0	0.525606	0.669566	0.119987	BAYES	
	0.822659	0.895604	0.622053	0.734175	0.851854	0.498492	25.639334	Decision Forest	
	0.606309	0	0	0	0.530922	0.669491	0.131186	NeuralNetwork	

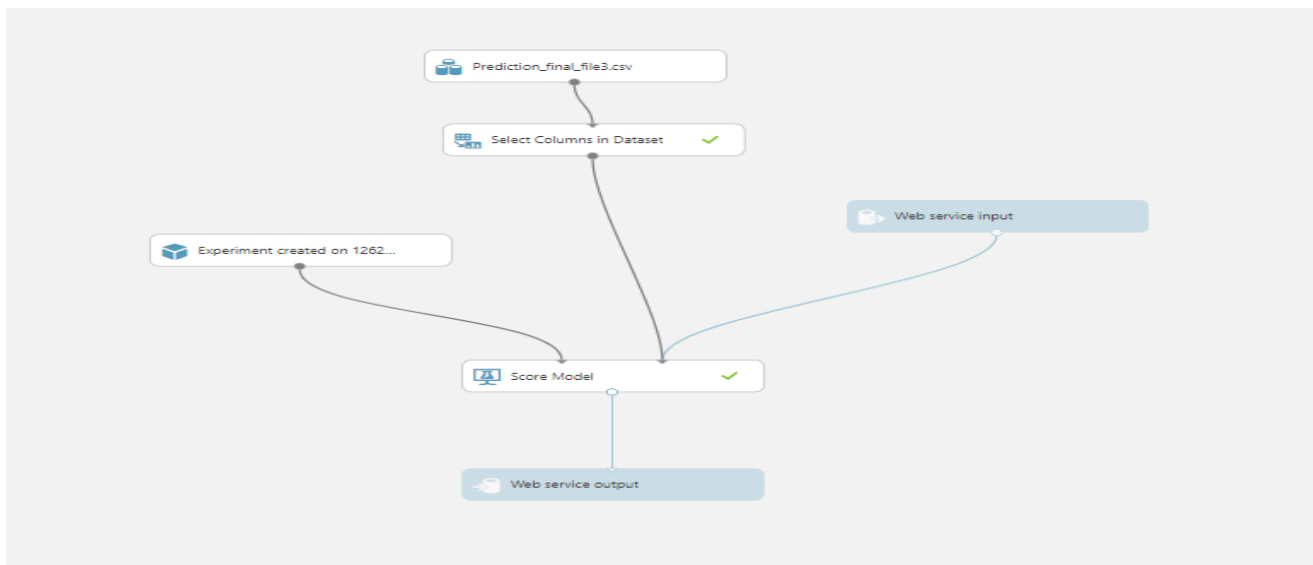
Based upon various metrics like confusion matrix, Training Log loss, Precision and Accuracy Decision Forest looks like the best one.

3.7 Deploying Decision Forest on AZURE ML:

Here's the training experiment of Decision Forest



Here's the screenshot of Deployed Two Class Decision model :



3.8 FLASK APPLICATION:

Here's a screenshot of the application that is running on local

← → ↻ 127.0.0.1:5000

Apps Y Combinator TeamWork.Enterprise TechCrunch BI Business Insider Techmeme Blackboard Top Data Science Onl https://dreamtolearn Data-Driven Pi

Please give your Input Details to know if an advertisement will be clicked or not

platform :

state:

Document ID: :

Traffic Source :

Mean Cf :

Ad Id :

Here's the running demo of the application running on local:

← → ↻ 127.0.0.1:5000

Apps Y Combinator TeamWork.Enterprise TechCrunch BI Business Insider Techmeme Bb Blackboard Top Data Science Onl https://dreamtolearn. PC

Please give your Input Details to know if an advertisement will be clicked or no

platform :

state:

Document ID: :

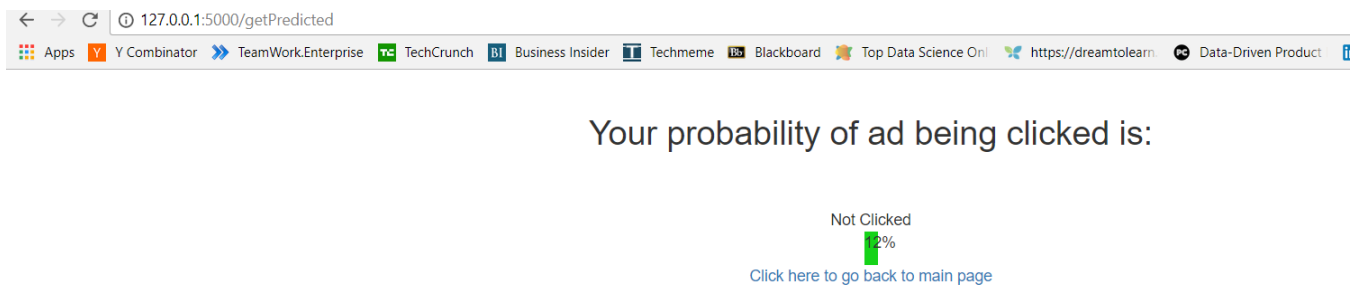
Traffic Source :

Mean Cf :

Ad Id :

[Submit](#)

HERE's the output:



Screenshots of Code used in Pycharm

```

import ssl
# If you are using Python 3+, import urllib instead of urllib2
import urllib.request
import urllib2
from urllib2 import HTTPError
import json
import os.path

from flask import Flask, Response, request, render_template, send_from_directory

app = Flask(__name__)
tmpl_dir = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'templates')
port = int(os.getenv('PORT', 8080))


@app.route("/")
def welcome():
    print(tmpl_dir)
    return render_template('clicksAffectedAssessmentIndex.html')

def root_dir(): # pragma: no cover
    return os.path.abspath(os.path.dirname(__file__))

def get_file(filename): # pragma: no cover
    try:
        src = os.path.join(root_dir(), filename)
        return open(src).read()
    except IOError as exc:
        return str(exc)

@app.route('/getPredicted', methods=['POST'])
def getPredictedDetails():
    data = {
        "Inputs": {
            "input1":

```

 **Platform and Plugin Updates**
PyCharm Community Edition is ready to [update](#).

```

getPredictedDet...

37
38     "input1":
39         {
40             "ColumnNames": ["platform", "state", "document_id", "traffic_source", "mean_cf", "ad_id"],
41             "Values": [ (request.form.get("platform")),
42                         (request.form.get("state")),
43                         (request.form.get("document_id")),
44                         (request.form.get("traffic_source")),
45                         (request.form.get("mean_cf")),
46                         (request.form.get("ad_id")),
47                         ], ],
48         }, },
49     "GlobalParameters": {
50     }
51 }
52
53 body = str.encode(json.dumps(data))
54
55
56 url = 'https://ussouthcentral.services.azureml.net/workspaces/bad877e669504b97a7e55d59b9bec551/services/a60ad87d7f1b4e2f89f6c54e77308380/execute'
57 #url = 'https://ussouthcentral.services.azureml.net/workspaces/bad877e669504b97a7e55d59b9bec551/services/a60ad87d7f1b4e2f89f6c54e77308380/execute'
58 api_key = 'Xzzd0kYb1H0/F9p6aMzon7k9Lo5MKQ1ZwNWy1h8LUQUB+PX7RqRiJlTl0cOB8zh4Mw1XFHzYhvzCyNjMgqqRvw==' # Replace this with the API key for the workspace
59 headers = {'Content-Type': 'application/json', 'Authorization': ('Bearer ' + api_key)}
60 #context = ssl.create_unverified_context()
61 req = urllib.request.Request(url, body, headers)
62 #req = urllib2.Request(url, body, headers)
63 # try:
64 print(url)
65 response = urllib.request.urlopen(req)
66
67
68 #response = urllib2.urlopen(req)
69 # If you are using Python 3+, replace urllib2 with urllib.request in the above code:
70 # req = urllib.request.Request(url, body, headers)
71 # response = urllib.request.urlopen(req)
72
73 result = response.read().decode('utf-8')
74 # result= json.loads(response.read())

#response = urllib2.urlopen(req)
# If you are using Python 3+, replace urllib2 with urllib.request in the above code:
# req = urllib.request.Request(url, body, headers)
# response = urllib.request.urlopen(req)

result = response.read().decode('utf-8')
# result= json.loads(response.read())
# for key, value in dict.items(result["ColumnNames"]):
#     print(key, value)
jsonList=json.loads(result)
values=jsonList['Results']['output1']['value']['Values']
print(result)
print(values)
prediction=float(values[0][7])
#prediction = 0.68
if (prediction > 0.5):
    resultStatus="Clicked"
    print(values[0][7])
else:
    resultStatus = "Not Clicked"
print(resultStatus)
# except urllib.error as error:
#     print("The request failed with status code: " + str(error.code))
#
#     # Print the headers - they include the request ID and the timestamp, which are useful for debugging the failure
#     print(error.info())
#
#     print(json.loads(error.read()))

#return send_from_directory('js', path)
#return render_template('UserDashboard.html', resultStatus=resultStatus)
prediction=round(prediction,2)*100
print(prediction)
return render_template('clickedFactors.html', prediction=prediction, resultStatus=resultStatus)

if __name__ == "__main__":
    app.run()
#app.run(host='0.0.0.0', port=port, debug=True)

```

Platform and Plugin Updates

PyCharm Community Edition is ready to [update](#).

Platform and Plugin Updates

PyCharm Community Edition is ready to [update](#).

4.1 CHALLENGES:

1. I've never web scraped dataset after logging in. So that was a challenging task in the beginning but requests library documentation was enough to walk me through it
2. Uploading data on azure blobs was something I was trying for the first time as I've used Amazon S3 previously. The UI is bit complex including the process but there was a good article online that helped me pass through it and I could eventually successfully load the data
3. Next major challenge was dealing with the class imbalance problem. After thorough research I came across SMOTE that is a type of oversampling and it could solve this problem of imabalancing
4. Since, I was using Azure ML for the first time understanding how it functions how models work took few days to get me going. But, then I found the concept really interesting and is easy to use for building models.
5. Another challenge in the Azure ML part was how to compare all the models from one experiment itself. After thorough research online and looking at different components I came up with a manual code and functions to get a csv of files together so that we can directly compare them and select the best model.
6. After selecting Decision Class as the best model, deploying the models on Azure ML was something new but the MSDN docs were good enough to walk pass it
7. I was trying my hands on Flask application(PyCharm) for the first time so this was an interesting part integrating my model with the application through the REST API. This wasn't as challenging as it seemed before. The major problem was getting the application ready and hosting it online
8. I tried hosting my application on IBM Bluemix but I couldn't find my way out some memory issue. I also tried using Google Cloud platform hosting platform but due to time constraints couldn't set it up on GCP

4.2 CONCLUSION:

It was indeed a good project to work on with major focus being on the deployment and cloud part. Having worked on modeling before the major emphasis was to gain hands on tools that are used for deployment of models and Azure ML proved to be a good resource contributing to my learning curve. Azure ML not only makes modeling computation faster as it uses its own servers for training and testing but it also makes the deployment easier as compared to methods that are used in python. Also, the Azure Blob storage looked like a good storage option after S3 and I could also add it many of different technologies I worked upon during the project. PyCharm and application that is based on Python and Flask framework was a difficult task to deal with as I have never worked with any of the web technologies before. So, linking the HTML pages and setting up the links was indeed a good learning. The only disappointment with this project is I couldn't host it online otherwise it could have been a complete project to submit.

4.3 REFERENCES:

- <http://www.aiswaryas.com/outbrain-click-prediction.html>
- Y. W. Chang, C. J. Hsieh, K. W. Chang, M. Ringgaard, and C.-J. Lin, "Training and testing lowdegree polynomial data mappings via linear SVM," Journal of Machine Learning Research, vol. 11, pp. 1471–1490, 2010

- B. McMahan, G. Holt, D. Scully , “Ad Click Prediction: a View from the Trenches”
- Y. Juan, Y. Xuhang, W. Chin, “Field-aware Factorization Machines for CTR Prediction”
- <https://www.kaggle.com/franckjay/easy-random-forests>
- <http://cs229.stanford.edu/proj2016/report/JaiswalGopinathLimaye-OutbrainClickPrediction-report.pdf>
- <https://www.slideshare.net/AlexeyGrigorev/outbrain-click-prediction-71724151>
- <https://www.kaggle.com/anokas/outbrain-eda>
- <https://docs.microsoft.com/en-us/azure/machine-learning/preview>
- <https://docs.microsoft.com/en-us/azure/machine-learning/preview/tutorial-bikeshare-dataprep>
- <https://www.jetbrains.com/pycharm/documentation/>

THANK YOU