CIS5560 Term Project Tutorial

Authors: Anjali Baldawa; Dhruvi Patel; Verusca Capuno; Digeshkumar Kansara

Instructor: Jongwook Woo

Date: 05/17/2020

Lab Tutorial

Anjali Baldawa (abaldaw@calstatela.edu)

Dhruvi Patel(dpatel86@calstatela.edu)

Verusca Capuno(vcapuno@calstatela.edu)

Digeshkumar Kansara(dkansar@calstatela.edu)

05/17/2020

Outbrain Ad Prediction using Spark

Objectives

In this hands-on lab, you will learn how to:

- Get data from Kaggle
- Sampling data using Jupytar Notebook and Pandas
- Upload data, create Experiment, join data, feature engineering, train, test data and measure accuracy using different algorithm in Azure ML Studio
- Upload data, create cluster, use notebook and Pyspark file, create schema, join data, cross validation, tuning parameter, train, test data, measure accuracy of algorithm using different measures and using different algorithms in data bricks.

- Upload data, use Pyspark file, create schema, join data, cross validation, tuning parameter, train, test data, measure accuracy of algorithm using different measures and using different algorithms In Oracle BDCE.

Platform Specification

Azure ML

- Free workspace
- 10 GB storage
- Single node

Data Bricks community Edition

- Databricks subscription
- Cluster 6.5 ML (includes Scala 2.11, Spark 2.4.5)
- 15.3 GB Memory, 2 Cores, 1 DBU
- Python Version 3

Oracle BDCE

- Cluster 5.2 (includes Apache Spark 2.4)
- Python Version 3
- Cluster size 1000.3 GB
- Number of Processors 32
- Memory Size 247.625312GB
- CPU Speed 2.20 GHz

Step 1: Get data from Kaggle Website

1) The following steps explain how to download data from Kaggle Website.

   - To download Outbrain ad click prediction data we need to navigate to Kaggle website using following link.
     https://www.kaggle.com/c/outbrain-click-prediction/data

     Note: Following are additional steps to download data.

     1. Navigate to Rule Tab from Top menu.
     2. Press I Understand Button.
     3. Enter your Phone number.
     4. It will send you send digit password.

5. Enter password and Now you are ready to download data.

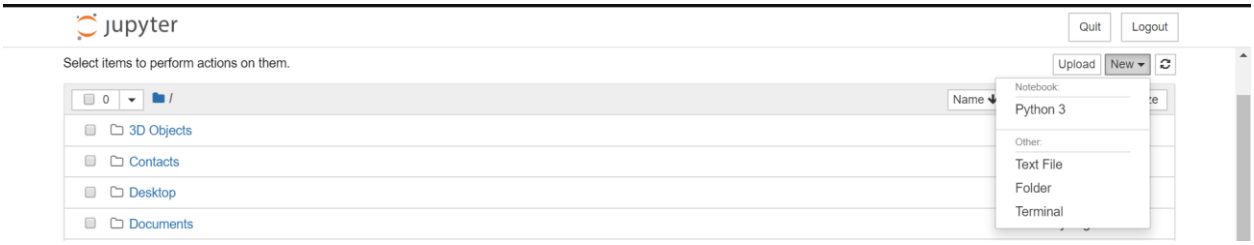
- Now move to data tab from Top menu and download .csv files.

Note: The size of the files is huge so you need to have a good internet connection to download the dataset. Approximately, it would take 1 hour to download 1 file. The download time depends on the internet speed. Once you download files into your local pc, you need to unzip the file in your local system.

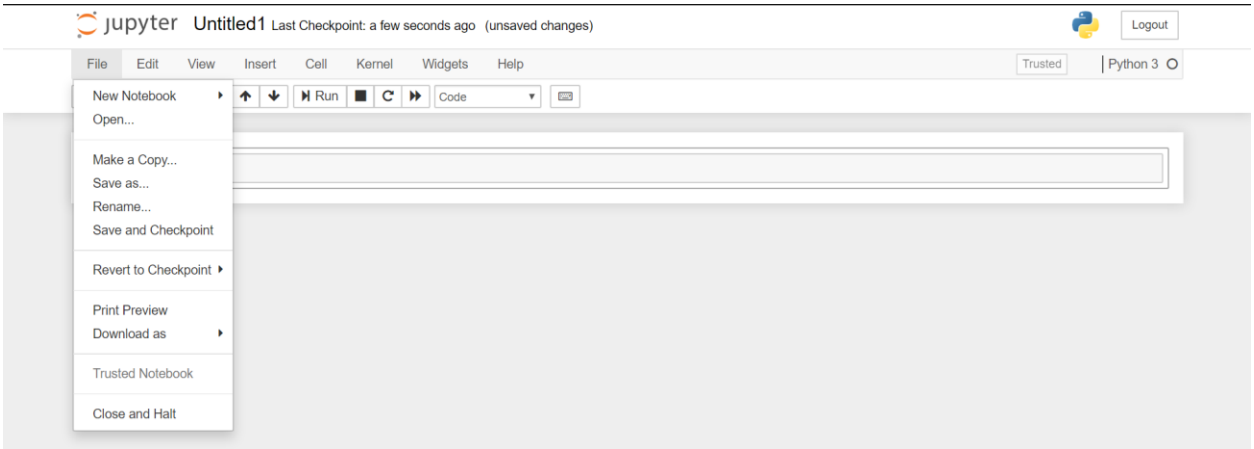
Step 2: Sample Data using Jupyter Notebook

To sample data in smaller CSV file We need have Python3 and Anaconda. We are using pandas library to sample dataset.

1. Download Python.
    1. Download Python from following link: https://www.python.org/downloads/
    2. You need to choose proper version according to your system requirement MAC OS, Windows or Linux.
    3. Press Downloads button (Current latest version is 3.8.2)
    4. After Downloading it is simple install on your system.
2. Download Anaconda. (Minimum 4GB, Recommended 8GB)
    1. Download Anaconda from following link: https://www.anaconda.com/products/individual
    2. Scroll Down to Bottom.
    3. Choose your Operating system and download appropriate version.
    4. It will take l little while as size is 465 MB
    5. Unzip downloaded anaconda file.
    6. Install on your system as it is simple steps just press next and in last press finish button.
3. Launch Jupyter notebook and sample data.
    1. Start Anaconda in your system.
    2. You will see jupyter notebook option and press Launch button it will open in your default browser. It will open following screen for you. From right side top pane click on new and select Python which will create new notebook with python3.



It will open following window. From file choose **open** option.

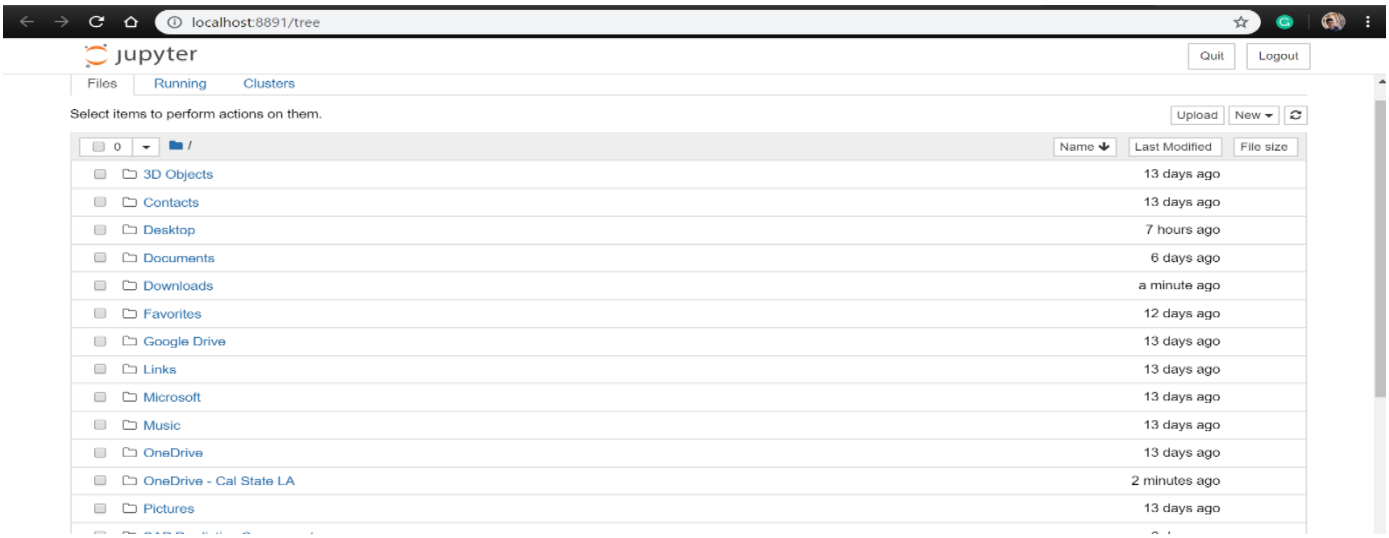


It will show you all directories in your system.

Now you need to download **Project_sample.ipynb** file provided to you.

Now navigate to directory where your file is located and press open.

It will open Notebook file for you.



Then follow steps in file using which you can easily sample data.

Step 3: Azure ML Studio

In step 2, we sampled data using jupyter notebook. Now we have our sampled files (Around 40 MB) to import in Azure ML studio.

Create an Azure ML Account

Azure ML offers a free-tier account, which you can use to complete the experiment.

Sign Up for a Microsoft Account

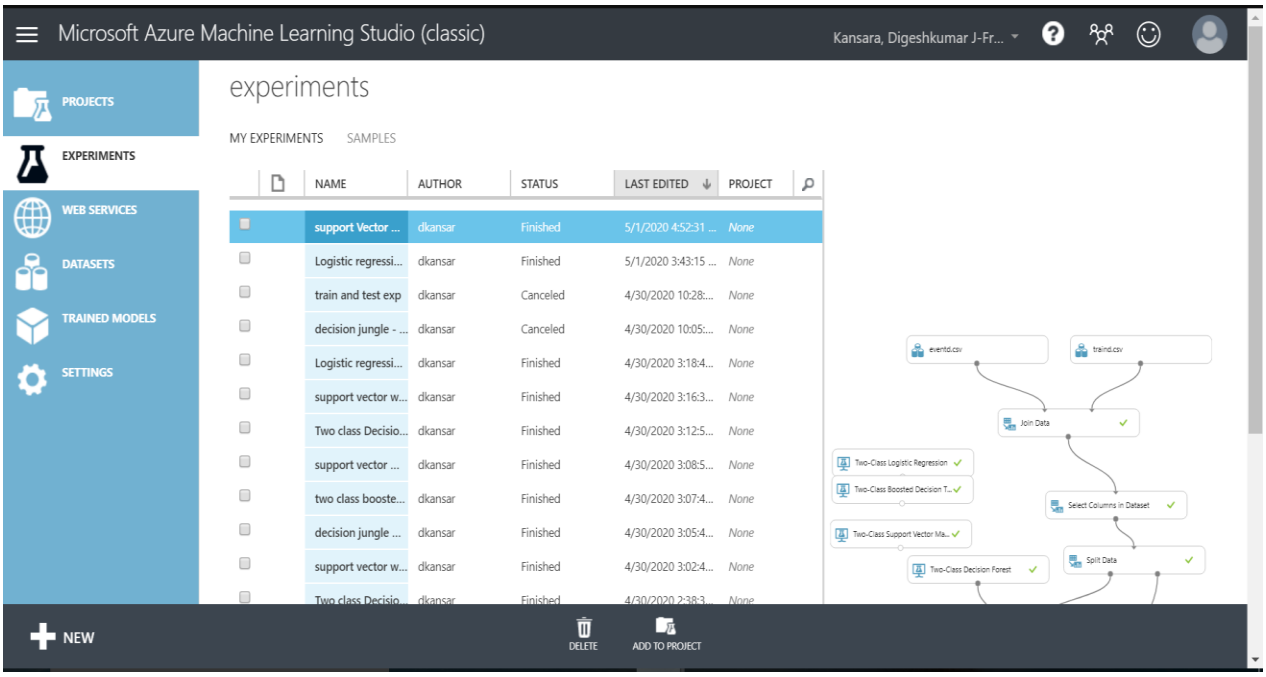
1. If you do not already have a Microsoft account, sign up for one at https://signup.live.com/. You don't need to use your school email account to sign up but you can use any email account.

Sign Up for a Free Azure ML Account

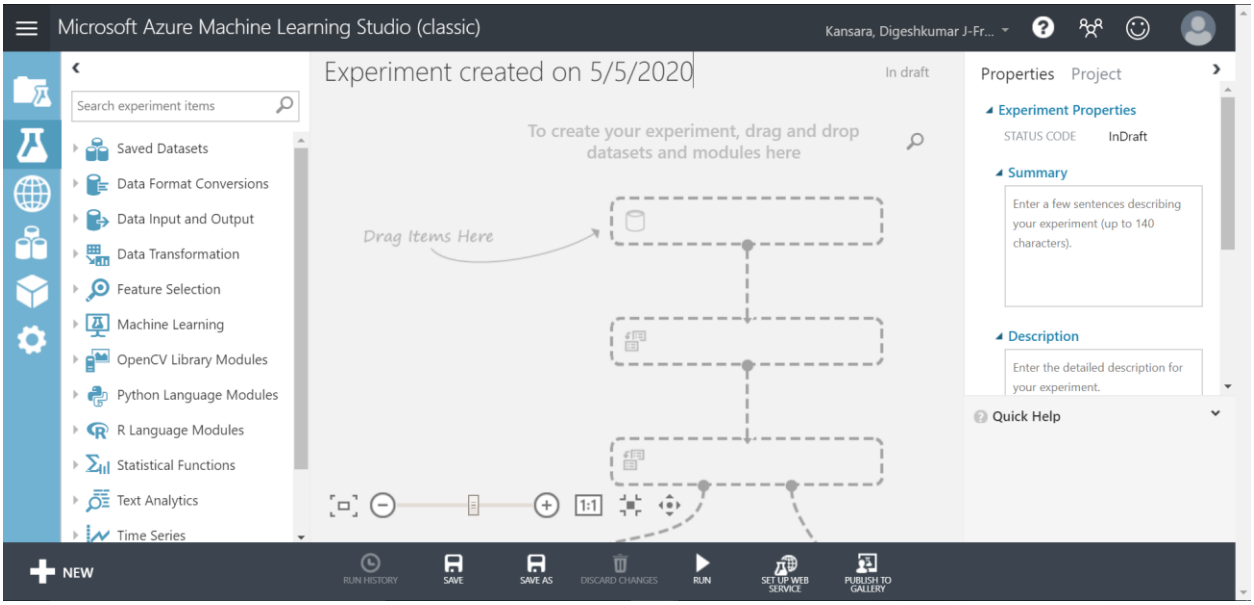
1. Browse to https://azure.microsoft.com/en-us/services/machine-learning/ login and click Get started now
2. When prompted, choose the option to sign in, and sign in with your Microsoft account credentials.
3. On the **Welcome** page, watch the overview video if you want to see an introduction to Azure ML Studio. Then close the **Welcome** page by clicking the checkmark icon.

Create Experiment and Upload Data Set (CSV files) in ML studio

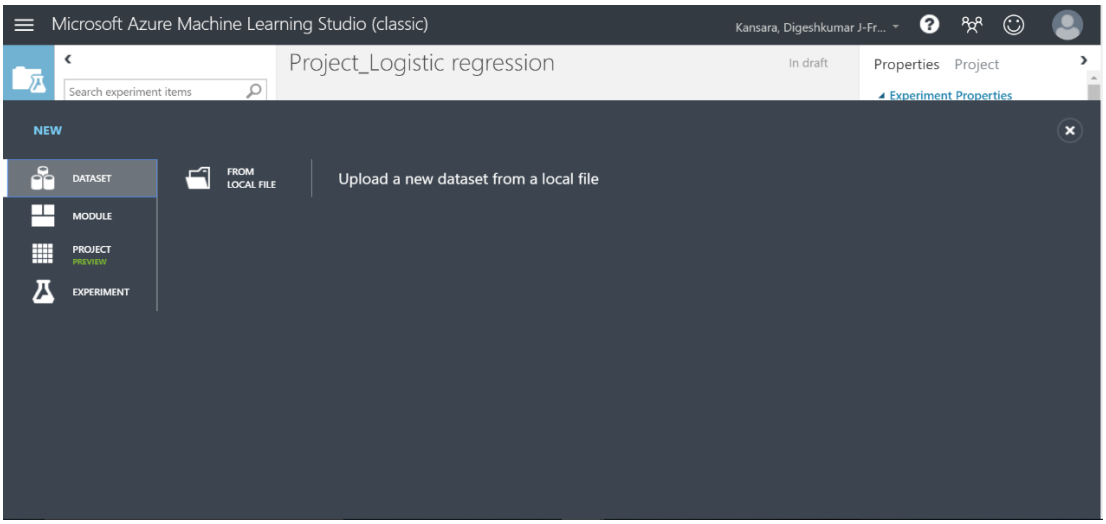
1. You should now be in Azure ML Studio with the Experiments page selected, which looks like the following image (if not, click the Studio tab at the top of the page).



2. In the Studio, at the bottom left, click NEW. Then in the collection of Microsoft samples, select
Blank Experiment. This creates a blank experiment, which looks similar to the following image.

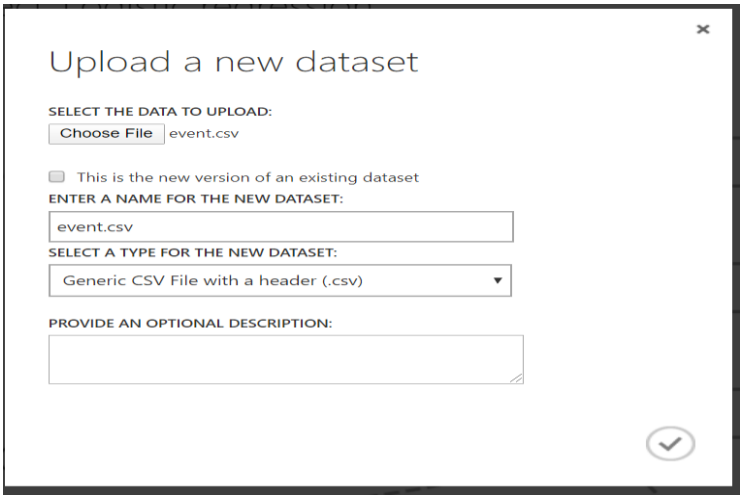


3. Change the title of your experiment from “Experiment created on “today’s date” to “Project_Logistic regression”.
4. Now we have to upload the dataset for that we have to click NEW at the bottom left. Then in the NEW dialog box, click the DATASET tab as shown in the following image.



5. Click FROM LOCAL FILE then in the upload a new dataset dialog box, browse to select the eventd.csv file from the folder where you extracted the lab files on your local computer and enter the following details as shown in the image below, and then click the OK icon.
   - This is a new version of an existing dataset: Unselected
   - Enter a name for the new dataset: eventd
   - Select a type for the new dataset: Generic CSV file with a header (.csv)
   - Provide an optional description: Diabetes patient appointments.

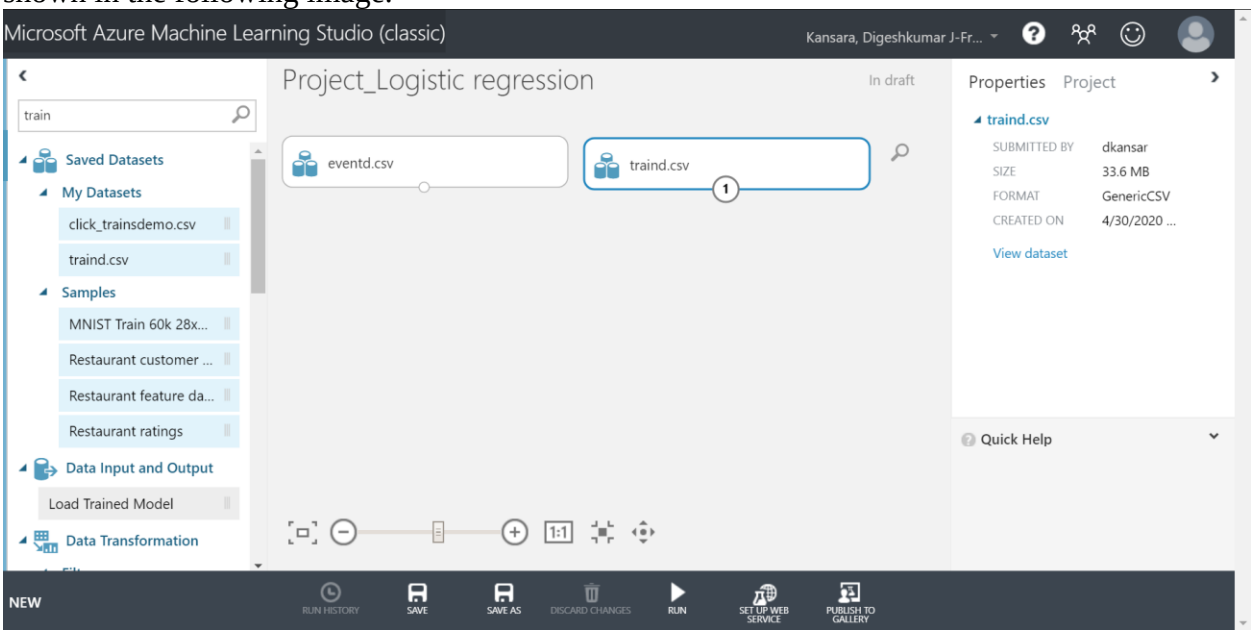
Note: May be file names are different in your case. Depending on what you gave during sampling of data in jupyter notebook.



6. Wait for the upload of the dataset to be completed, and then on the experiment items pane, expand Saved Datasets and My Datasets to verify that the eventd dataset is listed.
7. Follow step 4-6 to upload traind and Promoted_content.csv file.

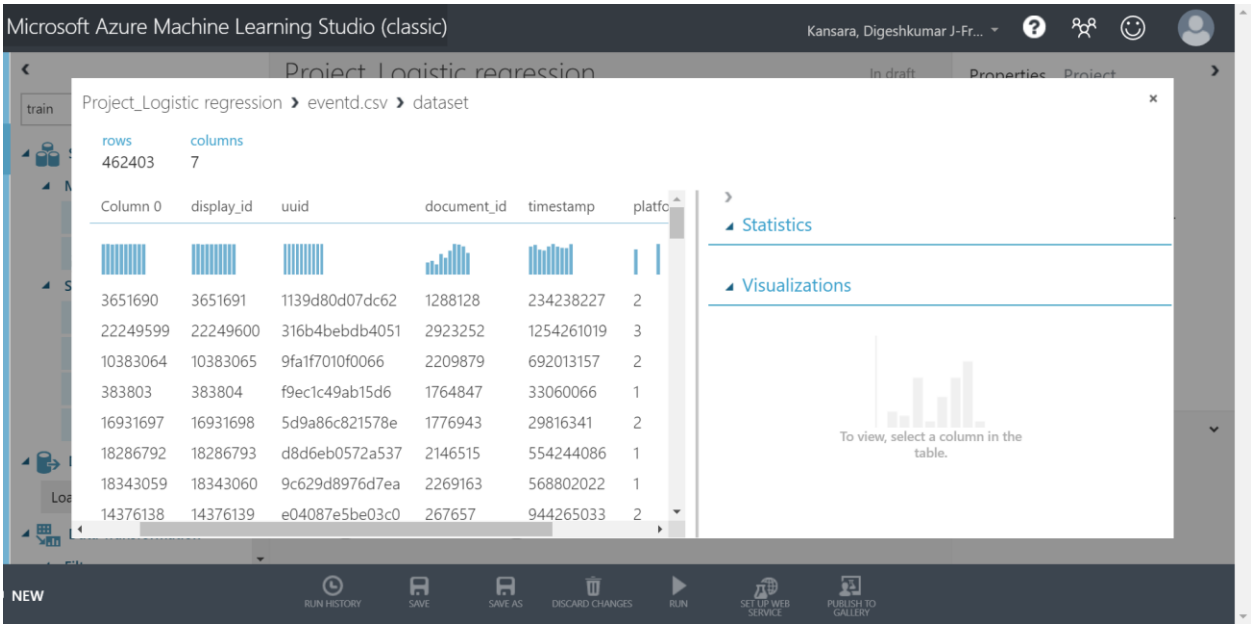
Create Experiment with and joining three CSV files into one.

1. In the experiment items pane on the left, expand Saved Datasets, expand Samples, and drag eventd and traind (One by One) to the experiment canvas in the middle of the page, as shown in the following image.

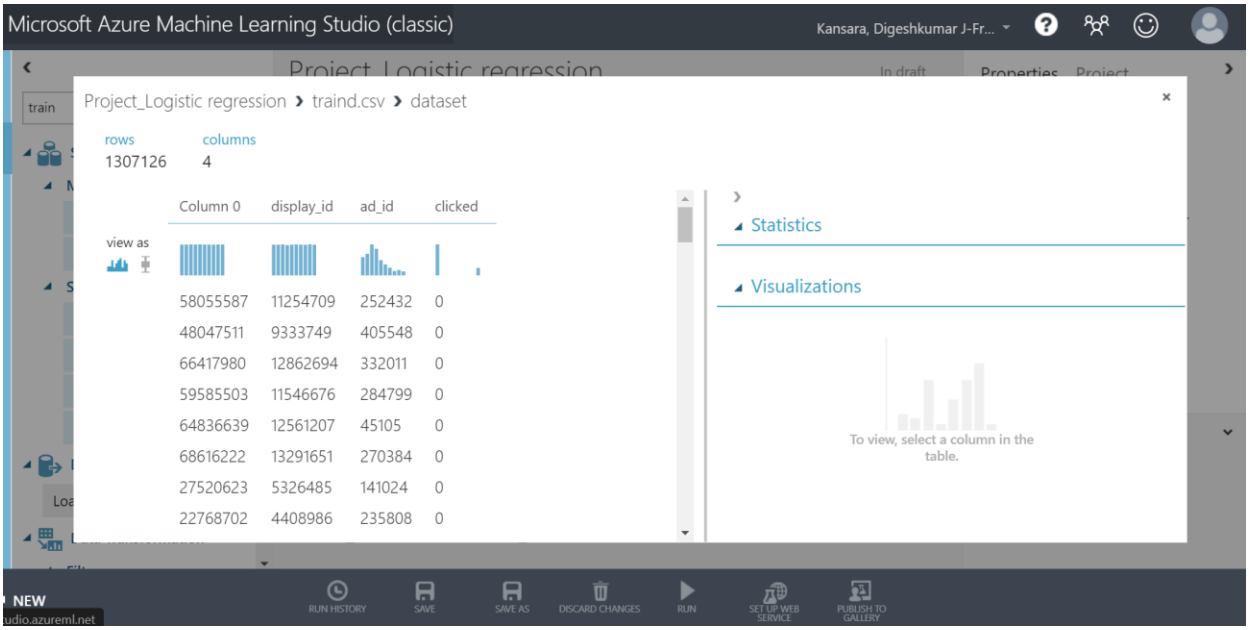


2. Right click on eventd, traind data set and click visualize to view the data in dataset.

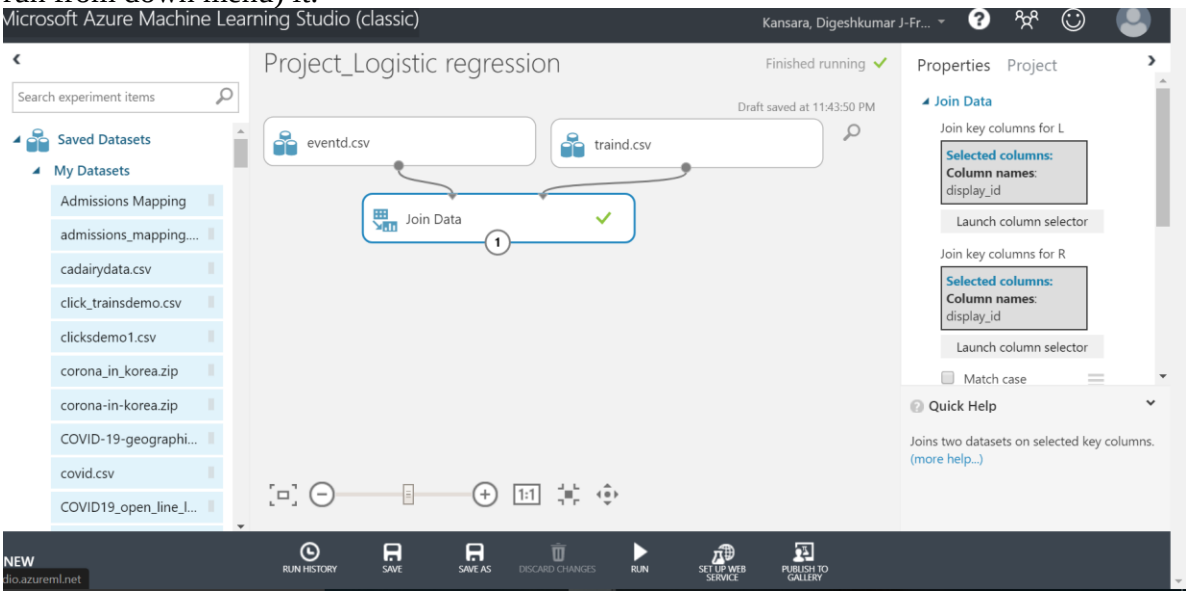
Eventd:



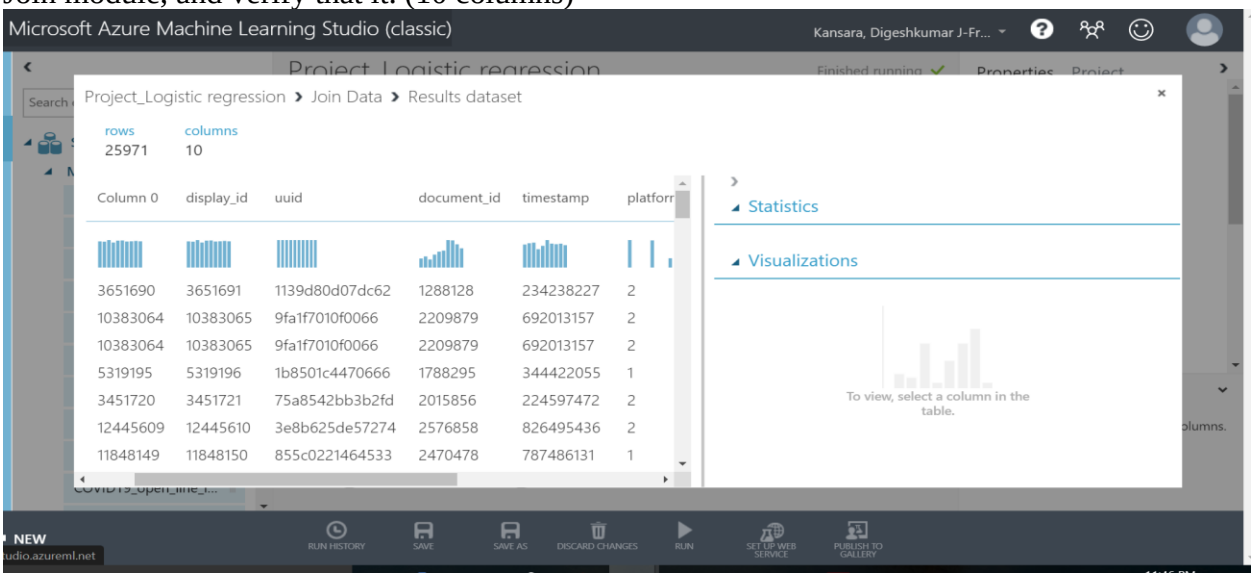
Traind:



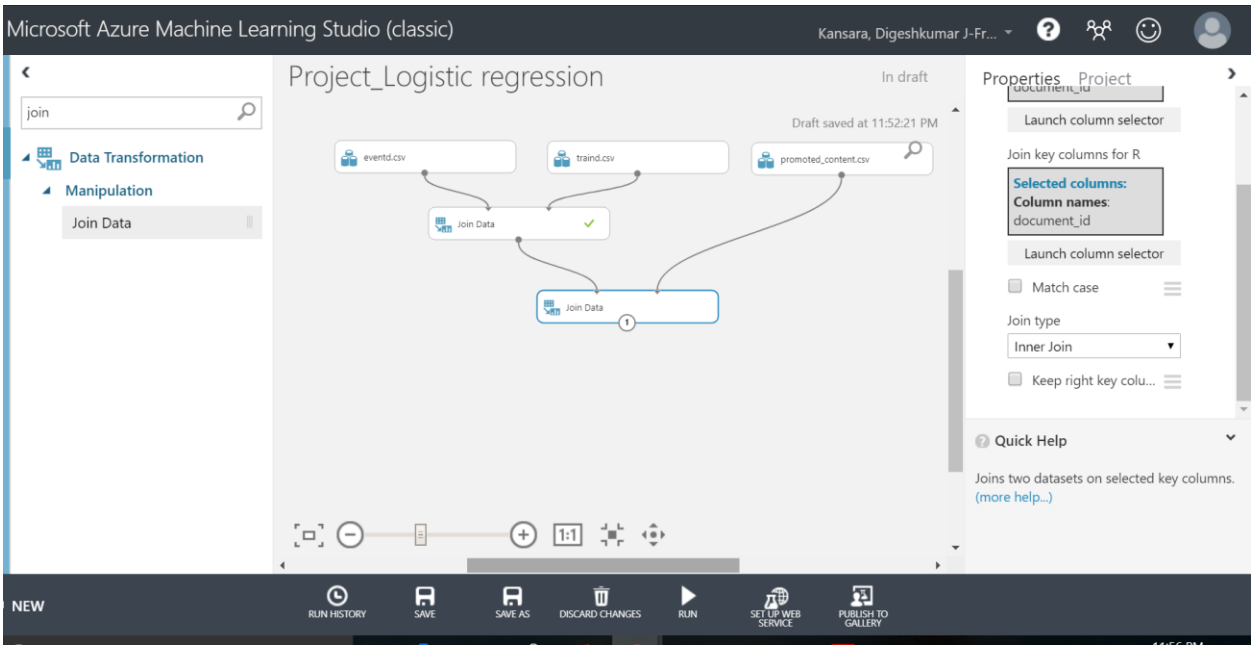
3. Close the dataset.
4. Search for the **Join** module and drag it to the canvas below both the **eventd** dataset and the **traind** dataset. Then connect the output ports from the **eventd** dataset and the **traind** dataset to the Dataset1 and Dataset2 input ports of the Join module respectively.
5. Select the **Join** module, and in the Properties pane, set the following properties:
   a. **Join key columns for L**: Launch the column selector and select **display_id.**
   b. **Join key columns for R**: Launch the column selector and select **display_id.**
   c. **Join type:** Inner Join
   d. **Keep right key column:** Unselected
6. Verify that your experiment resembles the following image, and then save and run (press run from down menu) it.



7. When the experiment has finished running, visualize the Results dataset output port of the Join module, and verify that it. (10 columns)



8. In the experiment items pane on the left, expand **Saved Datasets**, expand **My dataset**, and drag **Promoted_content** to the experiment canvas in the right side of the page and bring join module in after first join module. Connect output of first join module to left input of second join module and output of promoted_content module to right input of second join module. Verify your experiment connection as shown in following image.

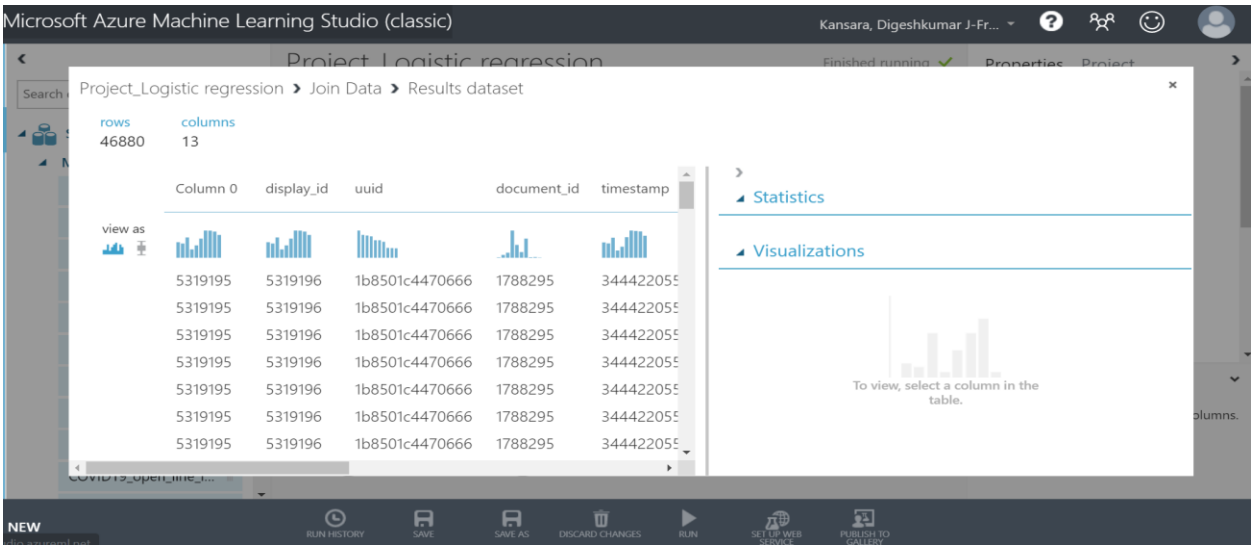


9. Select the Join module, and in the Properties pane, set the following properties:
   a. **Join key columns for L**: Launch the column selector and select **document_id.**
   b. **Join key columns for R**: Launch the column selector and select **document_id.**

c. **Join type:** Inner Join
d. **Keep right key column:** Unselected

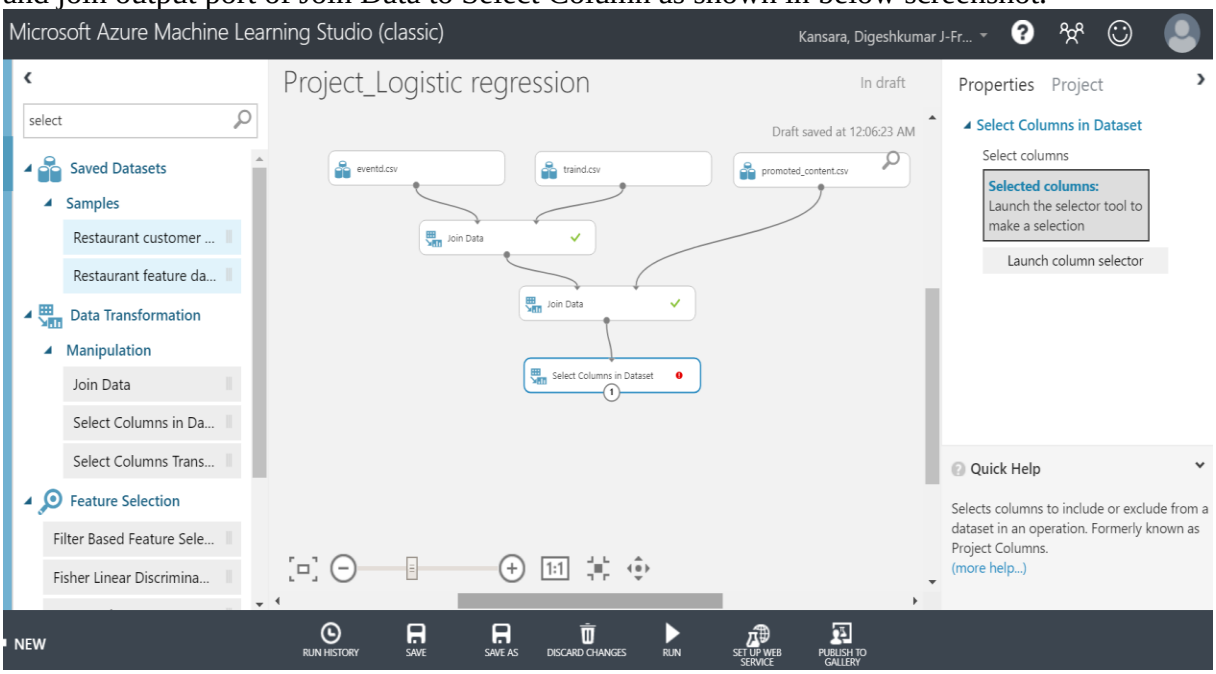
10. Save and Run Experiment.

11. When the experiment has finished running, visualize the result dataset output port of the Join module, and verify that it. (13 columns)

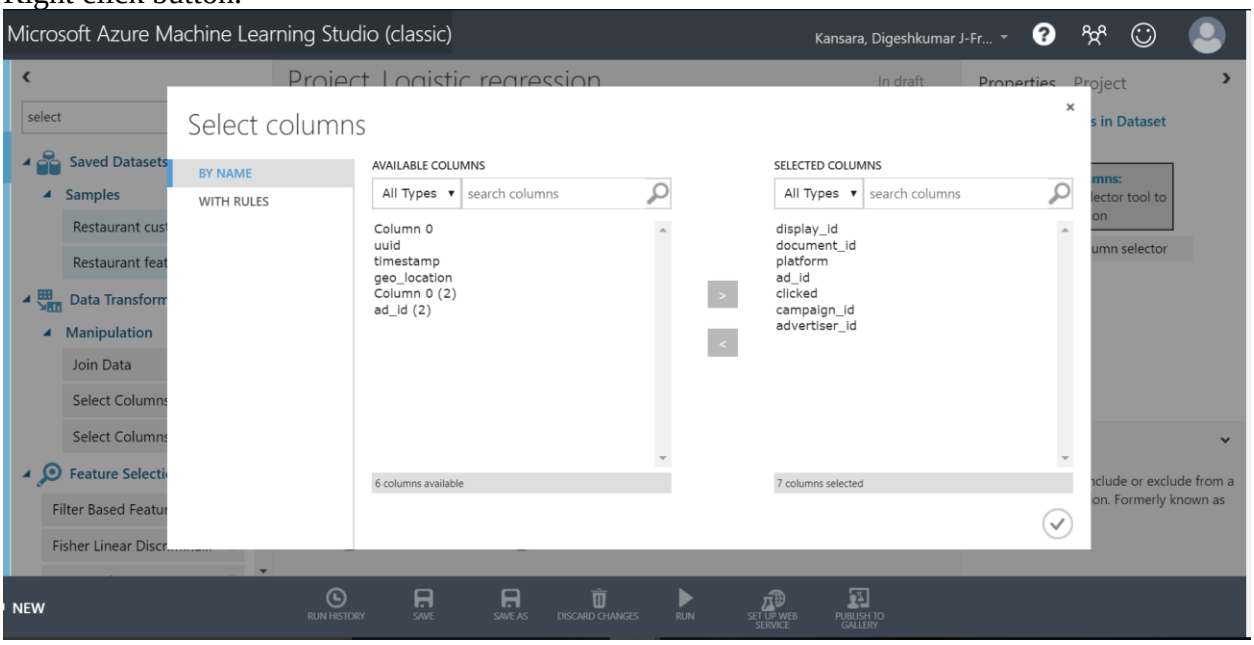


Prepare data for training and testing

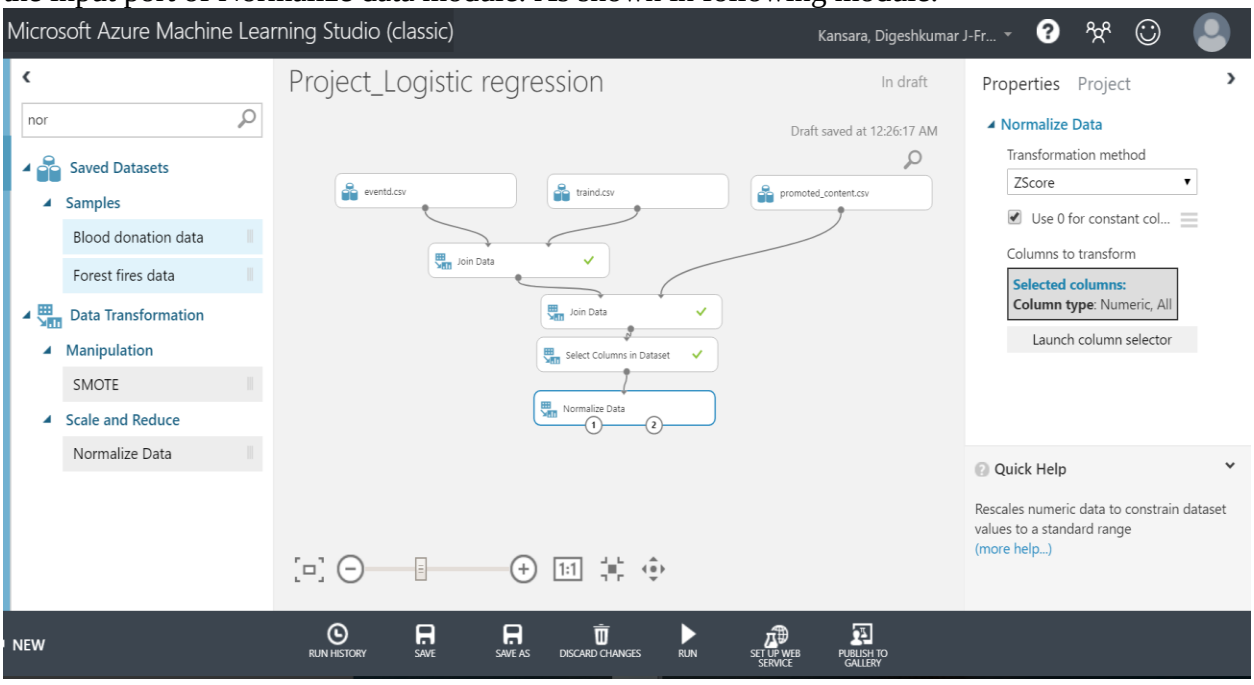
1. Search for **Project Columns (Select Columns in Data Set)** and drag it to the experiment and join output port of Join Data to Select Column as shown in below screenshot.



2. In the **Select columns** dialog box, select option "**BY Name"** and select **display_id, document_id, platform, ad_id, clicked, campgin_id, advertiser_id column** names as shown in the image below. Then click the **OK** icon to close the column selector. Click on Right click button.

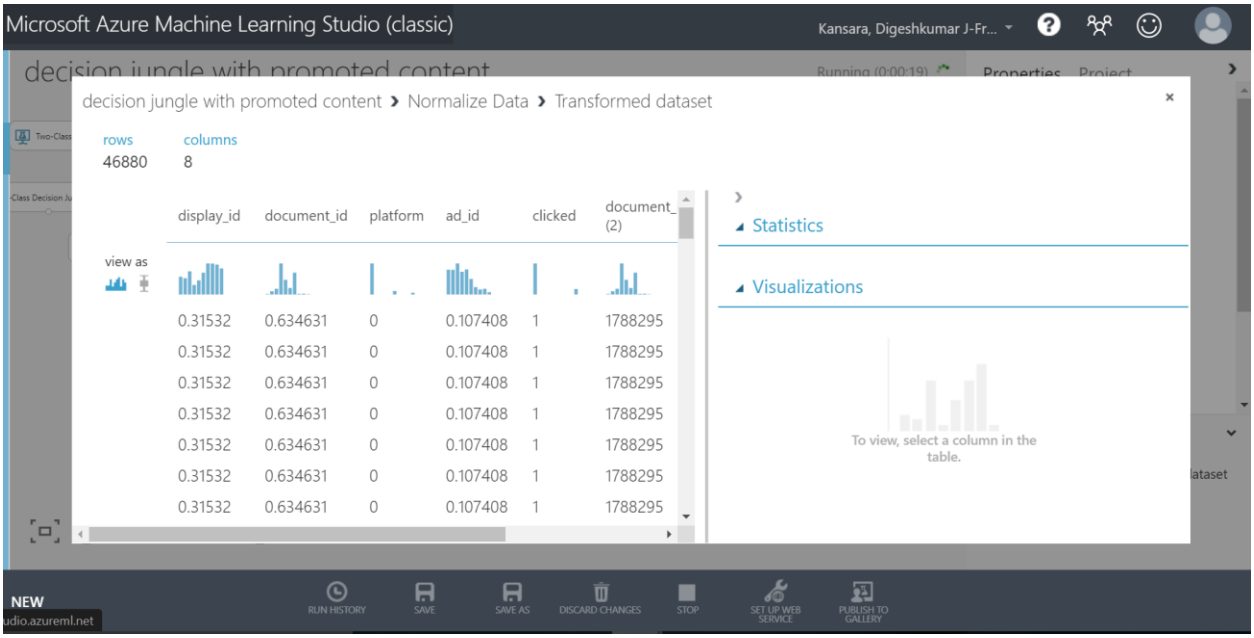


3. Save and Run Experiment.
4. Now we need to Normalize our dataset. From the left experiment pane search for **Normalize data** and drag to experiment. Connect output port of **select column dataset** to the input port of Normalize data module. As shown in following module.

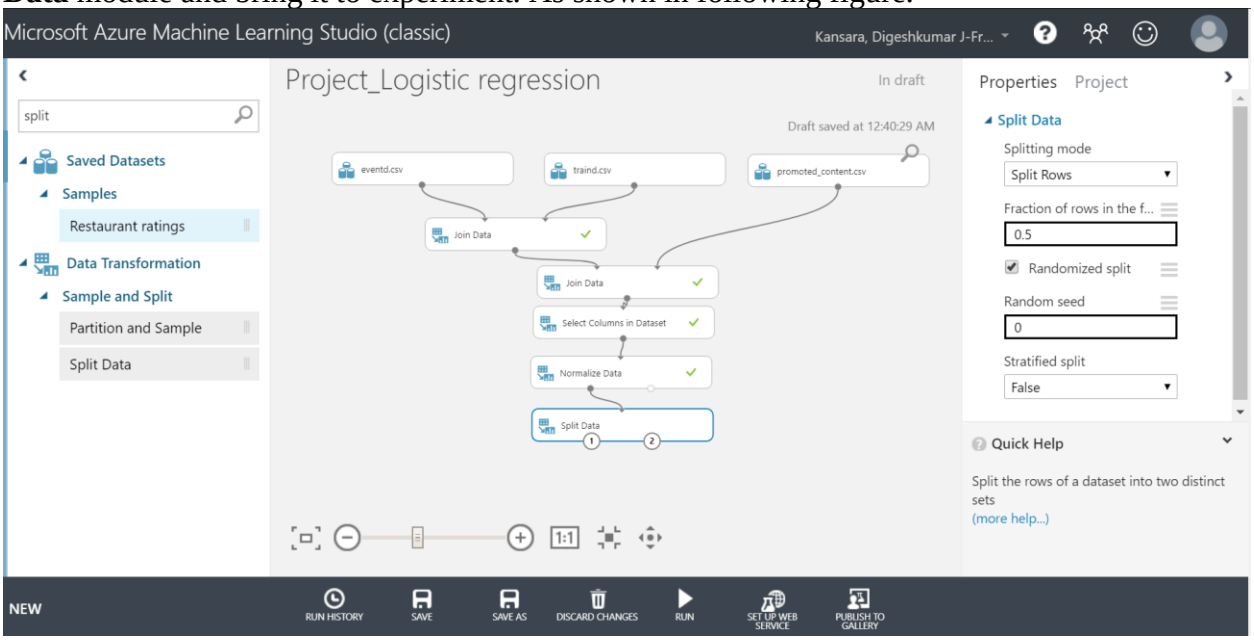


5. Select the **Normalize data** module, and in the Properties pane, set the following properties:
   a. **Transformation method**: Minmax**.**
   b. **Launch column selector**: select **By Name** and select **all columns**.

6. Save and Run the experiment. Right click on Normalize and select visualize to analyze the data.



7. Now we need to split data for training and testing. In left side experiment pane select **Split Data** module and bring it to experiment. As shown in following figure.



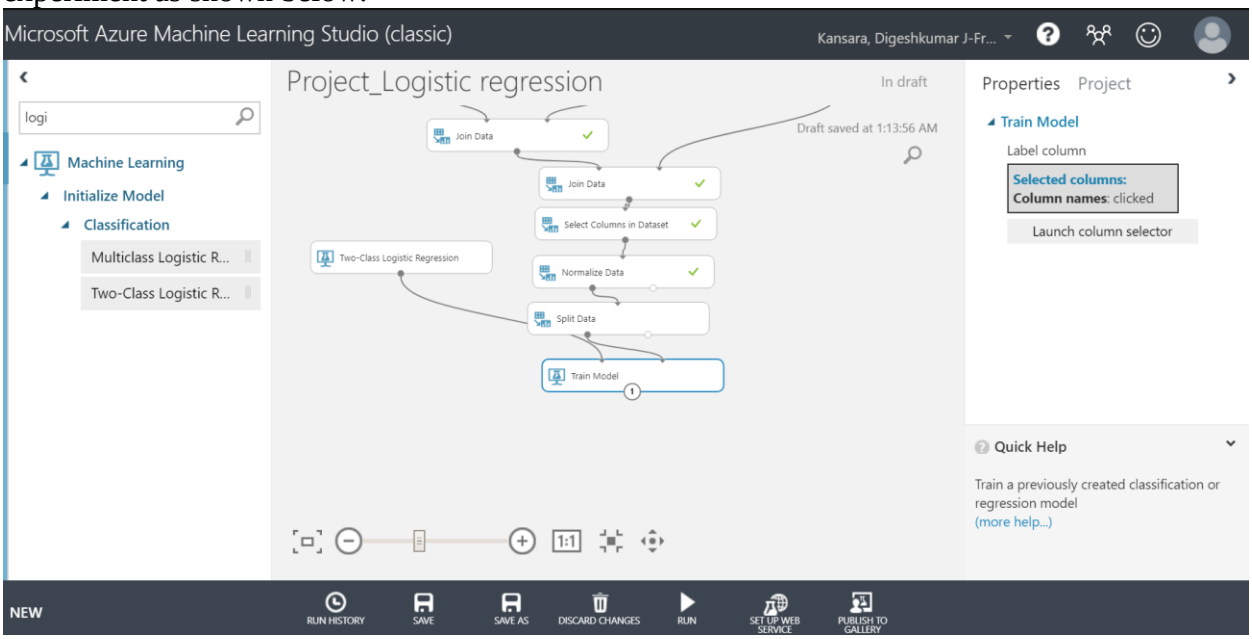
8. Select the **split module**, and in the Properties pane, set the following properties:
   a. **Splitting mode**: splitting Rows**.**
   b. **Fraction:** 0.7
   c. **Randomized Split:** Selected
   d. **Random Split:** 1234
9. Save and Run Experiment.

Train data and measure accuracy and AUC of different Algorithm

Algorithm 1: Two class Logistic Regression

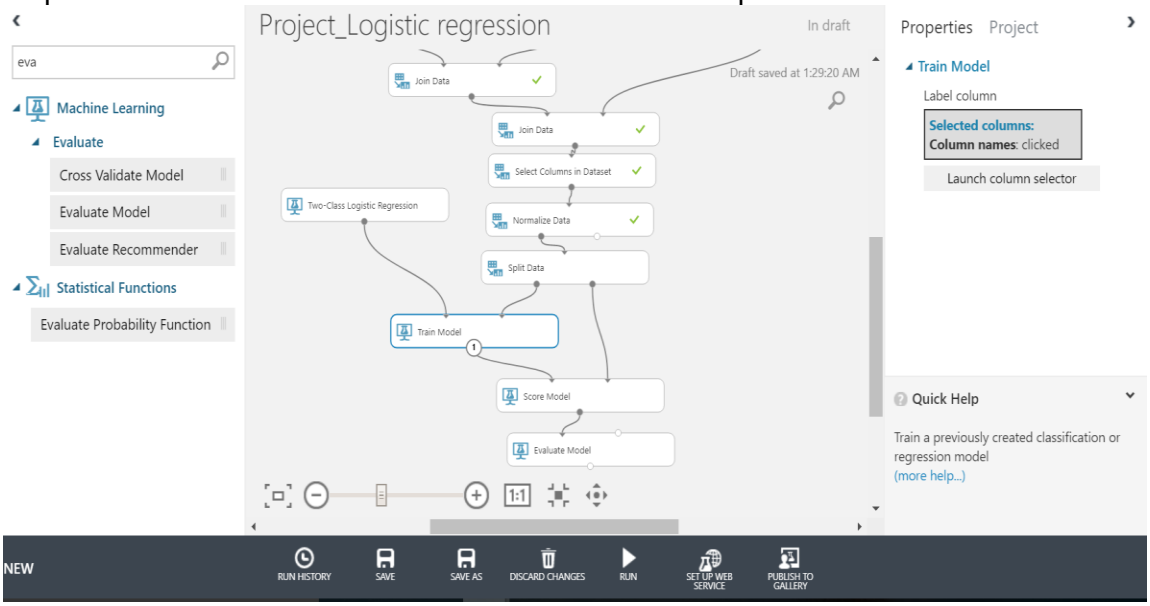
In previous step, we have created our training and test dataset. Next step is to measure accuracy of different Algorithm.

1. From left side experiment pane search for Two-class Logistic Regression module in the experiment and put it left side.
2. Bring **Train data** module in the experiment. Select launch column selector and select **clicked** column which we are going to predict.
3. Now bring Output of Two class Logistic Regression to left side of Train Data module and left output of split data module to right side of Train data module input. Now verify experiment as shown below.



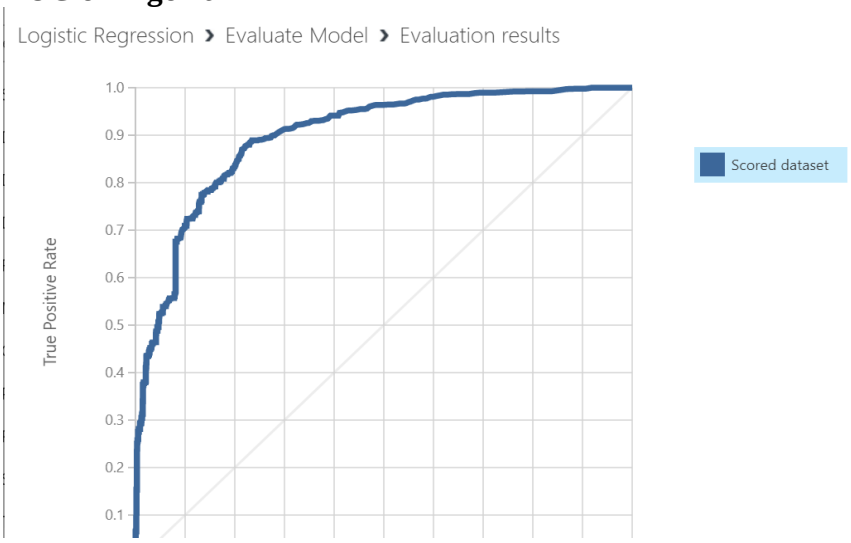
4. Save and Run the Experiment.
5. Now we need to Score and Evaluate our model.
6. From left side experiment pane drag Score module to the experiment. Connect output of train module to left input of score module and right output of split module to right input of score module.

7. From Left side experiment pane drag Evaluate Model module in experiment and connect output module of score module to Evaluate model. The experiment should look like below.



8. Save and Run the experiment.
9. Now right click on Evaluate model and select visualize and check output as follows.

AUC of Algorithm



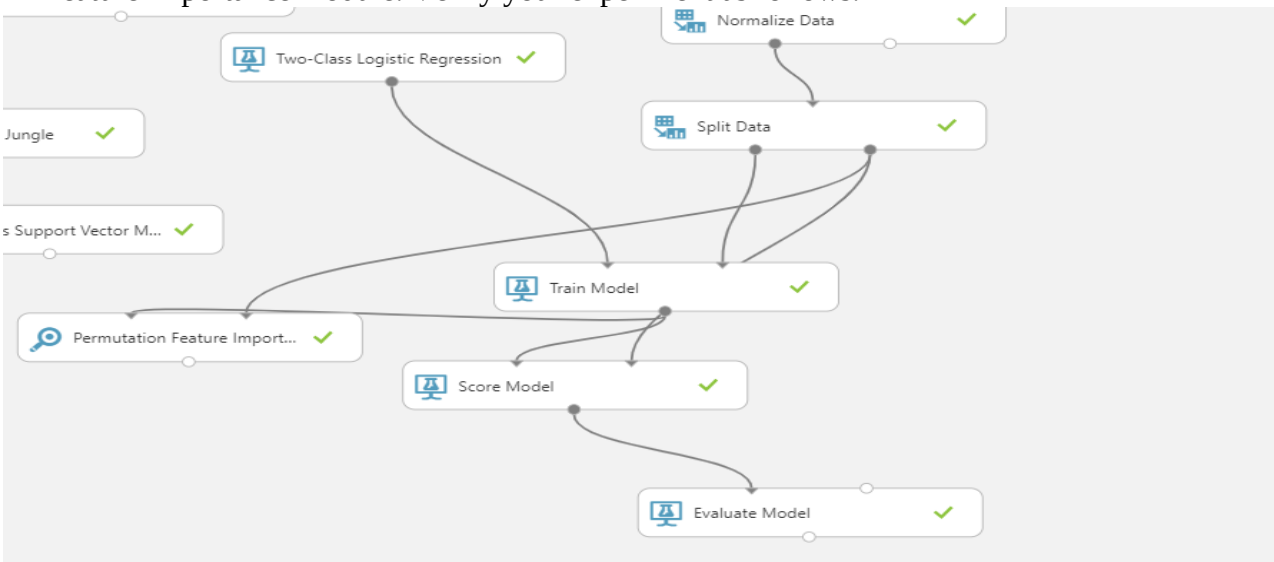
Accuracy of algorithm.

True Positive	False Negative	Accuracy	Precision	Threshold	AUC
866	1251	0.893	0.775	0.5	0.898

False Positive	True Negative	Recall	F1 Score
252	11695	0.409	0.535

Positive Label	Negative Label
1	0

10. **Now Bring Permutation Feature Importance** module to left side of experiment after training data. From Train Data Model output to right input of permutation feature importance module and connect right side of split data output to left input of permutation feature importance module. Verify your experiment as follows.



11. Save and Run Experiment. Now right click on permutation feature important module and click visualize. You can see result as following.

rows	columns
8	2

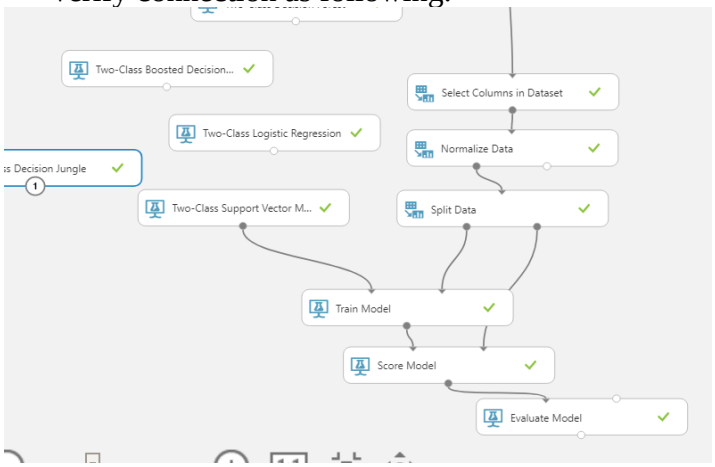
view as

Feature	Score
geo_location	0.096701
timestamp	0.005759
platform	0.003057
display_id	0.00128
ad_id	0.000427
document_id	-0.000071
campaign_id	-0.000711
advertiser_id	-0.000924

Algorithm 2: Support Vector Machine (SVM)

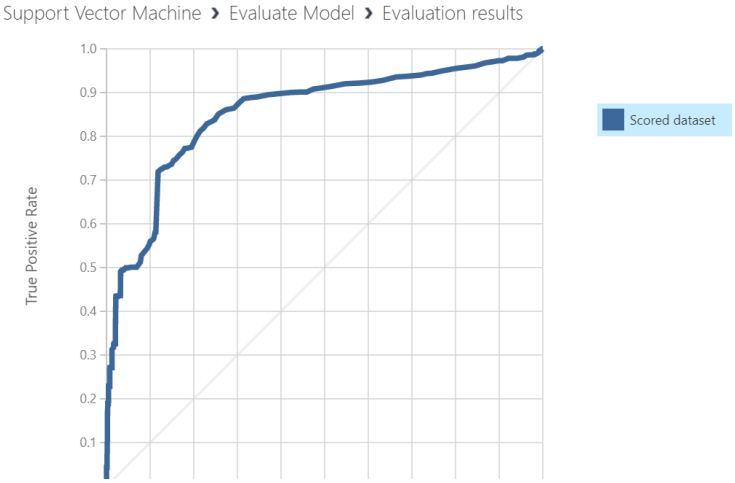
1. We have completed Ad prediction Using Logistic Regression. Now we are going to use support vector machine algorithm.
2. Open Logistic_Algorithm that we have created in previous step from left side experiment side pane find two class support vector machine and bring it to experiment module.

3. Now disconnect between Two class logistic regression with train data.
4. Connect output of Two class Support vector machine to left side of train data. You can verify connection as following.



5. Save and run Experiment. Visualize Accuracy and AUC as following.

AUC:



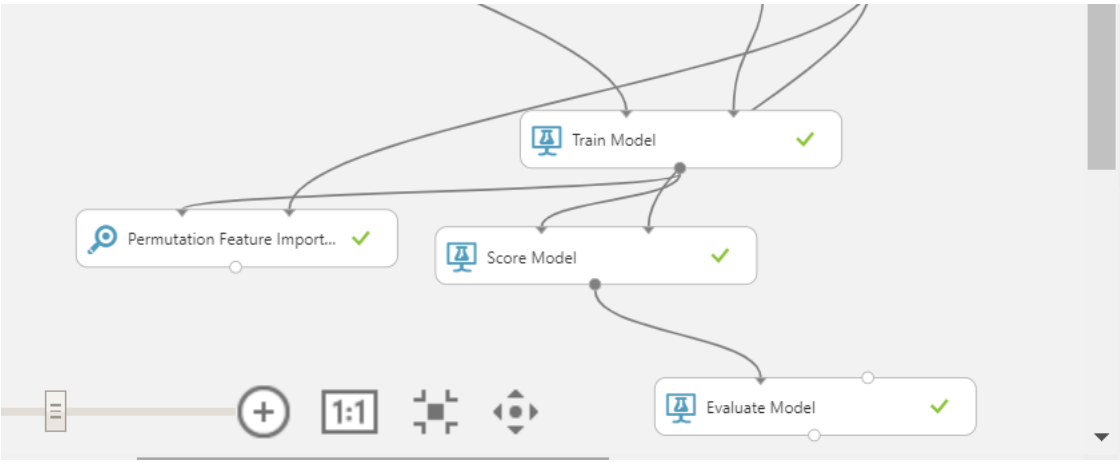
Accuracy of Algorithm:

True Positive	False Negative	Accuracy	Precision	Threshold	AUC
704	1413	0.882	0.742	0.5	0.855

False Positive	True Negative	Recall	F1 Score
245	11702	0.333	0.459

Positive Label	Negative Label
1	0

6. **Now Bring Permutation Feature Importance** module to left side of experiment after training data. From Train Data Model output to right input of permutation feature importance module and connect right side of split data output to left input of permutation feature importance module. Verify your experiment as follows.



7. Save and Run Experiment. Now right click on permutation feature important module and click visualize. You can see result as following.

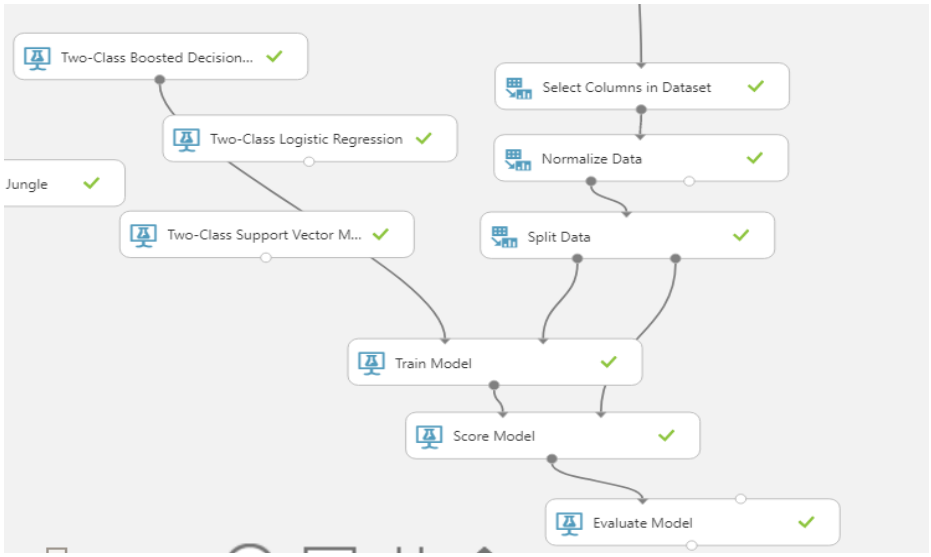
rows 8 columns 2

view as

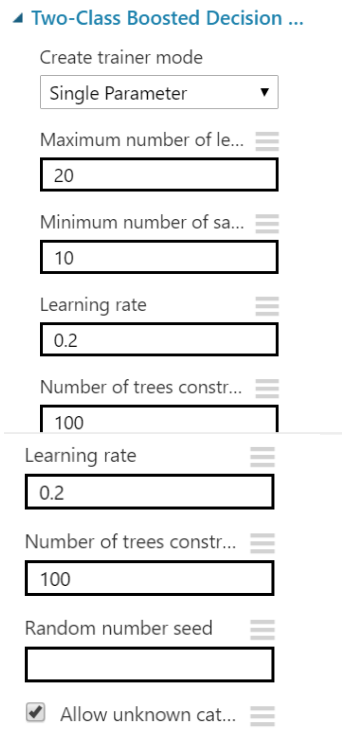
Feature	Score
geo_location	0.078214
display_id	0
document_id	0
timestamp	0
platform	0
ad_id	0
campaign_id	0
advertiser_id	0

Algorithm 3: Two class Boosted Decision Tree

1. We have completed Ad prediction using Support Vector Machine. Now We are going to use Two class Boosted Decision Tree algorithm.
2. Open Logistic_Algorithm that we have created in previous step from left side experiment side pane find two class Boosted Decision Tree and bring it to experiment module.
3. Now disconnect between Two class Boosted Decision Tree with train data.
4. Connect output of Two class Boosted Decision Tree to left side of train data. You can verify connection as following.

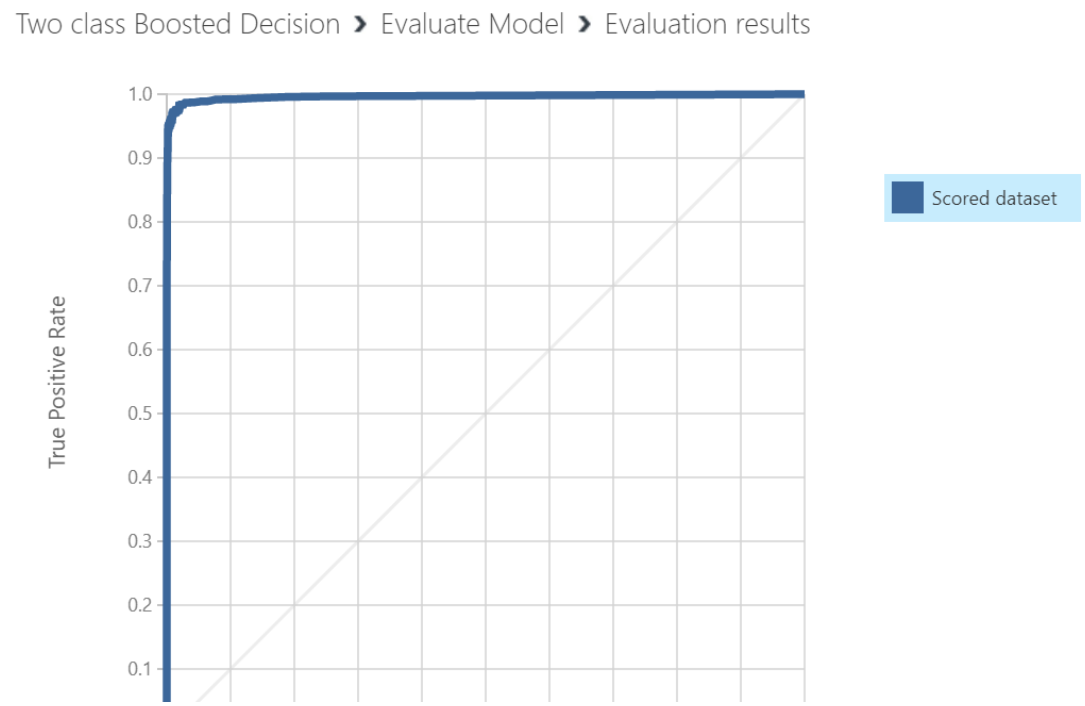


5. Check Property of Two class Boosted Decision Tree from right properties pane as shown in follow.



6. Save and run the experiment. Visualize Accuracy and AUC as following.

AUC:



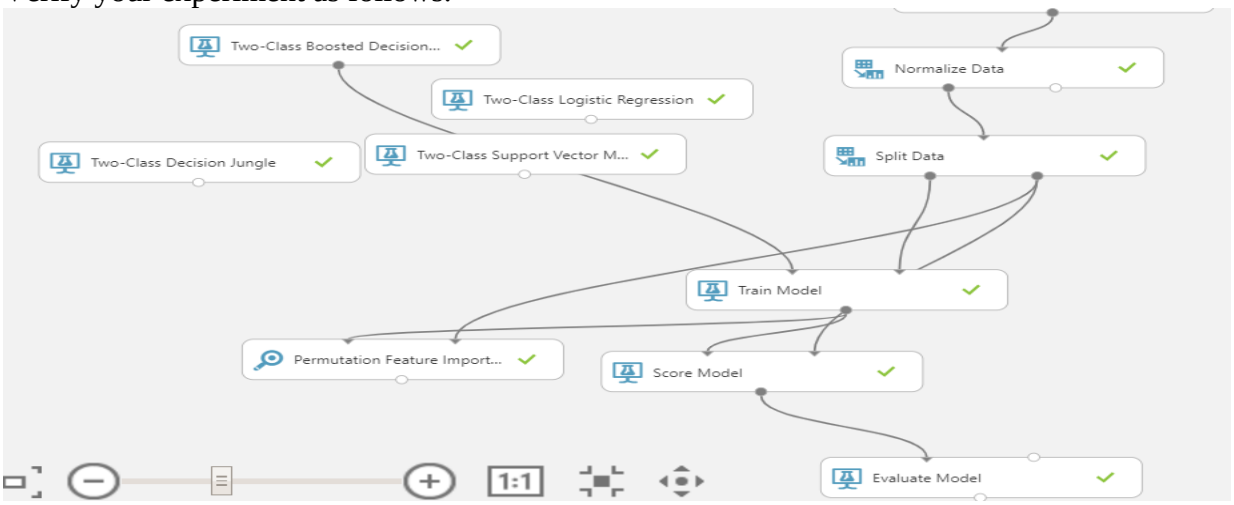
Accuracy:

True Positive	False Negative	Accuracy	Precision	Threshold	AUC
2020	97	0.989	0.969	0.5	0.996

False Positive	True Negative	Recall	F1 Score
64	11883	0.954	0.962

Positive Label	Negative Label
1	0

7. Now Bring Permutation Feature Importance module to left side of experiment after training data. From Train Data Model output to right input of permutation feature importance module and connect right side of split data output to left input of permutation feature importance module. Verify your experiment as follows.



8. Save and Run Experiment. Now right click on permutation feature important module and click visualize. You can see result as following.

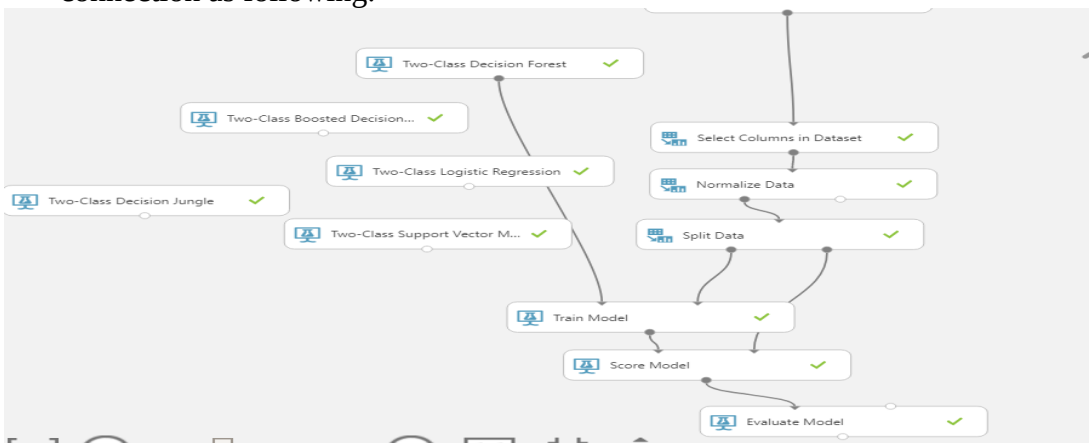
rows	columns
7	2

view as

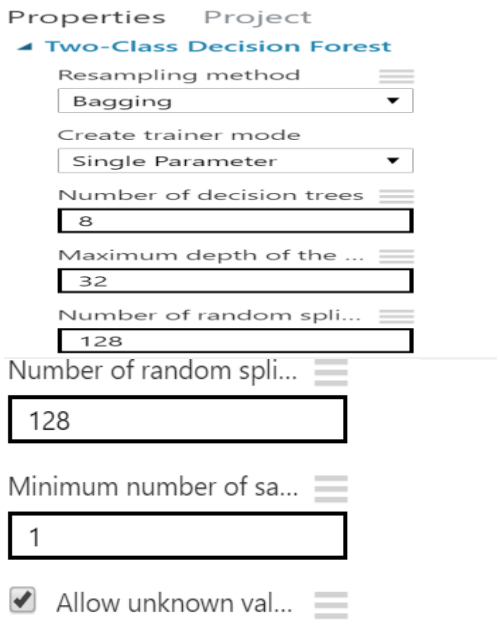
Feature	Score
ad_id	0.157992
display_id	0.096061
document_id	0.046786
advertiser_id	0.014078
campaign_id	0.00583
platform	0.002702
timestamp	0

Algorithm 4: Two Class Decision Forest

1. Now we are going to use Two class Boosted Decision Tree algorithm.
2. Open Logistic_Algorithm that we have created in previous step from left side experiment side pane find two class Decision Forest and bring it to experiment module.
3. Now disconnect between two class Decision Forest with train data.
4. Connect output of two class Decision Forest to left side of train data. You can verify connection as following.



5. Check Property of Two class Decision Forest from right Properties pane as shown in follow.



6. Save and run Experiment. Visualize Accuracy and AUC as following.

AUC:

Accuracy:

7.

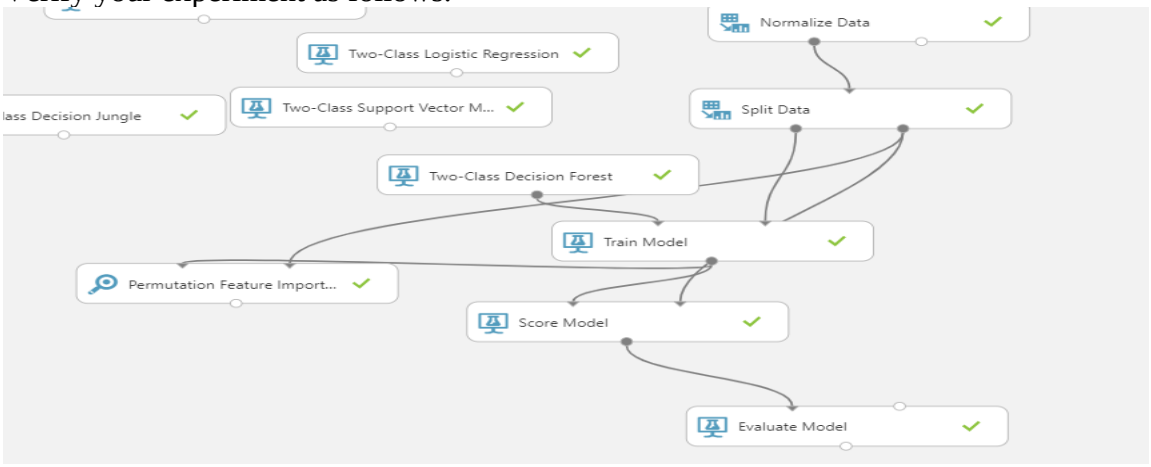
True Positive	False Negative	Accuracy	Precision	Threshold	AUC
1753	364	0.974	0.996	0.5	0.995

False Positive	True Negative	Recall	F1 Score
7	11940	0.828	0.904

Positive Label	Negative Label
1	0

---

7. Now Bring Permutation Feature Importance module to left side of experiment after training data. From Train Data Model output to right input of permutation feature importance module and connect right side of split data output to left input of permutation feature importance module. Verify your experiment as follows.



8. Save and Run Experiment. Now right click on permutation feature important module and click visualize. You can see result as following.

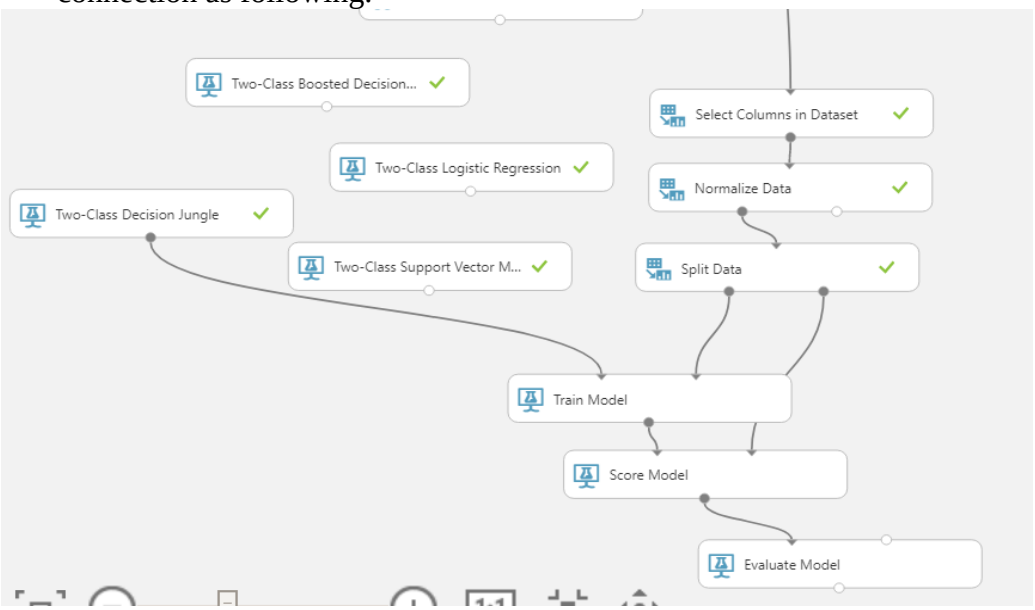
rows 7 columns 2

view as

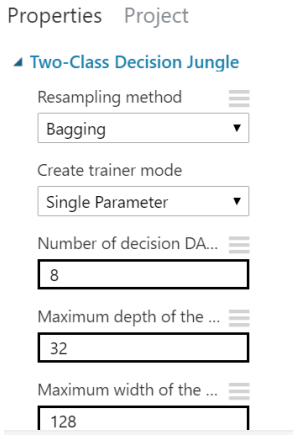
Feature	Score
ad_id	0.140785
timestamp	0.084044
display_id	0.054892
document_id	0.047426
advertiser_id	0.045577
platform	0.01351
campaign_id	0.002062

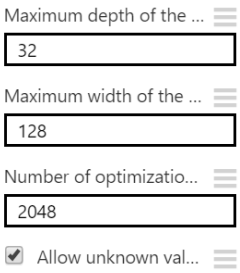
Algorithm 5: Two Class Decision Jungle

1. We have completed Ad prediction using Two Class Decision Forest. Now we are going to use Two Class Decision Jungle algorithm.
2. Open Logistic_Algorithm that we have created in previous step from left side experiment side pane find two class Decision Jungle and bring it to experiment module.
3. Now disconnect between two class Decision Jungle with train data.
4. Connect output of two class Decision Jungle to left side of train data. You can verify connection as following.



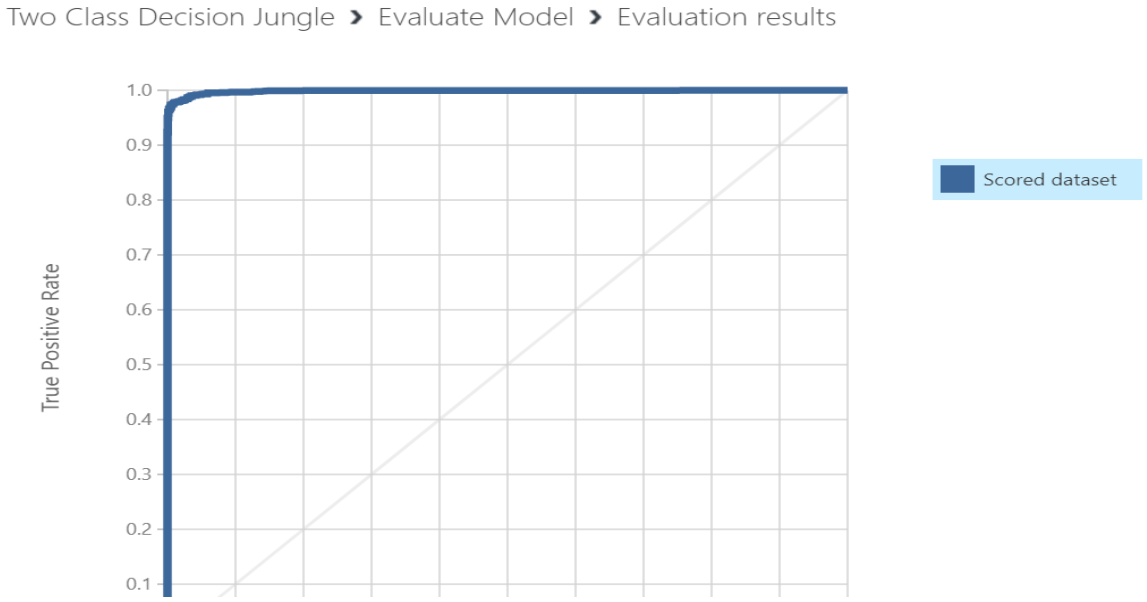
5. Check Property of Two class Decision Jungle from right Properties pane as shown in follow.



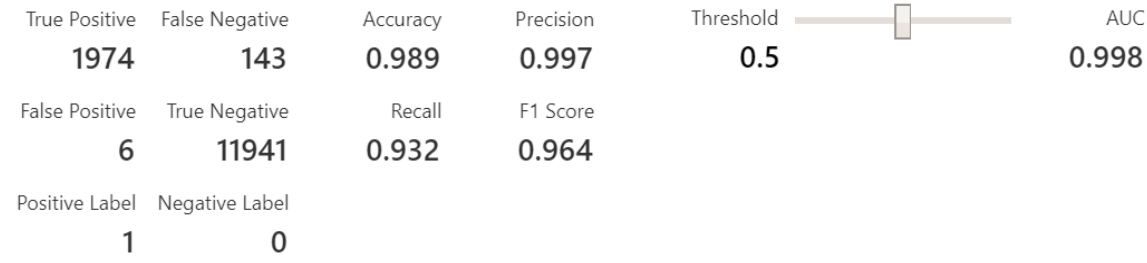


6. Save and run the experiment. Visualize Accuracy and AUC as following.

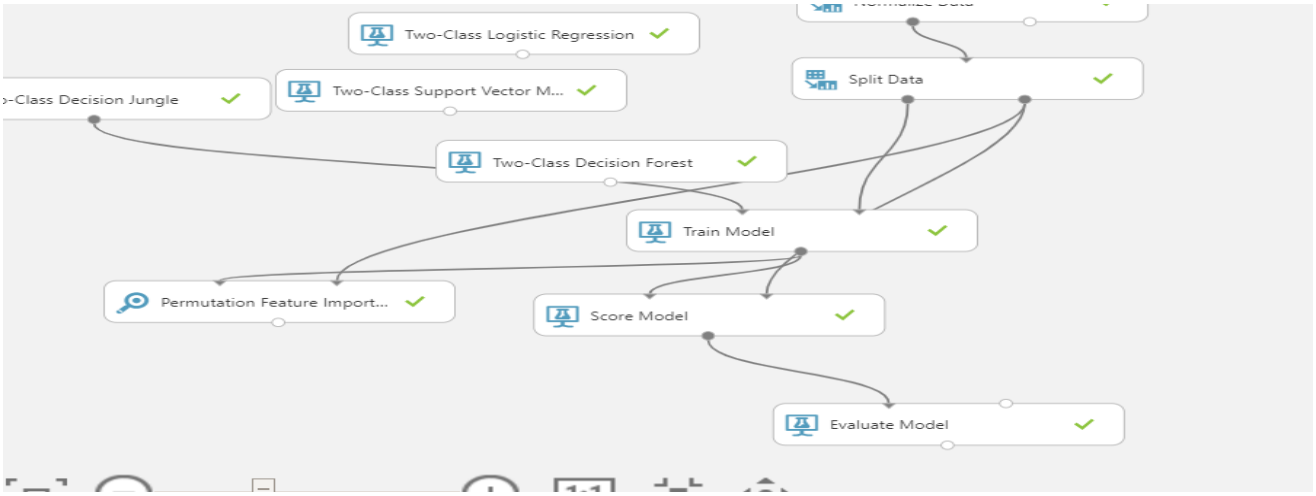
AUC:



Accuracy:



7. **Now Bring Permutation Feature Importance** module to left side of experiment after training data. From Train Data Model output to right input of permutation feature importance module and connect right side of split data output to left input of permutation feature importance module. Verify your experiment as follows.



8. Save and Run Experiment. Now right click on permutation feature important module and click visualize. You can see result as following.

Two Class Decision Jungle › Permutation Feat

rows 7 columns 2

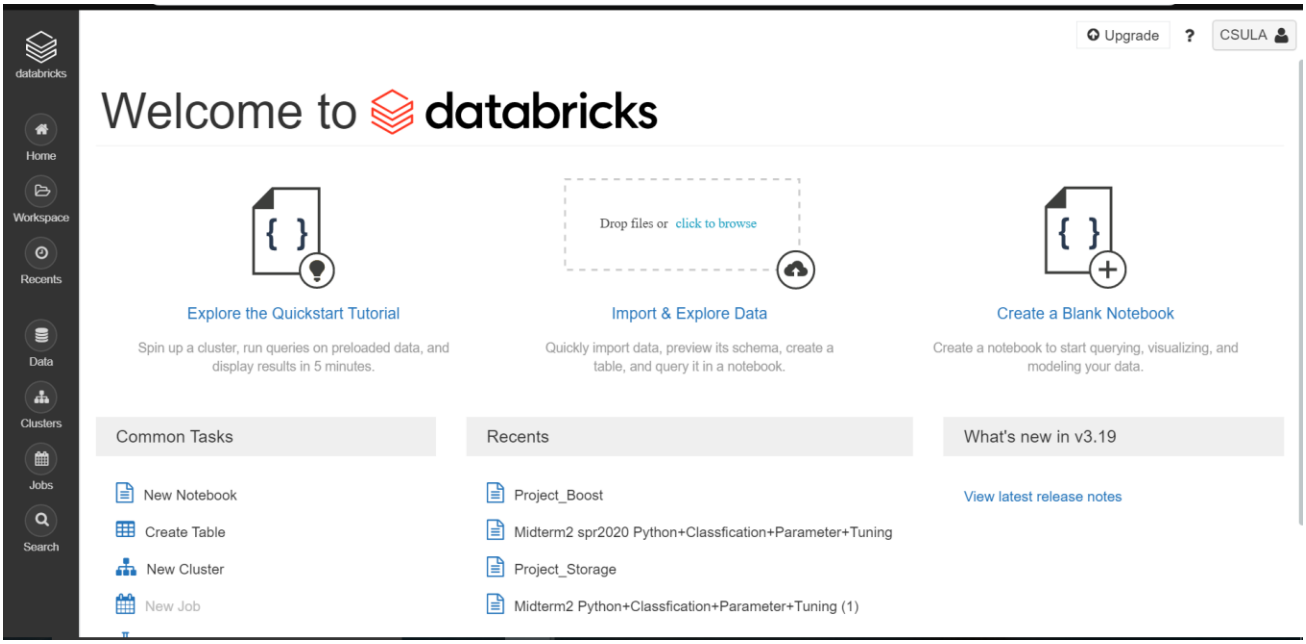
view as

Feature	Score
ad_id	0.130902
display_id	0.066695
document_id	0.058092
timestamp	0.055958
advertiser_id	0.044937
platform	0.023251
campaign_id	0.008604

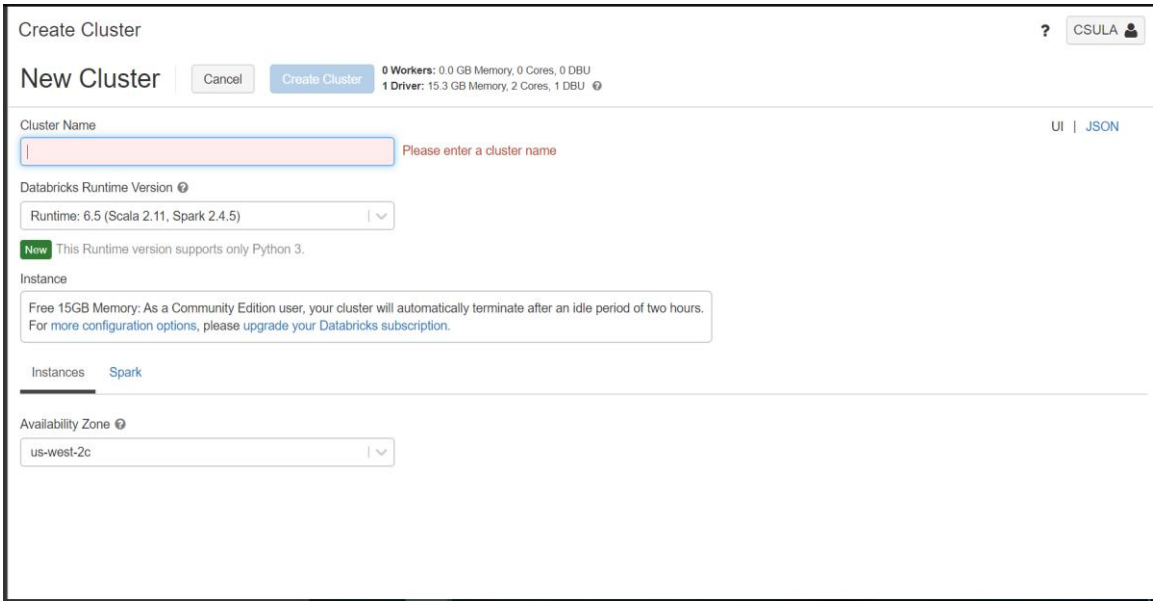
Step 4: Ad prediction using spark in Databricks

1. To Work in Databricks we need to navigate to Databricks Community Edition to login. If you **don't have** databricks community edition account, you need to first Sign Up for Community Edition.
   https://community.cloud.databricks.com/login.html?o=1035240001367446#notebook/2698979388235870/command/2698979388235882

2. When we login to databricks. We need to create a cluster. To create a cluster, we need to click on clusters button from left side pane as shown in following image.



3. Now it will navigate to **create cluster page** as shown in below. click on create cluster button.

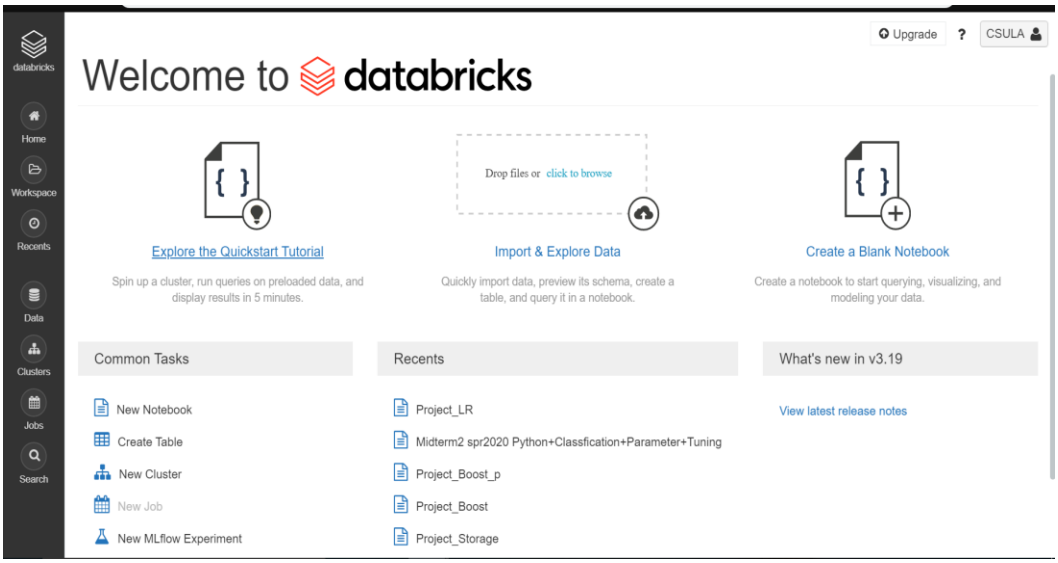


Cluster Name: Choose combination of any letter.
Databricks Runtime Version: Choose compatible to your project and system and click on create cluster button from top as it become visible after we enter cluster name and cluster version.

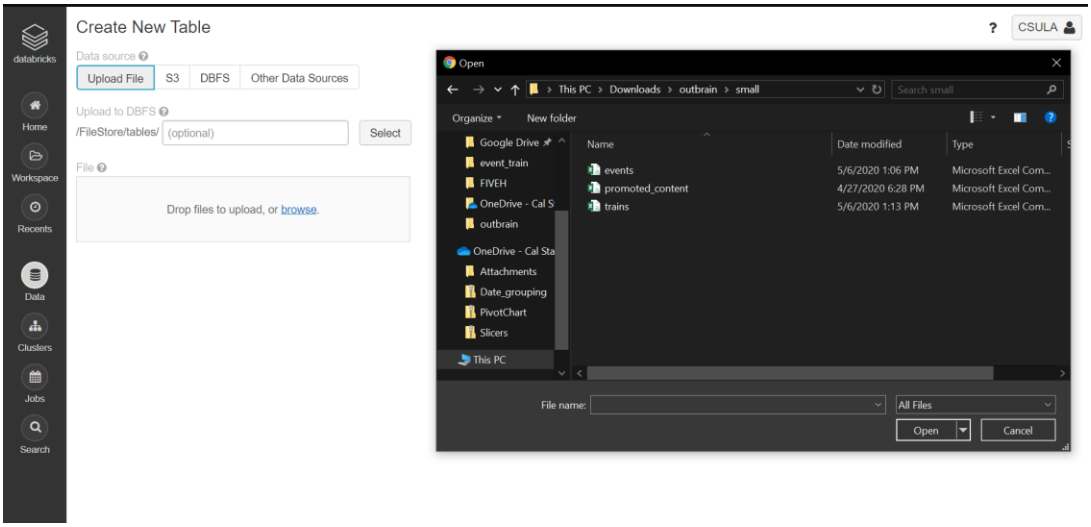
It will take 15-20 minute to create a cluster.

4. After launching a cluster, select **"Create Table"** at the **Databricks** UI/GUI in order to upload the file **trains.csv, events.csv** and **Promoted_content.csv.**

Note: I changed my file name trains.csv for some reasons. You can upload csv files with name which you gave during sampling in jupyter notebook in stage 2.

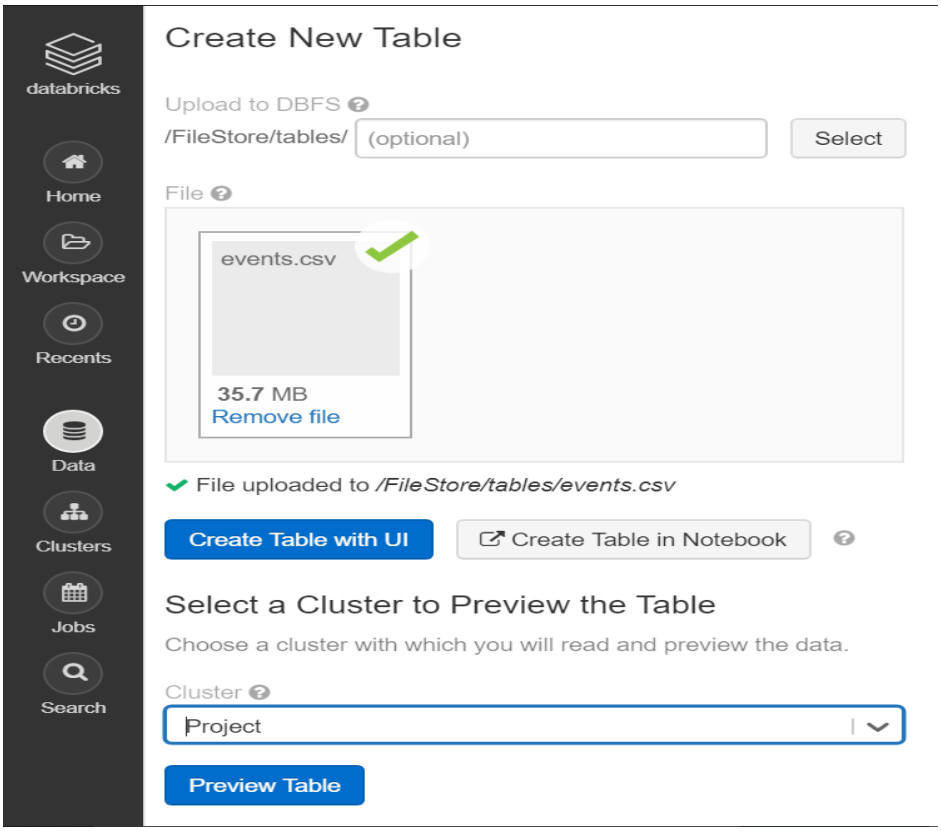


5. It will go to the following page. You need to select "browse" and locate the file to upload at the file explorer:

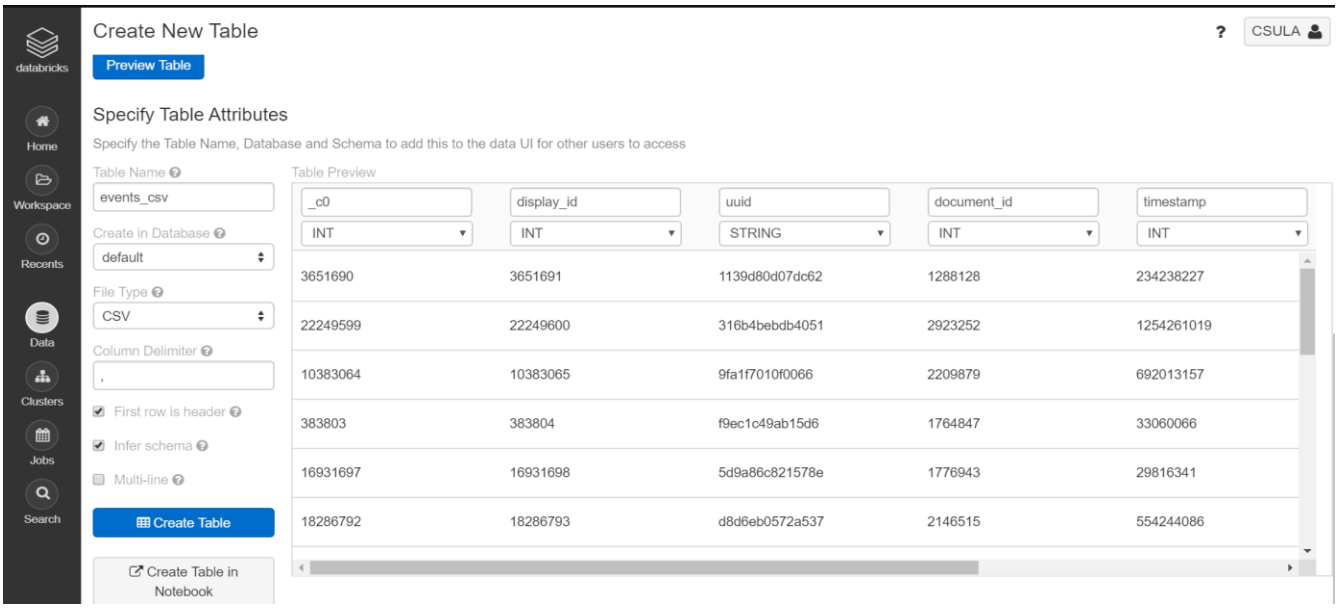


Note: File Size is 35MB it will take about 15 minutes.

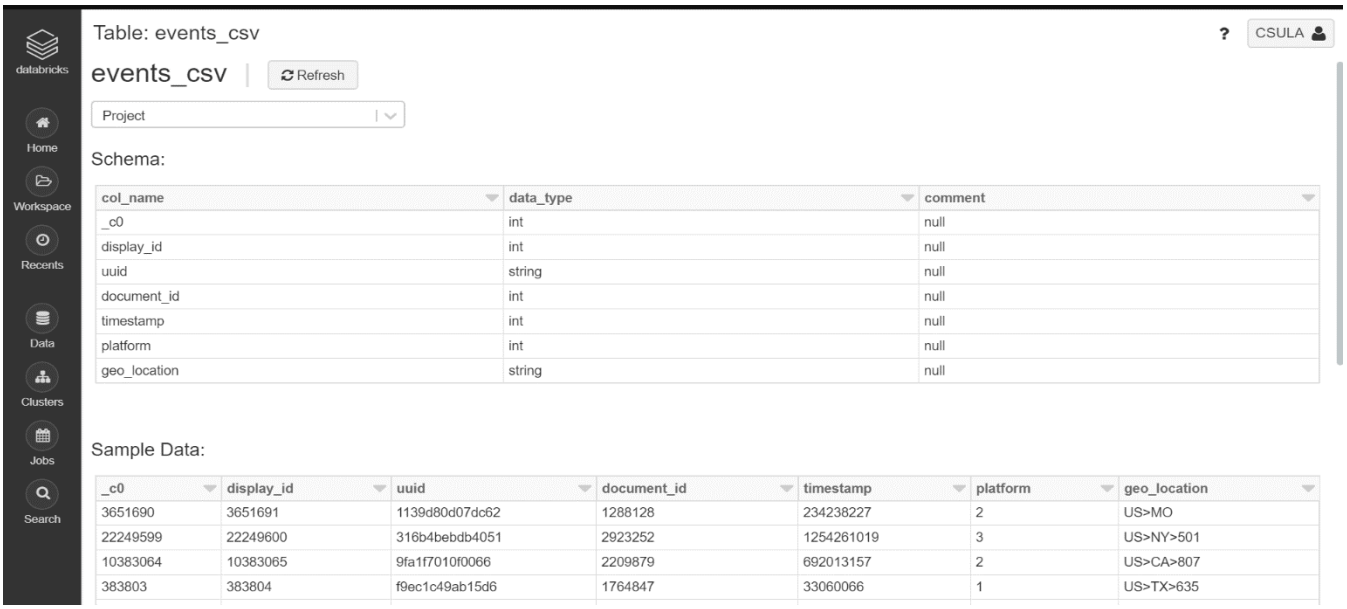
6. Create a table from the uploaded csv file.
   - Select "Create Table with UI" then select the cluster from the drop-down box, which you created above.
   - Then, click Preview Table button



7. It will display the portion of table columns, data types, and the values. You have to set up the data type and mark the head option as you have seen at ipynb at for example:
    - Table Name: table2015_csv
    - Mark “First row is header”
    - Mark “Infer Schema”
    - Then, select **Create Table** button

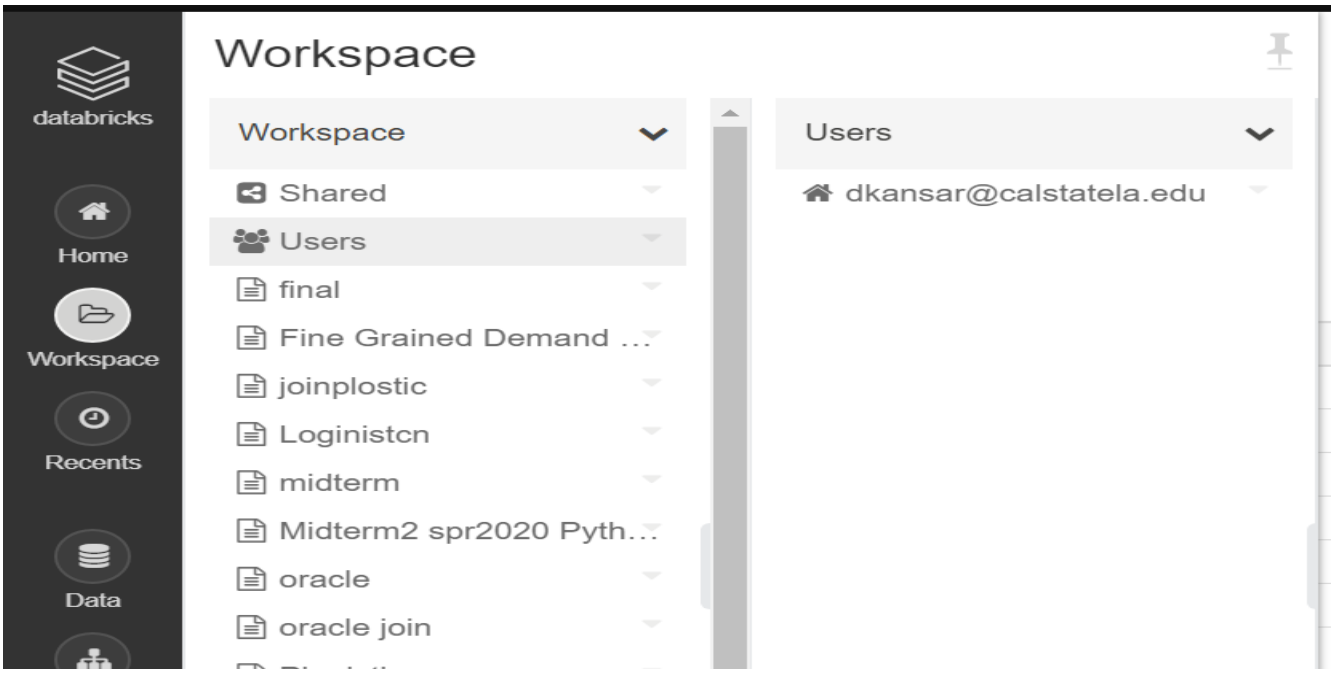


8. It will show the following table **"events_csv"** created.



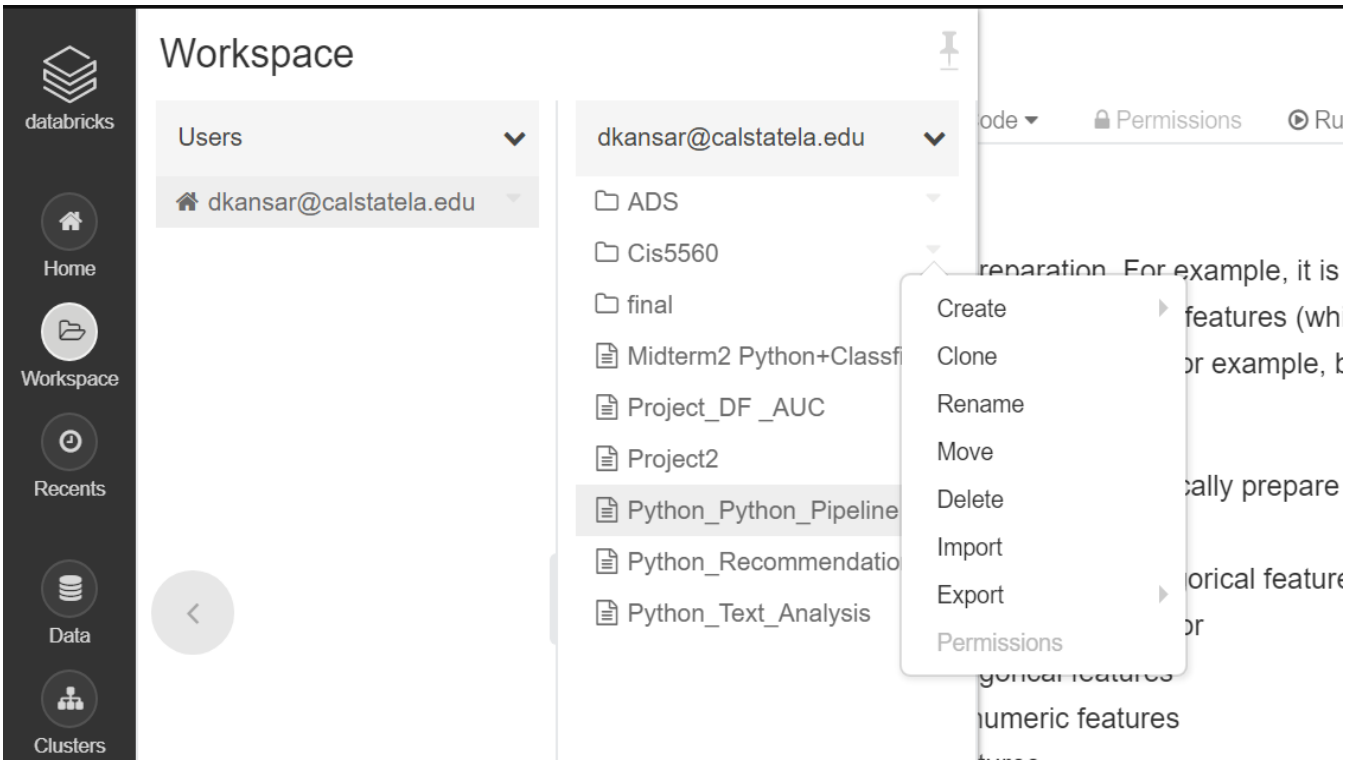
Now follow step from 4 to 8 and create table for trains, Promoted_content.

9. You have to create a folder named cis5560 of Data Bricks account, which should be at the left frame of the page at: Workspace > Users > Your Account Email > cis5560.

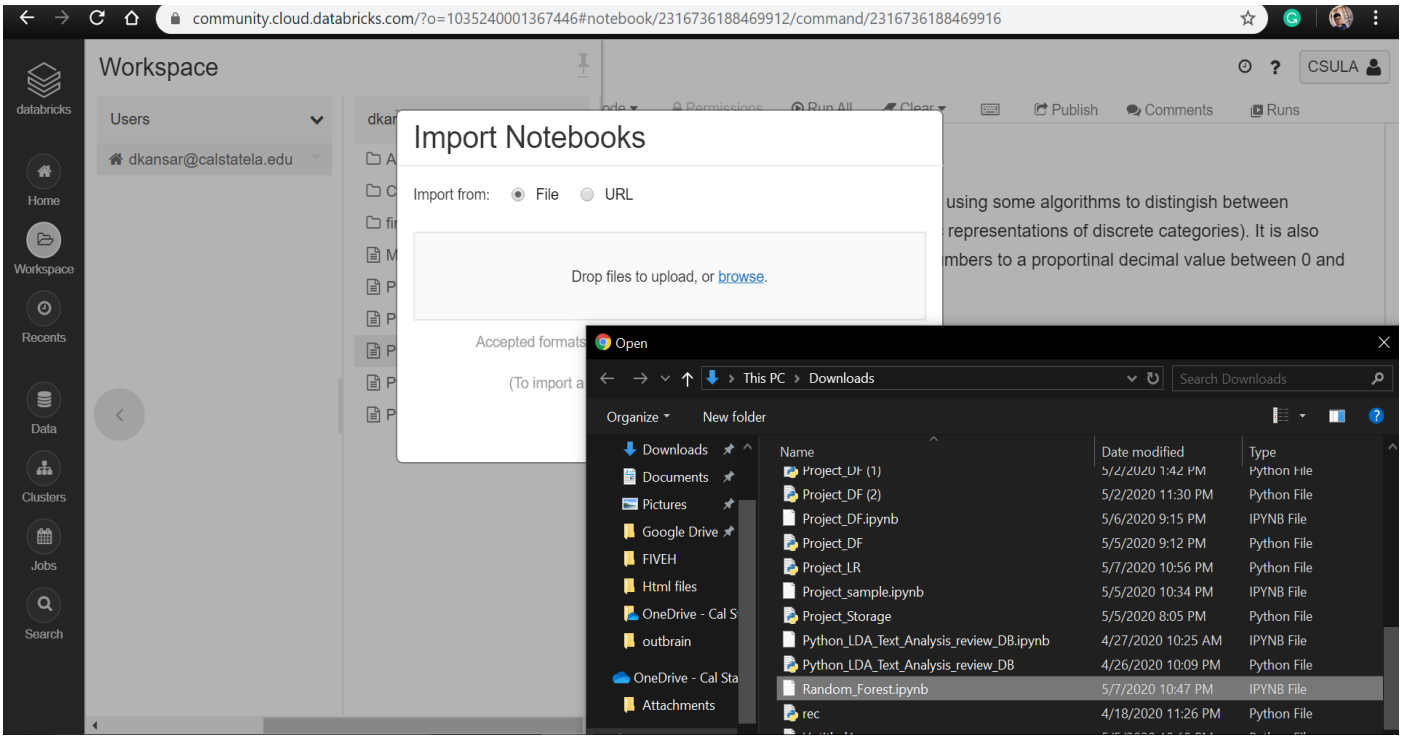


Random Forest Algorithm:

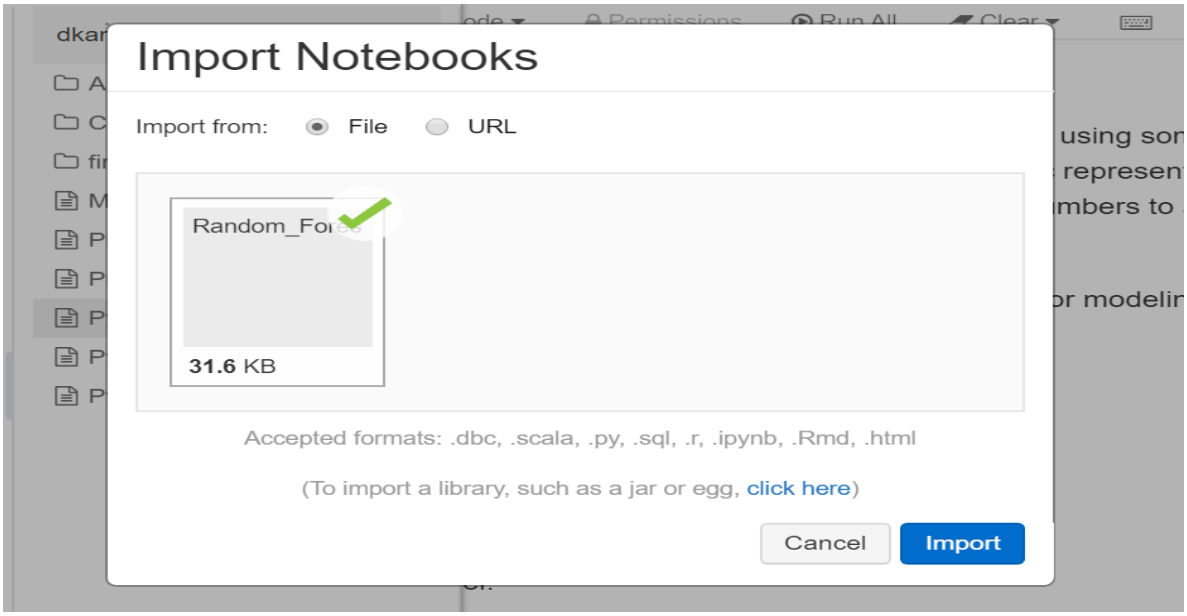
10. Import "Random_forest. ipynb" to the folder **cis5560**: **cis5560 > import**



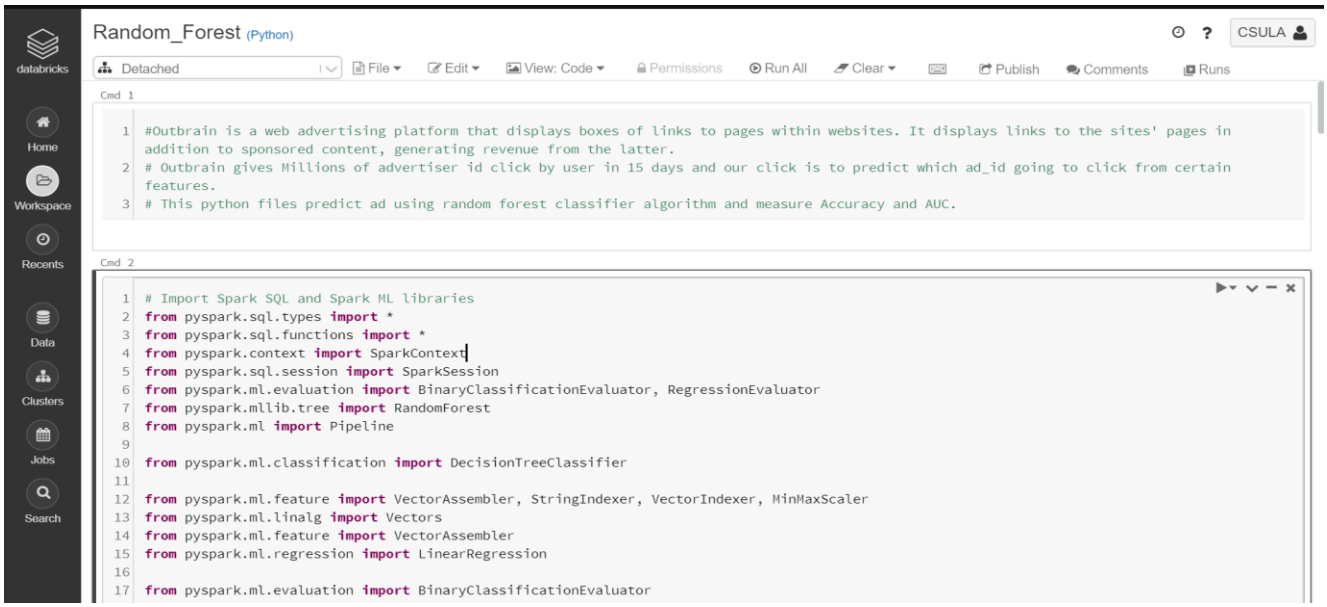
11. Now click on import and browse “Random_forest.ipynb” file.



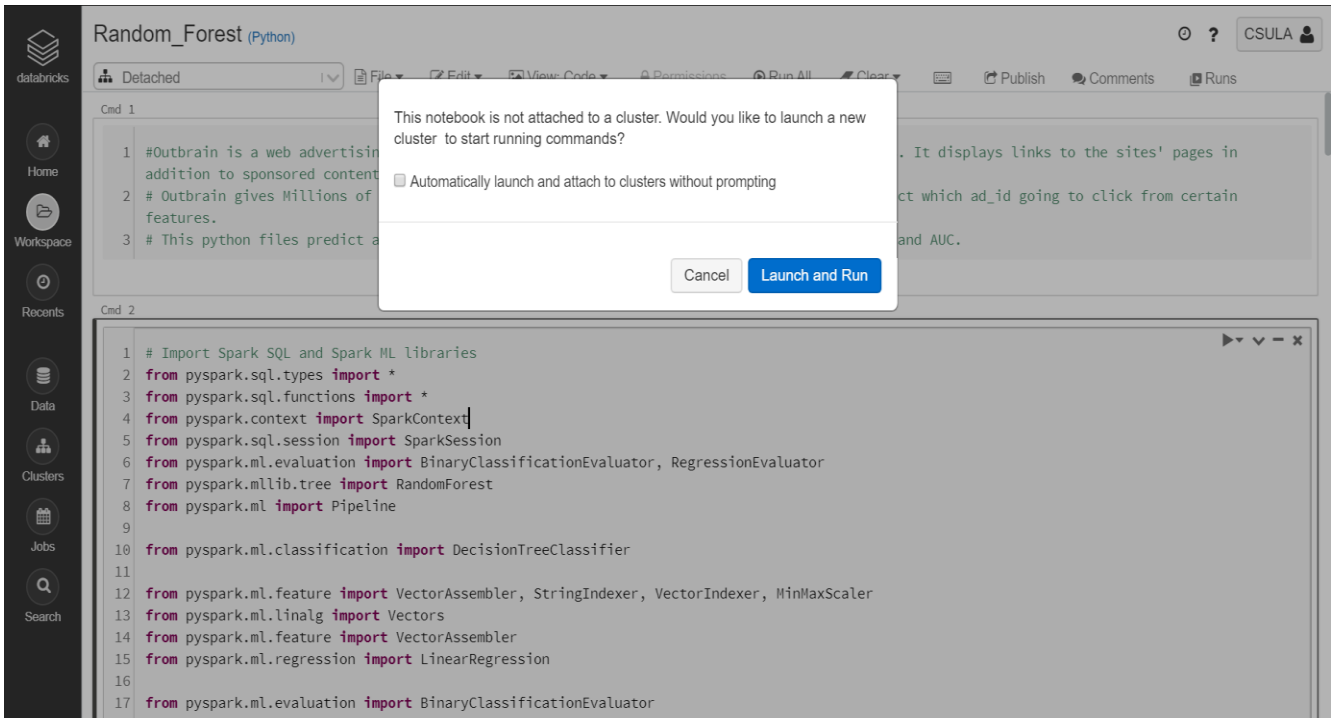
12. Select Import.



13. You are directed to the page below. **Play all codes** in the cells by selecting the **play** button:



14. If spark cluster is not attached, you will see the following window popped up. Mark the "Automatically Launch …" and select **"Attach and Run".**



15. Now, it runs all cells of the notebook imported. You will see the following at the end of the code:

Cmd 30

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluation = MulticlassClassificationEvaluator(
    labelCol="trueLabel", predictionCol="prediction", metricName="accuracy")
accuracy = evaluation.evaluate(prediction)
print(accuracy)
```

▸ (2) Spark Jobs

```
0.9520460358056266
```

Command took 2.89 minutes -- by dkansar@calstatela.edu at 5/12/2020, 11:53:35 AM on Project

Cmd 31

Following Code will give total Error in project

Cmd 32

```
print("Test Error = %g" % (1.0 - accuracy))
```

```
Test Error = 0.047954
```

Command took 0.03 seconds -- by dkansar@calstatela.edu at 5/12/2020, 11:53:35 AM on Project

Cmd 33

Random Forest Evaluator

Calculate AUC using Random Forest Evaluator.

Cmd 34

```
rf_evaluator =  MulticlassClassificationEvaluator(labelCol="trueLabel", predictionCol="prediction")
rf_auc = rf_evaluator.evaluate(prediction)
print("AUC for Random Forest is= ", rf_auc)
```

▸ (3) Spark Jobs

```
AUC for Random Forest is=  0.9499206293049177
```

Command took 4.14 minutes -- by dkansar@calstatela.edu at 5/12/2020, 11:53:35 AM on Project

Cmd 35

```
# Only for Classification Logistic Regression not for Linear Regression

tp = float(predicted.filter("prediction == 1.0 AND truelabel == 1").count())
fp = float(predicted.filter("prediction == 1.0 AND truelabel == 0").count())
tn = float(predicted.filter("prediction == 0.0 AND truelabel == 0").count())
fn = float(predicted.filter("prediction == 0.0 AND truelabel == 1").count())
metrics = spark.createDataFrame([
 ("TP", tp),
 ("FP", fp),
 ("TN", tn),
 ("FN", fn),
 ("Precision", tp / (tp + fp)),
 ("Recall", tp / (tp + fn))],["metric", "value"])
metrics.show()

```

▸ (7) Spark Jobs

▸ metrics: pyspark.sql.dataframe.DataFrame = [metric: string, value: double]

```
+---------+------------------+
|   metric|             value|
+---------+------------------+
|       TP|            1562.0|
|       FP|             138.0|
|       TN|           11839.0|
|       FN|             537.0|
|Precision|0.9188235294117647|
|   Recall|0.7441638875655073|
+---------+------------------+
```

Command took 4.42 minutes -- by dkansar@calstatela.edu at 5/12/2020, 12:49:16 PM on Project

Logistic Regression:

16. Now Follow **Step 10-14** and import Logistic_Regression.ipynb file.
17. Now run the notebook imported. You will see the following at the end of the code:

Cmd 26

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluation = MulticlassClassificationEvaluator(
    labelCol="trueLabel", predictionCol="prediction", metricName="accuracy")
accuracy = evaluation.evaluate(prediction)
print("Accuracy of Logistic Regression is: ",accuracy)
```

▸ (2) Spark Jobs

```
Accuracy of Logistic Regression is:  0.8530731429385371
```

Command took 17.39 seconds -- by dkansar@calstatela.edu at 5/12/2020, 11:31:11 PM on Project

Cmd 27

Following Code will give total Error in project

Cmd 28

```
print("Test Error = %g" % (1.0 - accuracy))
```

```
Test Error = 0.146927
```

Command took 0.03 seconds -- by dkansar@calstatela.edu at 5/12/2020, 11:31:11 PM on Project

Cmd 29

Logistic Regression Evaluator

Calculate AUC using Logistic Regression Evaluator.

Cmd 30

```
rf_evaluator =  MulticlassClassificationEvaluator(labelCol="trueLabel", predictionCol="prediction")
rf_auc = rf_evaluator.evaluate(prediction)
print("AUC for Logistic Regression is= ", rf_auc)
```

Cmd 31

```
tp = float(predicted.filter("prediction == 1.0 AND truelabel == 1").count())
fp = float(predicted.filter("prediction == 1.0 AND truelabel == 0").count())
tn = float(predicted.filter("prediction == 0.0 AND truelabel == 0").count())
fn = float(predicted.filter("prediction == 0.0 AND truelabel == 1").count())
metrics = spark.createDataFrame([
 ("TP", tp),
 ("FP", fp),
 ("TN", tn),
 ("FN", fn),
 ("Precision", tp ),
 ("Recall", tp / (tp + fn))],["metric", "value"])
metrics.show()

```

▸ (7) Spark Jobs

▸ metrics: pyspark.sql.dataframe.DataFrame = [metric: string, value: double]

```
+---------+--------+
|   metric|   value|
+---------+--------+
|       TP|     0.0|
|       FP|     0.0|
|       TN|11978.0|
|       FN| 2063.0|
|Precision|     0.0|
|   Recall|     0.0|
+---------+--------+
```

Command took 41.49 seconds -- by dkansar@calstatela.edu at 5/13/2020, 12:24:09 AM on Project

Gradient_Boosted Algorithm:

18. Now Follow **Step 10-14** and import Gradient_boost.ipynb file.
19. Now, it runs all cells of the notebook imported. You will see the following at the end of the code:

```
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
evaluation = MulticlassClassificationEvaluator(
    labelCol="trueLabel", predictionCol="prediction", metricName="accuracy")
accuracy = evaluation.evaluate(prediction)
print(accuracy)
```

▸ (2) Spark Jobs

```
0.8984530229917683
```

Command took 25.96 seconds -- by dkansar@calstatela.edu at 5/12/2020, 12:55:10 PM on Project

Cmd 27

Following Code will give total Error in project

Cmd 28

```
print("Test Error = %g" % (1.0 - accuracy))
```

```
Test Error = 0.101547
```

Command took 0.01 seconds -- by dkansar@calstatela.edu at 5/12/2020, 12:55:11 PM on Project

Cmd 29

Gradient Boosting Evaluator

Calculate AUC using Gradient Boosting Evaluator.

Cmd 30

```
rf_evaluator =  MulticlassClassificationEvaluator(labelCol="trueLabel", predictionCol="prediction")
rf_auc = rf_evaluator.evaluate(prediction)
print("AUC for Gradient Boost is= ", rf_auc)
```

▸ (3) Spark Jobs

```
AUC for Gradient Boost is=  0.8750931883418767
```

Cmd 31

```
# Only for Classification Logistic Regression not for Linear Regression

tp = float(predicted.filter("prediction == 1.0 AND truelabel == 1").count())
fp = float(predicted.filter("prediction == 1.0 AND truelabel == 0").count())
tn = float(predicted.filter("prediction == 0.0 AND truelabel == 0").count())
fn = float(predicted.filter("prediction == 0.0 AND truelabel == 1").count())
metrics = spark.createDataFrame([
 ("TP", tp),
 ("FP", fp),
 ("TN", tn),
 ("FN", fn),
 ("Precision", tp / (tp + fp)),
 ("Recall", tp / (tp + fn))],["metric", "value"])
metrics.show()
```

▸ (7) Spark Jobs

▸ metrics: pyspark.sql.dataframe.DataFrame = [metric: string, value: double]

```
+---------+-------------------+
|   metric|              value|
+---------+-------------------+
|       TP|              653.0|
|       FP|               10.0|
|       TN|            12008.0|
|       FN|             1421.0|
|Precision| 0.9849170437405732|
|   Recall|0.31485053037608485|
+---------+-------------------+
```

Command took 53.06 seconds -- by dkansar@calstatela.edu at 5/12/2020, 12:55:11 PM on Project

Decision Tree Classifier:

20. Now Follow **Step 10-14** and import Decision Tree Classifier.ipynb file.

21. Now, it runs all cells of the notebook imported. You will see the following at the end of the code:

Step 5: Ad prediction using spark in Oracle BDE

1. We need to import our dataset (event, train and Prmoted_content) to oracle BDE cluster. In order to do that we need to navigate to directory where we stored our csv files.

2. We can login to Oracle BDE by any terminal.
   Ssh dkansar@129.150.76.160

```
C:\Users\pdmuser>ssh dkansar@129.150.76.160
-- WARNING -- This system is for the use of authorized users only. Individuals
using this computer system without authority or in excess of their authority
are subject to having all their activities on this system monitored and
recorded by system personnel. Anyone using this system expressly consents to
such monitoring and is advised that if such monitoring reveals possible
evidence of criminal activity system personnel may provide the evidence of such
monitoring to law enforcement officials.

dkansar@129.150.76.160's password:
Last login: Fri May  8 04:45:41 2020 from 47.155.196.173
```

   Note: Username (dkansar) and IP address may be different in your case.

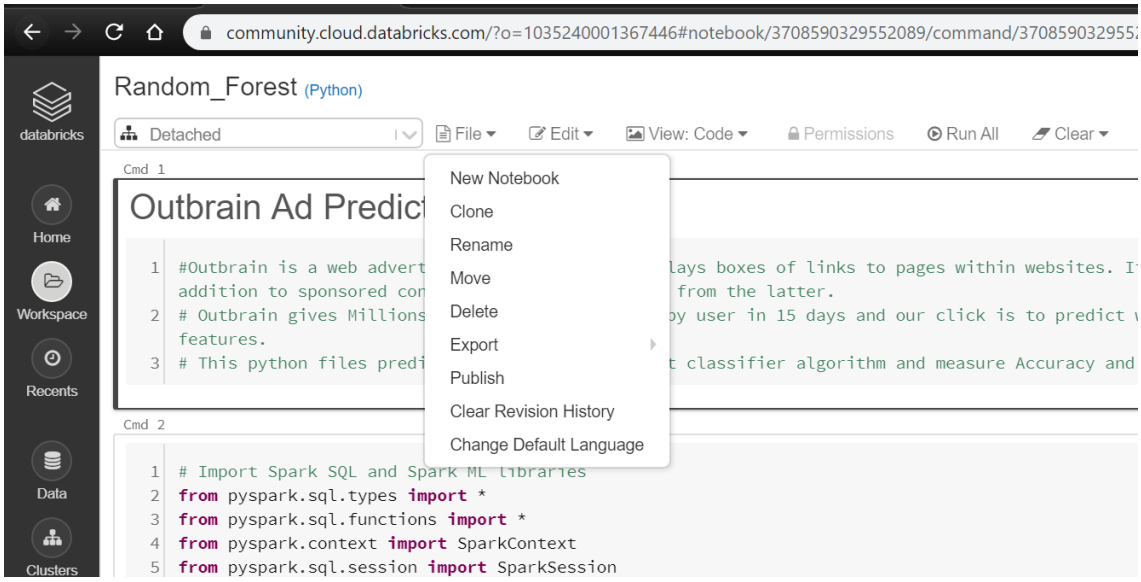
3. After navigating to directory open terminal your system and write following command.
   scp *.csv dkansar@129.150.76.160:~/

Note: 1: Instead of * you need to put different file names event, train and Promoted content. File name may also be different in your case.
2: Ip Address and Username (dkansar) may be different in your case.

4. Now we need to file our csv files to hdfs directory we can do it by following.
   a. Hdfs dfs -put event.csv (File name may be different.)
   b. Hdfs dfs -put train.csv (File name may be different.)
   c. Hdfs dfs -put Promoted_content.csv (File name may be different.)
5. Now we can check our file uploaded in hdfs.
   Hdfs dfs -ls

```
-bash-4.1$ hdfs dfs -ls
Found 9 items
-rw-r--r--   2 dkansar hdfs  285719991 2020-05-04 07:50 event.csv
-rw-r--r--   2 dkansar hdfs 1208549589 2020-05-02 18:23 eventd.csv
-rw-r--r--   2 dkansar hdfs   35715640 2020-05-06 20:22 events.csv
-rw-r--r--   2 dkansar hdfs   72088113 2020-05-05 21:19 flights.csv
drwxr-xr-x   - dkansar hdfs          0 2020-04-27 05:19 lda
-rw-r--r--   2 dkansar hdfs   13886609 2020-05-02 18:23 promoted_content.csv
-rw-r--r--   2 dkansar hdfs  234703736 2020-05-04 07:25 train.csv
-rw-r--r--   2 dkansar hdfs 1159782400 2020-04-30 08:11 traind.csv
-rw-r--r--   2 dkansar hdfs   36395707 2020-05-06 20:18 trains.csv
-bash-4.1$
```

6. Now we have our csv files in hdfs.
7. In step 3, we have predicted data in data bricks now you need to login to databricks and open Logistic_Regression file. Now browse File>Export>source file as shown in following image.



Using the same step, you need to download Gradient_Boosted.py and Random_Forest.py file.

4. Now open Python notebook file in any editor and make change as follows.
   IS_SPARK_SUBMIT_CLI = True (change from false to True)

5. After navigating to directory open terminal your system and write following command.
   scp *.py dkansar@129.150.76.160:~/

Note: 1: Instead of * you need to put different file names Logistic Regression, Gradient boost and Random_Forest, Decision Tree Classifier. File name may also be different in your case.
2: Ip Address and Username (dkansar) may be different in your case.

6. Now we need run Logistic Regression in Oracle BDE as follows.
   Spark-submit Logistic_Regression.py
   Now we can measure Accuracy and AUC as follows.
   Note: In Oracle BDCE you can only able to see Result of last command Confusion Matrix.
   1) If you want to see result of AUC Comment out Confusion Matrix Code (you can Comment line by just putting # in front of line). Then upload and save file using Step 5 and 6 you can able to see AUC of algorithm.
   2) If you want to see result of Accuracy Comment out Confusion Matrix Code (you and AUC Commands Comment line by just putting # in front of line). Then upload and run file using step 5 and 6 you able to see Accuracy of Algorithm.

Accuracy:

```
20/05/04 22:55:21 INFO Executor: Finished task 48.0 in stage 143.0 (TID 6257). 1827 bytes resu
20/05/04 22:55:21 INFO TaskSetManager: Finished task 48.0 in stage 143.0 (TID 6257) in 16 ms o
20/05/04 22:55:21 INFO TaskSchedulerImpl: Removed TaskSet 143.0, whose tasks have all complete
20/05/04 22:55:21 INFO DAGScheduler: ResultStage 143 (countByValue at MulticlassMetrics.scala:
20/05/04 22:55:21 INFO DAGScheduler: Job 31 finished: countByValue at MulticlassMetrics.scala:
.830304235215
Test Error = 0.169696
20/05/04 22:55:21 INFO SparkContext: Invoking stop() from shutdown hook
20/05/04 22:55:21 INFO SparkUI: Stopped Spark web UI at http://10.129.201.54:4041
20/05/04 22:55:21 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
20/05/04 22:55:21 INFO MemoryStore: MemoryStore cleared
```

AUC:

```
20/05/07 05:20:27 INFO TaskSchedulerImpl: Removed TaskSet 161.0, whose tasks have all completed, from pool
20/05/07 05:20:27 INFO DAGScheduler: ResultStage 161 (collectAsMap at MulticlassMetrics.scala:53) finished in 0.202
20/05/07 05:20:27 INFO DAGScheduler: Job 34 finished: collectAsMap at MulticlassMetrics.scala:53, took 4.343779 s
('AUC for Logistic Regression is= ', 0.7539270092721175)
20/05/07 05:20:27 INFO SparkContext: Invoking stop() from shutdown hook
```

Confusion Matrix:

```
+---------+-------+
|   metric|  value|
+---------+-------+
|       TP|    0.0|
|       FP|    0.0|
|       TN|13086.0|
|       FN| 2508.0|
|Precision|    0.0|
|   Recall|    0.0|
+---------+-------+

20/05/13 05:15:48 INFO SparkContext: Invoking stop() from shutdown hook
20/05/13 05:15:48 INFO SparkUI: Stopped Spark web UI at http://10.129.201.54:4041
20/05/13 05:15:48 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
```

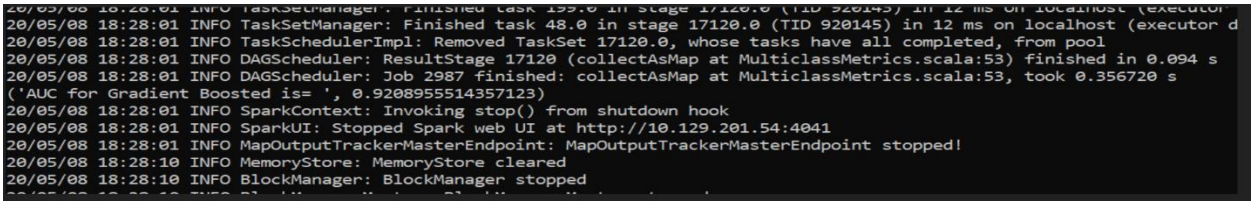
7. Now we need run Gradient_Boosted in Oracle BDE as follows.
   Spark-submit Gradient_Boosted.py

   Now we can measure Accuracy and AUC as follows.

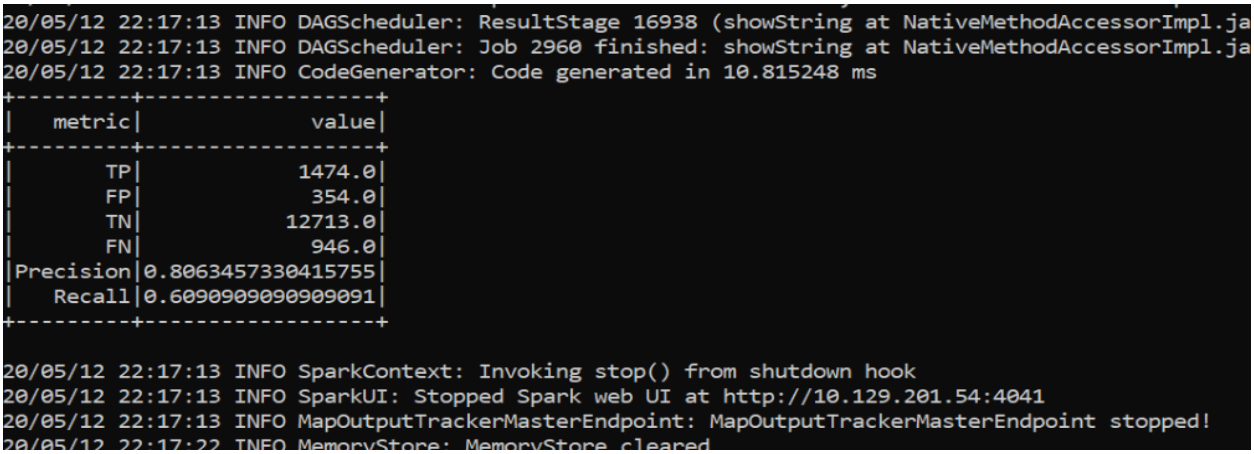
Accuracy:

```
20/05/08 06:46:39 INFO TaskSchedulerImpl: Removed TaskSet 17078.0, whose tasks have all completed, from pool
20/05/08 06:46:39 INFO DAGScheduler: ResultStage 17078 (countByValue at MulticlassMetrics.scala:42) finished in 0.108 s
20/05/08 06:46:39 INFO DAGScheduler: Job 2980 finished: countByValue at MulticlassMetrics.scala:42, took 0.528726 s
('Accuracy of for Gradient Boosted is= ', 0.9247810743693635)
Test Error = 0.0752189
20/05/08 06:46:39 INFO SparkContext: Invoking stop() from shutdown hook
```

AUC:

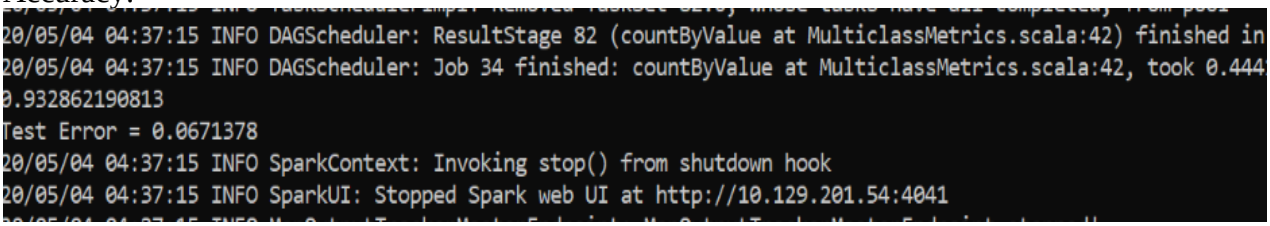


Confusion Matrix:

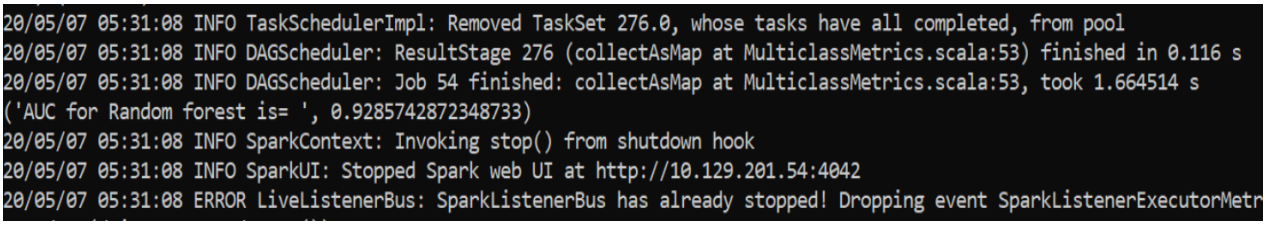


8. Now we need run Random_Forest in Oracle BDE as follows.
Spark-submit Random_Forest.py

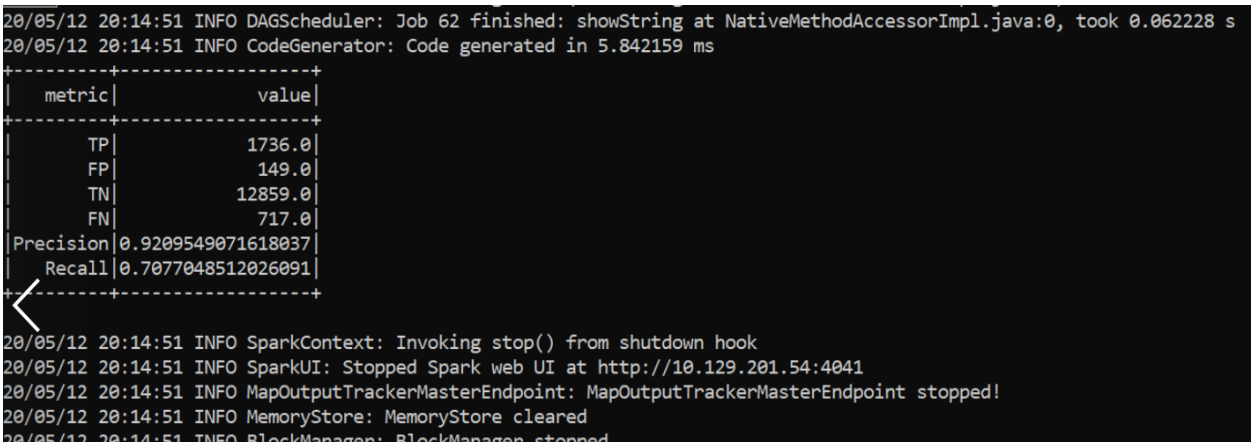
Now we can measure Accuracy and AUC as follows.
Accuracy:



AUC:

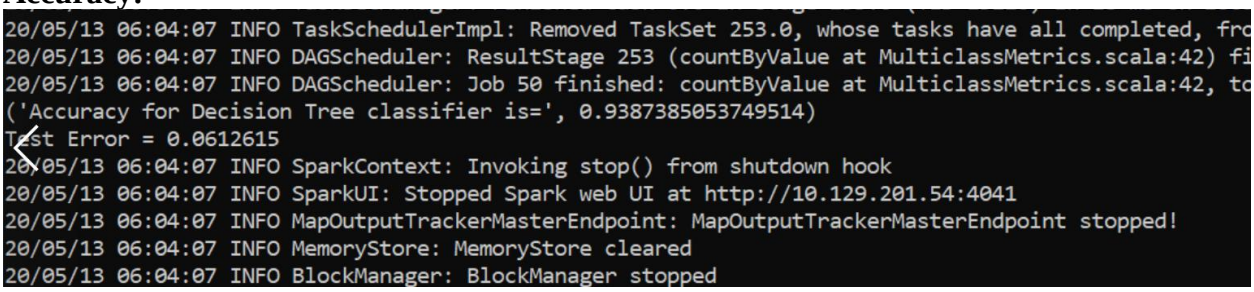


Confusion Matrix:

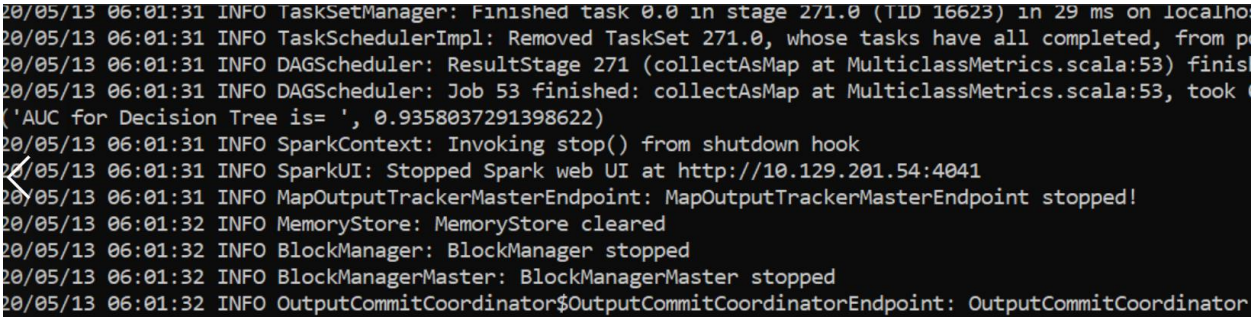


9. Now we need run Random_Forest in Oracle BDE as follows.
   Spark-submit Decision Tree Classifier.py

   Now we can measure Accuracy and AUC as follows.
   Accuracy:



AUC:



Confusion Matrix: