CIS5560 Term Project Tutorial

Authors: Tanvi Gawade, Swarnim Jambhule, Sushant Burde

Instructor: [Jongwook Woo]

Date: 05/19/2019

Lab Tutorial

Tanvi (tgawade@calstatela.edu)

Swarnim (sjambhu@calstatela.edu)

Sushant (sburde@calstatela.edu)

05/19/2019

Classification for Floating or Non-Floating Items from the Library Inventory On Data Bricks in Spark Machine Leaning

Objectives

List what your objectives are. In this hands-on lab, you will learn how to:

- Get data manually
- Create Spark cluster

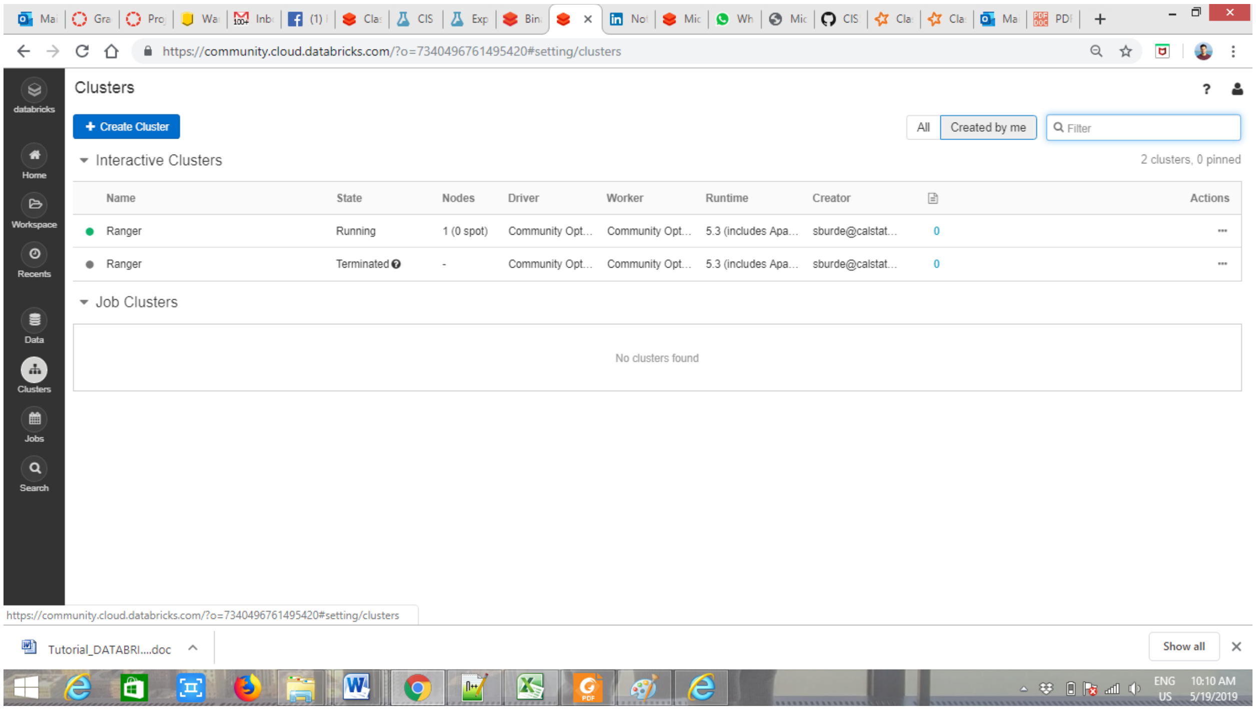
- Writing PySpark codes to develop a predictive model.
- Classification of Floating and Non-Floating values using Random Forest Classification and Gradient Boosting Tree Classification.
- Visualization

Platform Spec

- Data Bricks PySpark
- Databricks Runtime Version: 5.2(Incl. Apache Spark 2.4.0, Scala 2.11)
- Execution: Single Node
- Memory: 6GB Capacity

Step 1: Creating a Cluster in Data Bricks

1. This step is to create a cluster for the execution of the codes.



Clusters

Interactive Clusters — 2 clusters, 0 pinned

	Name	State	Nodes	Driver	Worker	Runtime	Creator		Actions
●	Ranger	Running	1 (0 spot)	Community Opt...	Community Opt...	5.3 (includes Apa...	sburde@calstat...	0	···
●	Ranger	Terminated	-	Community Opt...	Community Opt...	5.3 (includes Apa...	sburde@calstat...	0	···

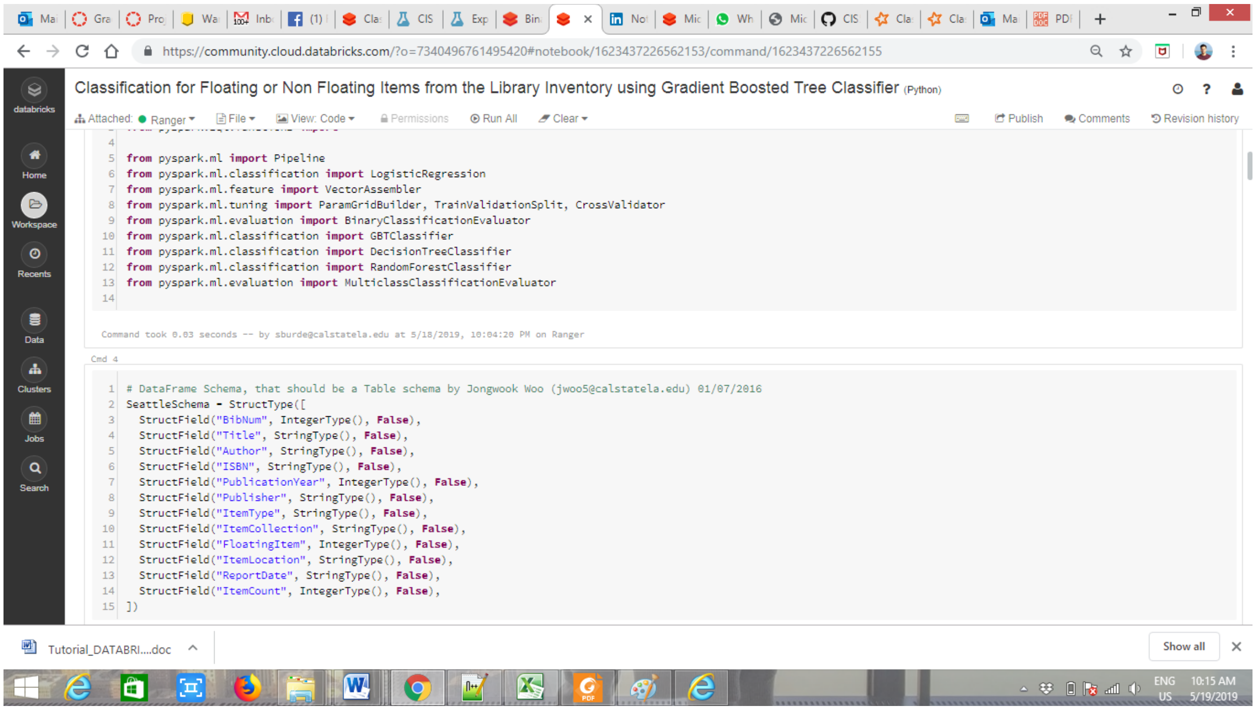
Job Clusters

No clusters found

Properties: -

- Python Version: 2
- Driver Type: Community Optimized
- Availability Zone: us-west-2c

Step 2: Prepare the Data



https://community.cloud.databricks.com/?o=7340496761495420#notebook/1623437226562153/command/1623437226562155
Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)
Attached: Ranger
File
View: Code
Permissions
Run All
Clear
Publish
Comments
Revision history
databricks
Home
Workspace
Recents
Data
Clusters
Jobs
Search
Tutorial_DATABRI....doc
Show all

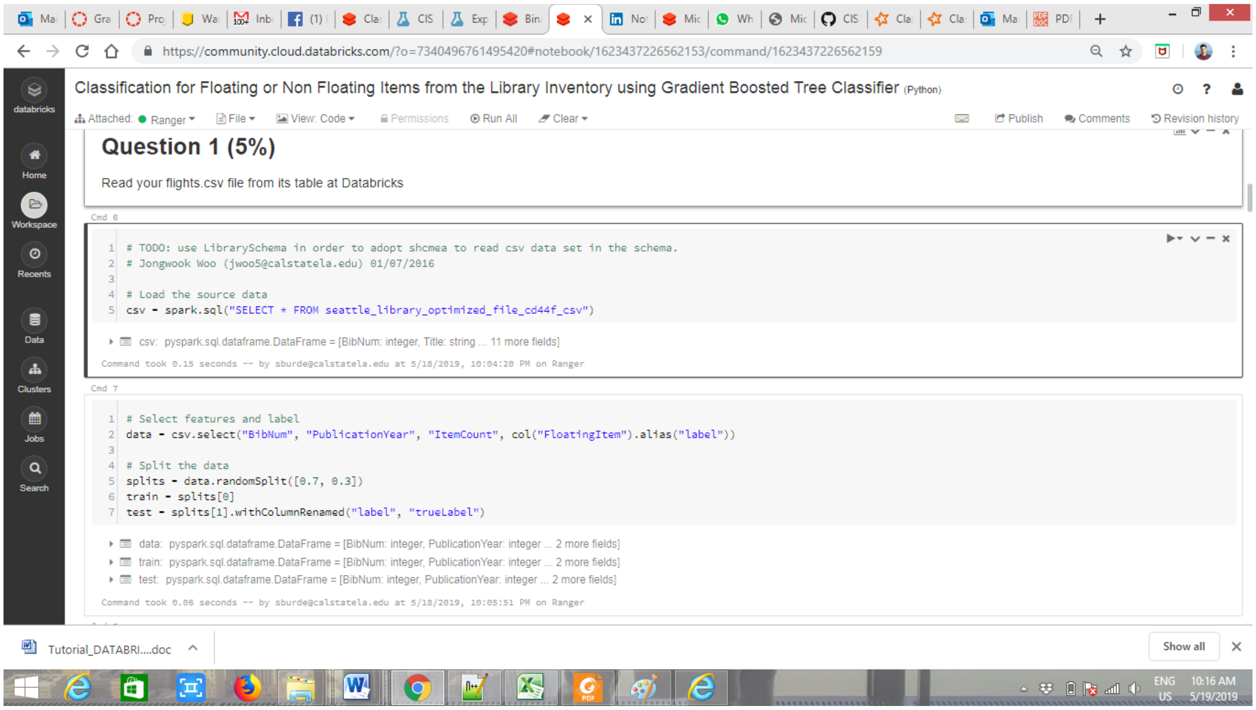
```
from pyspark.ml import Pipeline
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.tuning import ParamGridBuilder, TrainValidationSplit, CrossValidator
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.classification import GBTClassifier
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator

```

```
Command took 0.03 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:04:20 PM on Ranger
```

Cmd 4

```
# DataFrame Schema, that should be a Table schema by Jongwook Woo (jwoo5@calstatela.edu) 01/07/2016
SeattleSchema = StructType([
  StructField("BibNum", IntegerType(), False),
  StructField("Title", StringType(), False),
  StructField("Author", StringType(), False),
  StructField("ISBN", StringType(), False),
  StructField("PublicationYear", IntegerType(), False),
  StructField("Publisher", StringType(), False),
  StructField("ItemType", StringType(), False),
  StructField("ItemCollection", StringType(), False),
  StructField("FloatingItem", IntegerType(), False),
  StructField("ItemLocation", StringType(), False),
  StructField("ReportDate", StringType(), False),
  StructField("ItemCount", IntegerType(), False),
])
```

Step 3: Select data from the table



Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)

Question 1 (5%)

Read your flights.csv file from its table at Databricks

Cmd 6

```
# TODO: use LibrarySchema in order to adopt shcmea to read csv data set in the schema.
# Jongwook Woo (jwoo5@calstatela.edu) 01/07/2016

# Load the source data
csv = spark.sql("SELECT * FROM seattle_library_optimized_file_cd44f_csv")
```

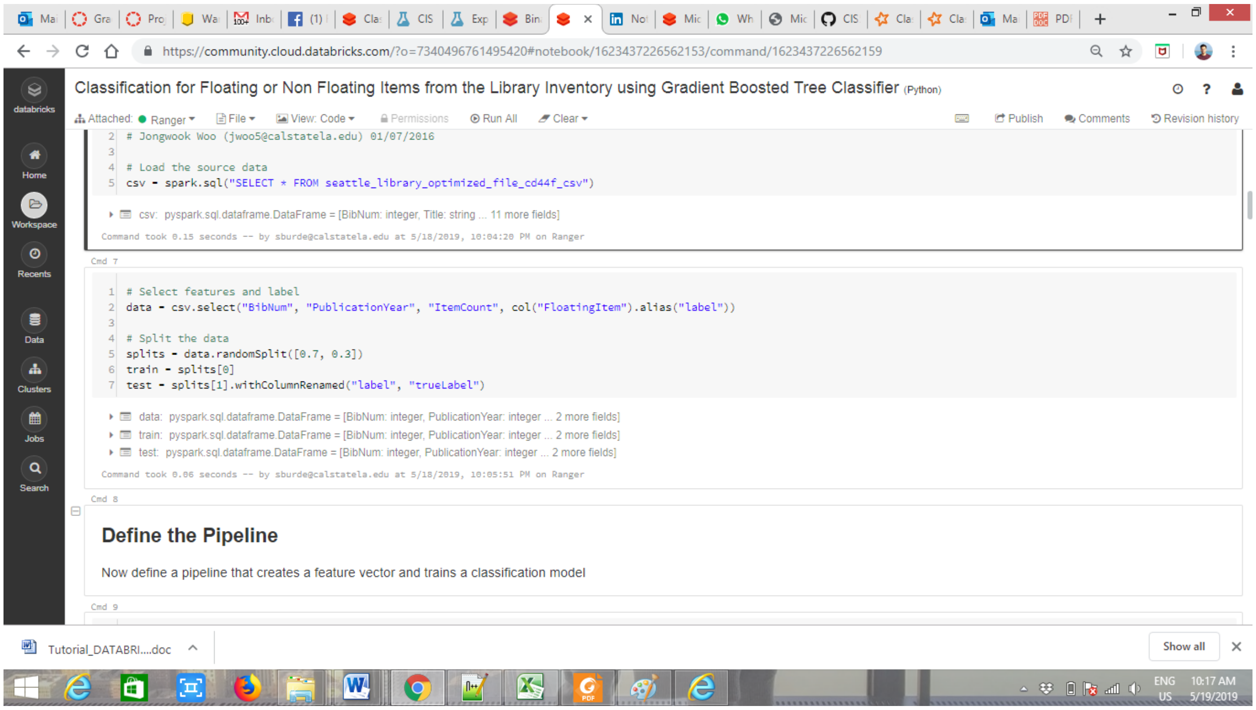
```
csv:  pyspark.sql.dataframe.DataFrame = [BibNum: integer, Title: string ... 11 more fields]
Command took 0.15 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:04:20 PM on Ranger
```

Cmd 7

```
# Select features and label
data = csv.select("BibNum", "PublicationYear", "ItemCount", col("FloatingItem").alias("label"))

# Split the data
splits = data.randomSplit([0.7, 0.3])
train = splits[0]
test = splits[1].withColumnRenamed("label", "trueLabel")
```

```
data:  pyspark.sql.dataframe.DataFrame = [BibNum: integer, PublicationYear: integer ... 2 more fields]
train:  pyspark.sql.dataframe.DataFrame = [BibNum: integer, PublicationYear: integer ... 2 more fields]
test:  pyspark.sql.dataframe.DataFrame = [BibNum: integer, PublicationYear: integer ... 2 more fields]
Command took 0.06 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:05:51 PM on Ranger
```

Step 4: Define for Train and Test the Data. (0.7 – Train, 0.3 – Test)



https://community.cloud.databricks.com/?o=7340496761495420#notebook/1623437226562153/command/1623437226562159

Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)

```
# Jongwook Woo (jwoo5@calstatela.edu) 01/07/2016

# Load the source data
csv = spark.sql("SELECT * FROM seattle_library_optimized_file_cd44f_csv")
```

```
csv:  pyspark.sql.dataframe.DataFrame = [BibNum: integer, Title: string ... 11 more fields]
Command took 0.15 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:04:20 PM on Ranger
```

Cmd 7

```
# Select features and label
data = csv.select("BibNum", "PublicationYear", "ItemCount", col("FloatingItem").alias("label"))

# Split the data
splits = data.randomSplit([0.7, 0.3])
train = splits[0]
test = splits[1].withColumnRenamed("label", "trueLabel")
```

```
data:  pyspark.sql.dataframe.DataFrame = [BibNum: integer, PublicationYear: integer ... 2 more fields]
train:  pyspark.sql.dataframe.DataFrame = [BibNum: integer, PublicationYear: integer ... 2 more fields]
test:  pyspark.sql.dataframe.DataFrame = [BibNum: integer, PublicationYear: integer ... 2 more fields]
Command took 0.06 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:05:51 PM on Ranger
```

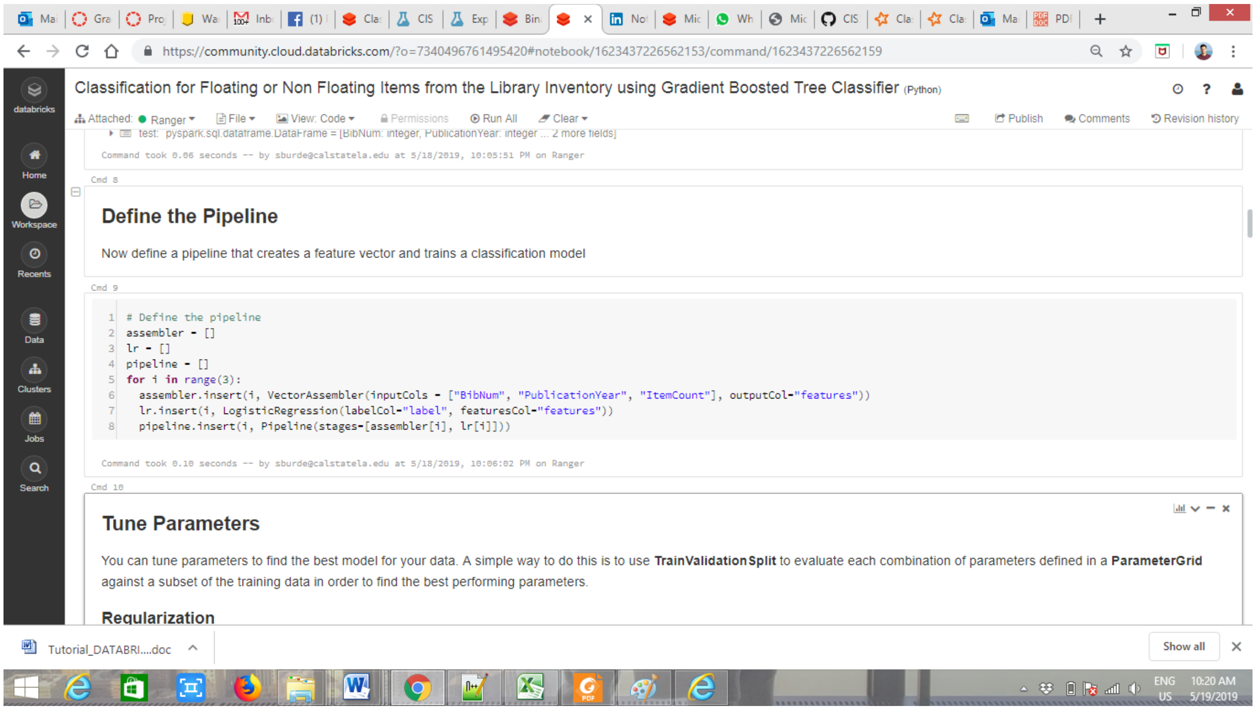
Cmd 8

Define the Pipeline

Now define a pipeline that creates a feature vector and trains a classification model

Cmd 9

Step 5: Define Pipeline for the data



https://community.cloud.databricks.com/?o=7340496761495420#notebook/1623437226562153/command/1623437226562159

Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)

Attached: ● Ranger

test: pyspark.sql.dataframe.DataFrame = [BibNum: integer, PublicationYear: integer ... 2 more fields]

```
Command took 0.06 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:05:51 PM on Ranger
```

Cmd 8

Define the Pipeline

Now define a pipeline that creates a feature vector and trains a classification model

Cmd 9

```python
# Define the pipeline
assembler = []
lr = []
pipeline = []
for i in range(3):
  assembler.insert(i, VectorAssembler(inputCols = ["BibNum", "PublicationYear", "ItemCount"], outputCol="features"))
  lr.insert(i, LogisticRegression(labelCol="label", featuresCol="features"))
  pipeline.insert(i, Pipeline(stages=[assembler[i], lr[i]]))
```

```
Command took 0.10 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:06:02 PM on Ranger
```

Cmd 10

Tune Parameters

You can tune parameters to find the best model for your data. A simple way to do this is to use **TrainValidationSplit** to evaluate each combination of parameters defined in a **ParameterGrid** against a subset of the training data in order to find the best performing parameters.

Regularization

Step 5: Tune the parameter

Tune Parameters

You can tune parameters to find the best model for your data. A simple way to do this is to use **TrainValidationSplit** to evaluate each combination of parameters defined in a **ParameterGrid** against a subset of the training data in order to find the best performing parameters.

Regularization

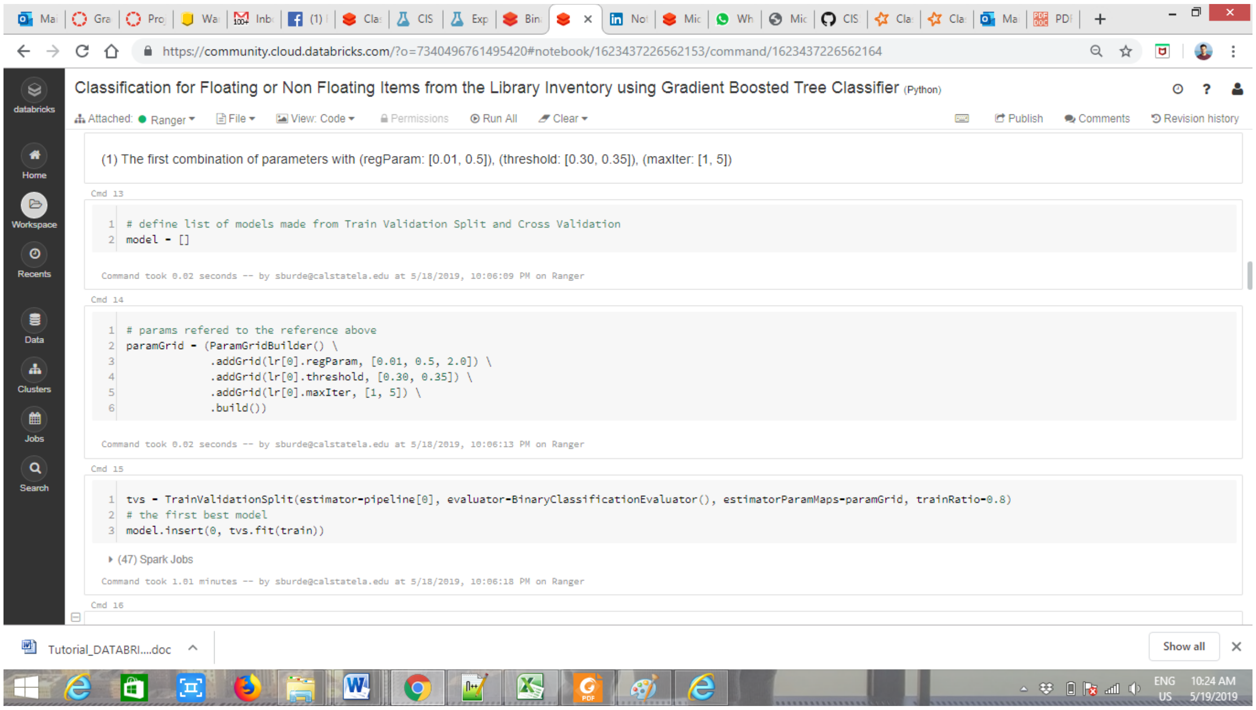
is a way of avoiding Imbalances in the way that the data is trained against the training data so that the model ends up being over fit to the training data. In other words It works really well with the training data but it doesn't generalize well with other data. That we can use a **regularization parameter** to vary the way that the model balances that way.

Training ratio of 0.7

it's going to use 70% of the the data that it's got in its training set to train the model and then the remaining 30% is going to use to validate the trained model.

In **ParamGridBuilder**, all possible combinations are generated from regParam, maxIter, threshold. So it is going to try each combination of the parameters with 70% of the the data to train the model and 30% to to validate it.

Step 5a: Train Validation Split with Threshold parameters



https://community.cloud.databricks.com/?o=7340496761495420#notebook/1623437226562153/command/1623437226562164
Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)
Attached: Ranger
File
View: Code
Permissions
Run All
Clear
Publish
Comments
Revision history
databricks
Home
Workspace
Recents
Data
Clusters
Jobs
Search
Tutorial_DATABRI....doc
Show all
ENG
US
10:24 AM
5/19/2019

(1) The first combination of parameters with (regParam: [0.01, 0.5]), (threshold: [0.30, 0.35]), (maxIter: [1, 5])

Cmd 13

```
# define list of models made from Train Validation Split and Cross Validation
model = []
```

```
Command took 0.02 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:06:09 PM on Ranger
```

Cmd 14

```
# params refered to the reference above
paramGrid = (ParamGridBuilder() \
             .addGrid(lr[0].regParam, [0.01, 0.5, 2.0]) \
             .addGrid(lr[0].threshold, [0.30, 0.35]) \
             .addGrid(lr[0].maxIter, [1, 5]) \
             .build())
```

```
Command took 0.02 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:06:13 PM on Ranger
```

Cmd 15

```
tvs = TrainValidationSplit(estimator=pipeline[0], evaluator=BinaryClassificationEvaluator(), estimatorParamMaps=paramGrid, trainRatio=0.8)
# the first best model
model.insert(0, tvs.fit(train))
```

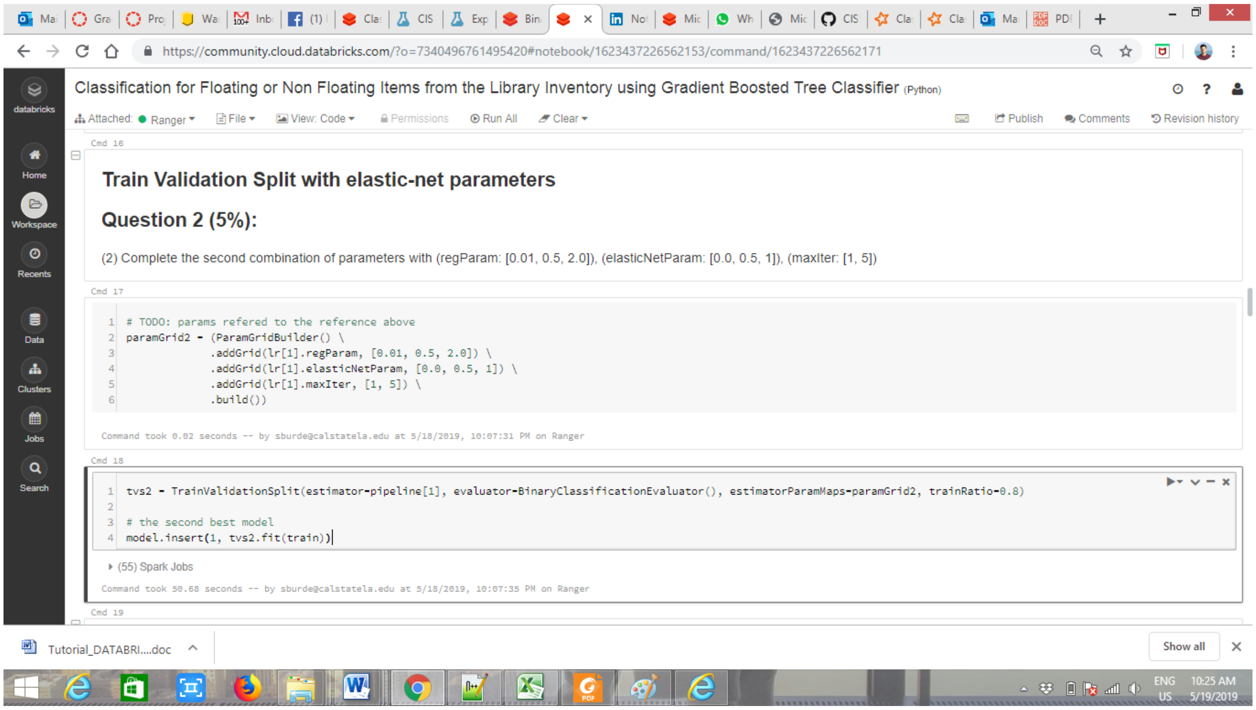
▸ (47) Spark Jobs

```
Command took 1.01 minutes -- by sburde@calstatela.edu at 5/18/2019, 10:06:18 PM on Ranger
```

Cmd 16

Step 5b: Train Validation Split with elastic-net parameters

The second combination of parameters with (regParam: [0.01, 0.5, 2.0]), (elasticNetParam: [0.0, 0.5, 1]), (maxIter: [1, 5])



Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)

Cmd 16

Train Validation Split with elastic-net parameters

Question 2 (5%):

(2) Complete the second combination of parameters with (regParam: [0.01, 0.5, 2.0]), (elasticNetParam: [0.0, 0.5, 1]), (maxIter: [1, 5])

Cmd 17

```
# TODO: params refered to the reference above
paramGrid2 = (ParamGridBuilder() \
             .addGrid(lr[1].regParam, [0.01, 0.5, 2.0]) \
             .addGrid(lr[1].elasticNetParam, [0.0, 0.5, 1]) \
             .addGrid(lr[1].maxIter, [1, 5]) \
             .build())
```

Command took 0.02 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:07:31 PM on Ranger

Cmd 18

```
tvs2 = TrainValidationSplit(estimator=pipeline[1], evaluator=BinaryClassificationEvaluator(), estimatorParamMaps=paramGrid2, trainRatio=0.8)

# the second best model
model.insert(1, tvs2.fit(train))
```

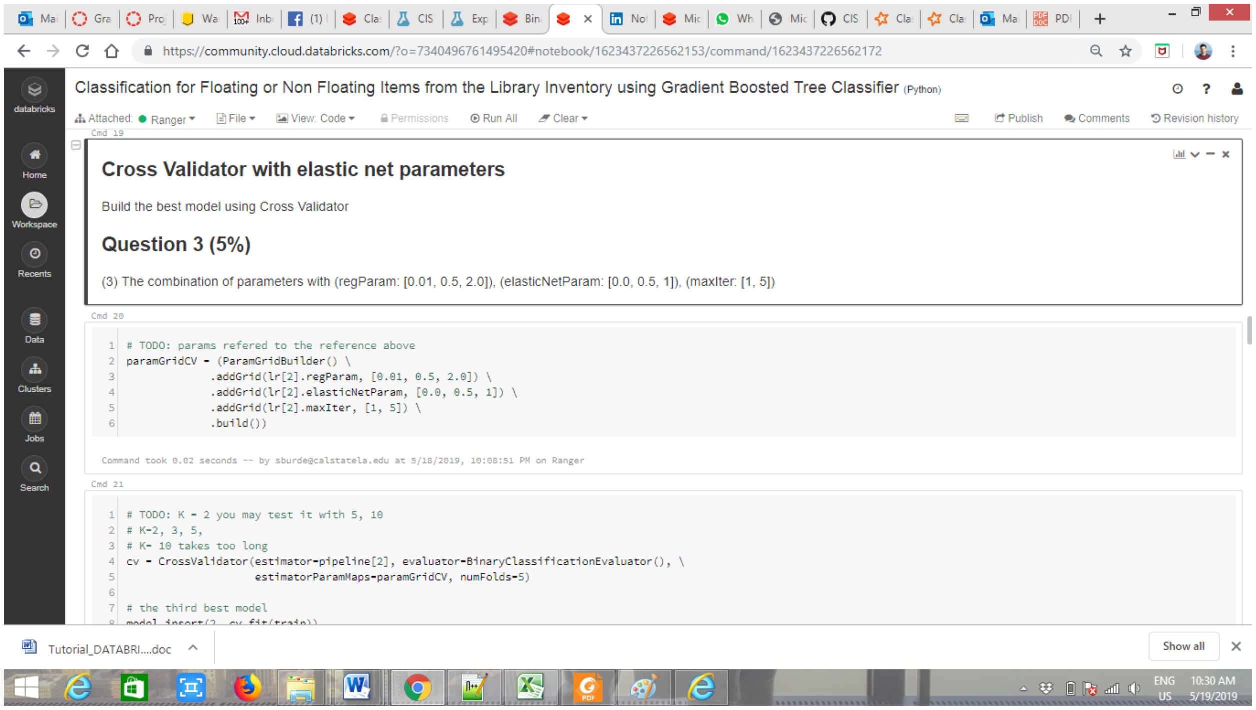
(55) Spark Jobs

Command took 50.68 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:07:35 PM on Ranger

Cmd 19

Step 5c: Cross Validator with elastic net parameters

The combination of parameters with (regParam: [0.01, 0.5, 2.0]), (elasticNetParam: [0.0, 0.5, 1]), (maxIter: [1, 5])



https://community.cloud.databricks.com/?o=7340496761495420#notebook/1623437226562153/command/1623437226562172

Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)

Cmd 19

Cross Validator with elastic net parameters

Build the best model using Cross Validator

Question 3 (5%)

(3) The combination of parameters with (regParam: [0.01, 0.5, 2.0]), (elasticNetParam: [0.0, 0.5, 1]), (maxIter: [1, 5])

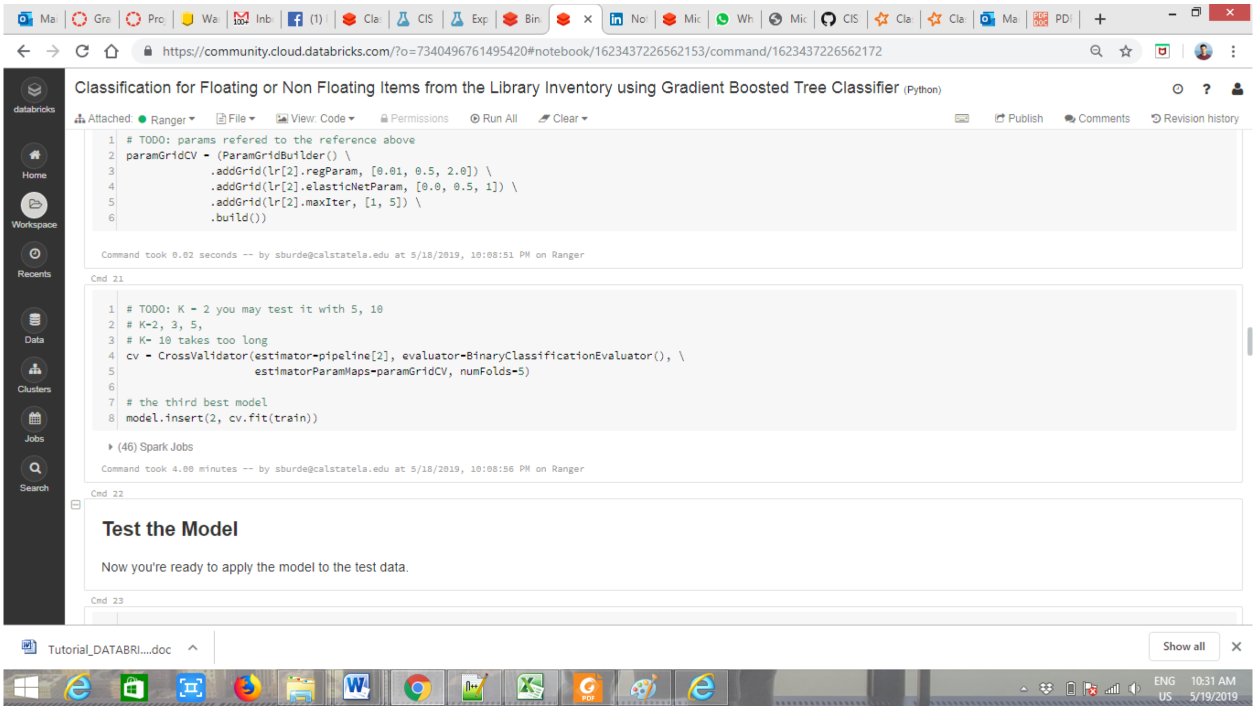
Cmd 20

```
# TODO: params refered to the reference above
paramGridCV = (ParamGridBuilder() \
            .addGrid(lr[2].regParam, [0.01, 0.5, 2.0]) \
            .addGrid(lr[2].elasticNetParam, [0.0, 0.5, 1]) \
            .addGrid(lr[2].maxIter, [1, 5]) \
            .build())
```

Command took 0.02 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:08:51 PM on Ranger

Cmd 21

```
# TODO: K = 2 you may test it with 5, 10
# K=2, 3, 5,
# K= 10 takes too long
cv = CrossValidator(estimator=pipeline[2], evaluator=BinaryClassificationEvaluator(), \
                    estimatorParamMaps=paramGridCV, numFolds=5)

# the third best model
model.insert(2, cv.fit(train))
```



https://community.cloud.databricks.com/?o=7340496761495420#notebook/1623437226562153/command/1623437226562172

Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)

```
# TODO: params refered to the reference above
paramGridCV = (ParamGridBuilder() \
            .addGrid(lr[2].regParam, [0.01, 0.5, 2.0]) \
            .addGrid(lr[2].elasticNetParam, [0.0, 0.5, 1]) \
            .addGrid(lr[2].maxIter, [1, 5]) \
            .build())
```

Command took 0.02 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:08:51 PM on Ranger

Cmd 21

```
# TODO: K = 2 you may test it with 5, 10
# K=2, 3, 5,
# K= 10 takes too long
cv = CrossValidator(estimator=pipeline[2], evaluator=BinaryClassificationEvaluator(), \
                    estimatorParamMaps=paramGridCV, numFolds=5)

# the third best model
model.insert(2, cv.fit(train))
```

(46) Spark Jobs

Command took 4.00 minutes -- by sburde@calstatela.edu at 5/18/2019, 10:08:56 PM on Ranger

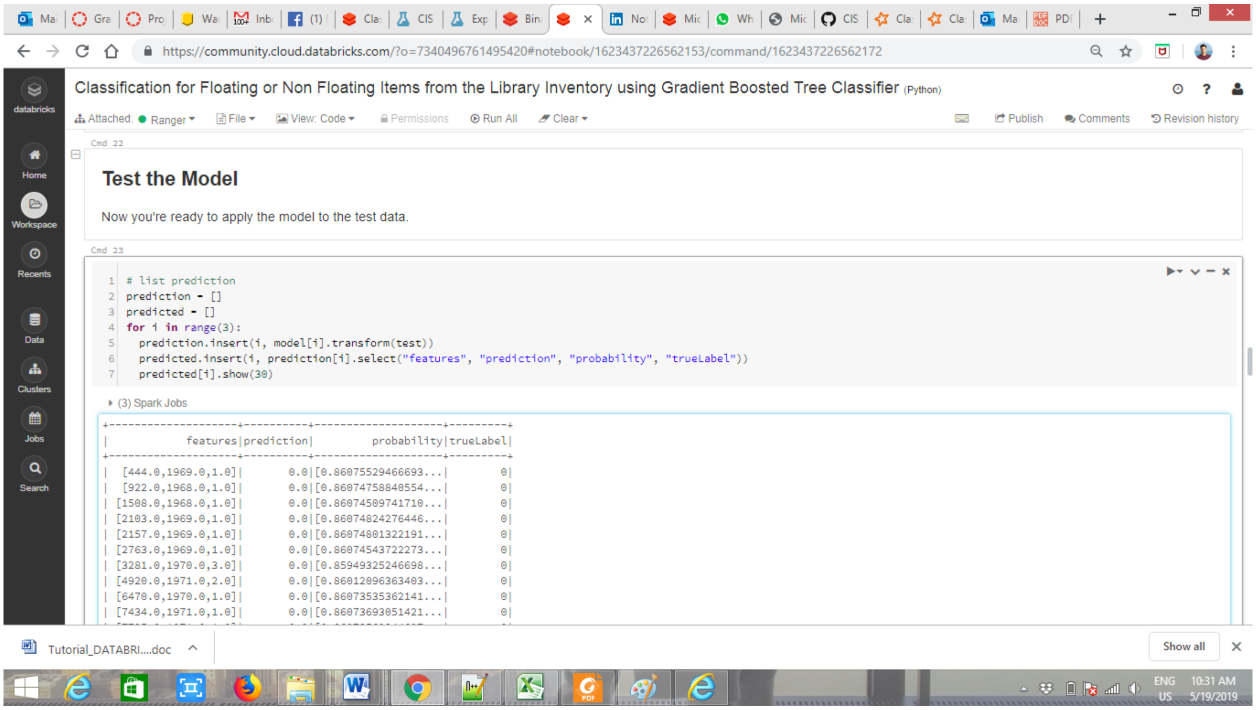
Cmd 22

Test the Model

Now you're ready to apply the model to the test data.

Cmd 23

Step 6: Test The model



Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)

Cmd 22

Test the Model

Now you're ready to apply the model to the test data.

Cmd 23

```
# list prediction
prediction = []
predicted = []
for i in range(3):
  prediction.insert(i, model[i].transform(test))
  predicted.insert(i, prediction[i].select("features", "prediction", "probability", "trueLabel"))
  predicted[i].show(30)
```

(3) Spark Jobs

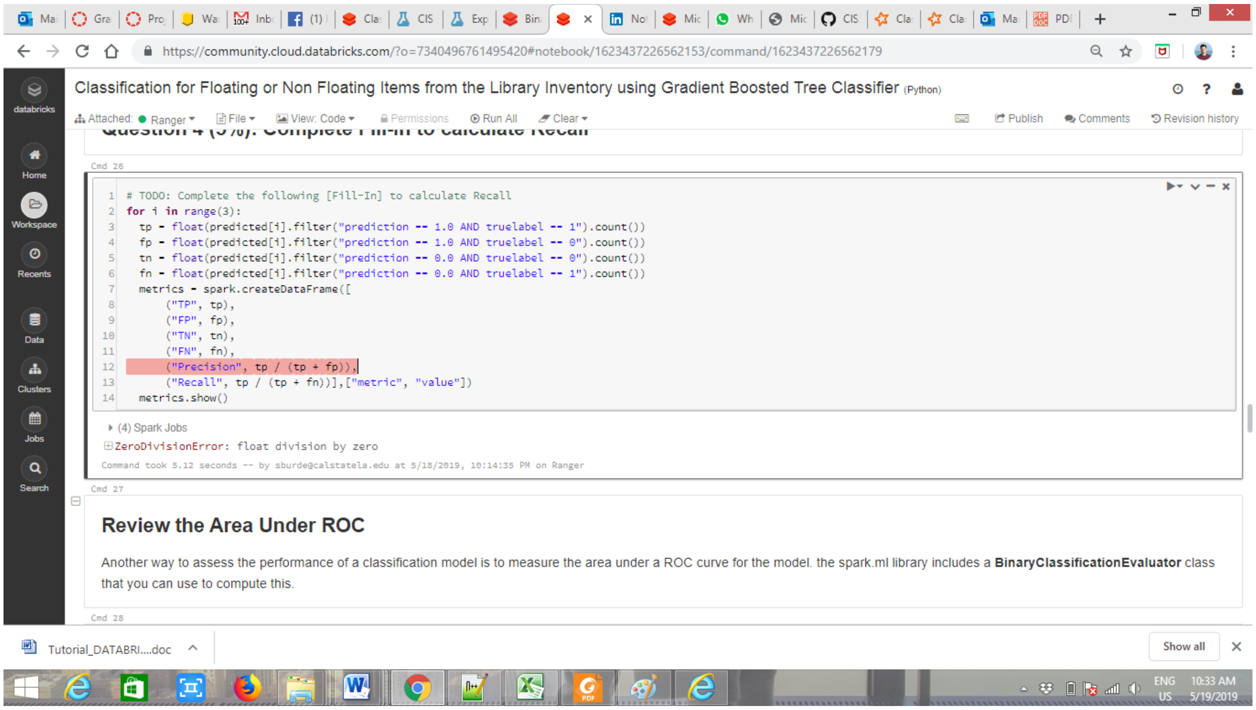
```
+-------------------+----------+--------------------+---------+
|           features|prediction|         probability|trueLabel|
+-------------------+----------+--------------------+---------+
|  [444.0,1969.0,1.0]|       0.0|[0.86075529466693...|        0|
|  [922.0,1968.0,1.0]|       0.0|[0.86074758840554...|        0|
| [1508.0,1968.0,1.0]|       0.0|[0.86074509741710...|        0|
| [2103.0,1969.0,1.0]|       0.0|[0.86074824276446...|        0|
| [2157.0,1969.0,1.0]|       0.0|[0.86074801322191...|        0|
| [2763.0,1969.0,1.0]|       0.0|[0.86074543722273...|        0|
| [3281.0,1970.0,3.0]|       0.0|[0.85949325246698...|        0|
| [4920.0,1971.0,2.0]|       0.0|[0.86012096363403...|        0|
| [6470.0,1970.0,1.0]|       0.0|[0.86073535362141...|        0|
| [7434.0,1971.0,1.0]|       0.0|[0.86073693051421...|        0|
```

Step 7: Compute Confusion Matrix Metrics

Classifiers are typically evaluated by creating a confusion matrix, which indicates the number of:

- True Positives
- True Negatives
- False Positives
- False Negatives

From these core measures, other evaluation metrics such as precision and recall can be calculated.



Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)

Question 4 (5%): Complete Fill-in to calculate Recall

Cmd 26

```
# TODO: Complete the following [Fill-In] to calculate Recall
for i in range(3):
  tp = float(predicted[i].filter("prediction == 1.0 AND truelabel == 1").count())
  fp = float(predicted[i].filter("prediction == 1.0 AND truelabel == 0").count())
  tn = float(predicted[i].filter("prediction == 0.0 AND truelabel == 0").count())
  fn = float(predicted[i].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()
```

(4) Spark Jobs

```
ZeroDivisionError: float division by zero
Command took 5.12 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:14:35 PM on Ranger
```

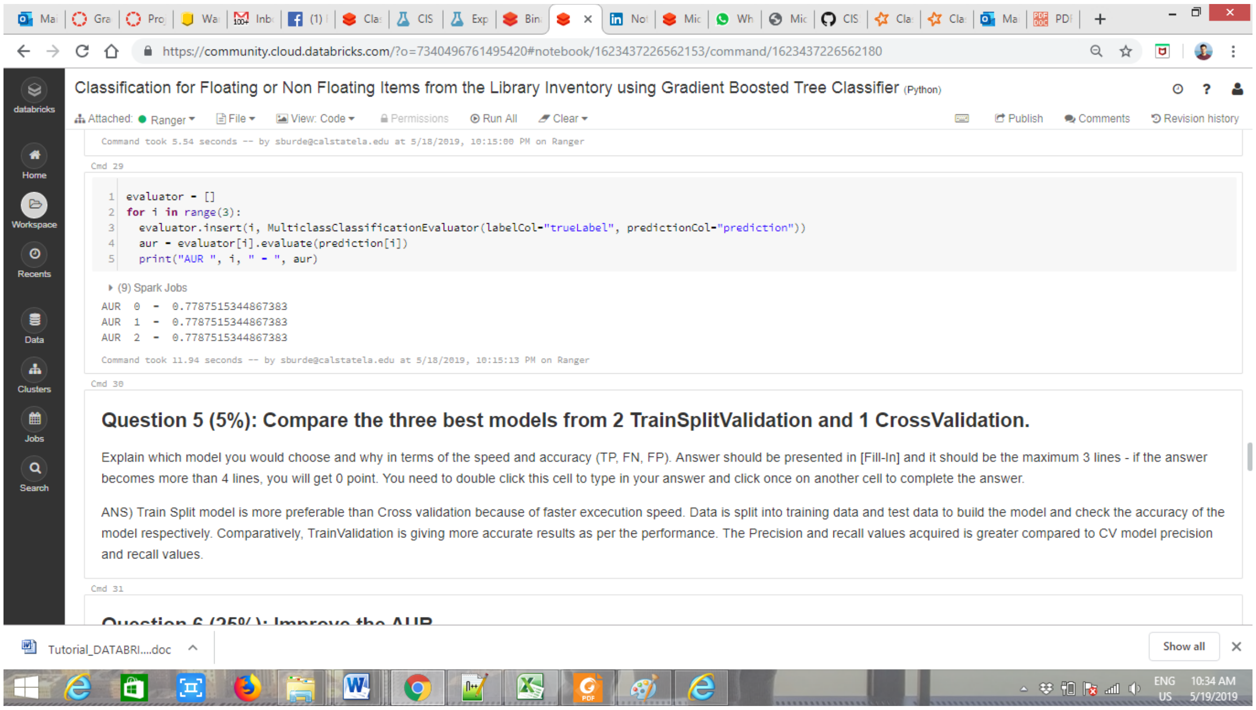
Cmd 27

Review the Area Under ROC

Another way to assess the performance of a classification model is to measure the area under a ROC curve for the model. the spark.ml library includes a **BinaryClassificationEvaluator** class that you can use to compute this.

Cmd 28

Step 8: Review the Area Under ROC


https://community.cloud.databricks.com/?o=7340496761495420#notebook/1623437226562153/command/1623437226562180
Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)
Attached: Ranger
File
View: Code
Permissions
Run All
Clear
Publish
Comments
Revision history
databricks
Home
Workspace
Recents
Data
Clusters
Jobs
Search
Tutorial_DATABRI....doc
Show all

```
Command took 5.54 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:15:00 PM on Ranger
```

Cmd 29

```
evaluator = []
for i in range(3):
  evaluator.insert(i, MulticlassClassificationEvaluator(labelCol="trueLabel", predictionCol="prediction"))
  aur = evaluator[i].evaluate(prediction[i])
  print("AUR ", i, " = ", aur)
```

(9) Spark Jobs

```
AUR  0  =  0.7787515344867383
AUR  1  =  0.7787515344867383
AUR  2  =  0.7787515344867383
Command took 11.94 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:15:13 PM on Ranger
```

Cmd 30

Question 5 (5%): Compare the three best models from 2 TrainSplitValidation and 1 CrossValidation.

Explain which model you would choose and why in terms of the speed and accuracy (TP, FN, FP). Answer should be presented in [Fill-In] and it should be the maximum 3 lines - if the answer becomes more than 4 lines, you will get 0 point. You need to double click this cell to type in your answer and click once on another cell to complete the answer.

ANS) Train Split model is more preferable than Cross validation because of faster excecution speed. Data is split into training data and test data to build the model and check the accuracy of the model respectively. Comparatively, TrainValidation is giving more accurate results as per the performance. The Precision and recall values acquired is greater compared to CV model precision and recall values.

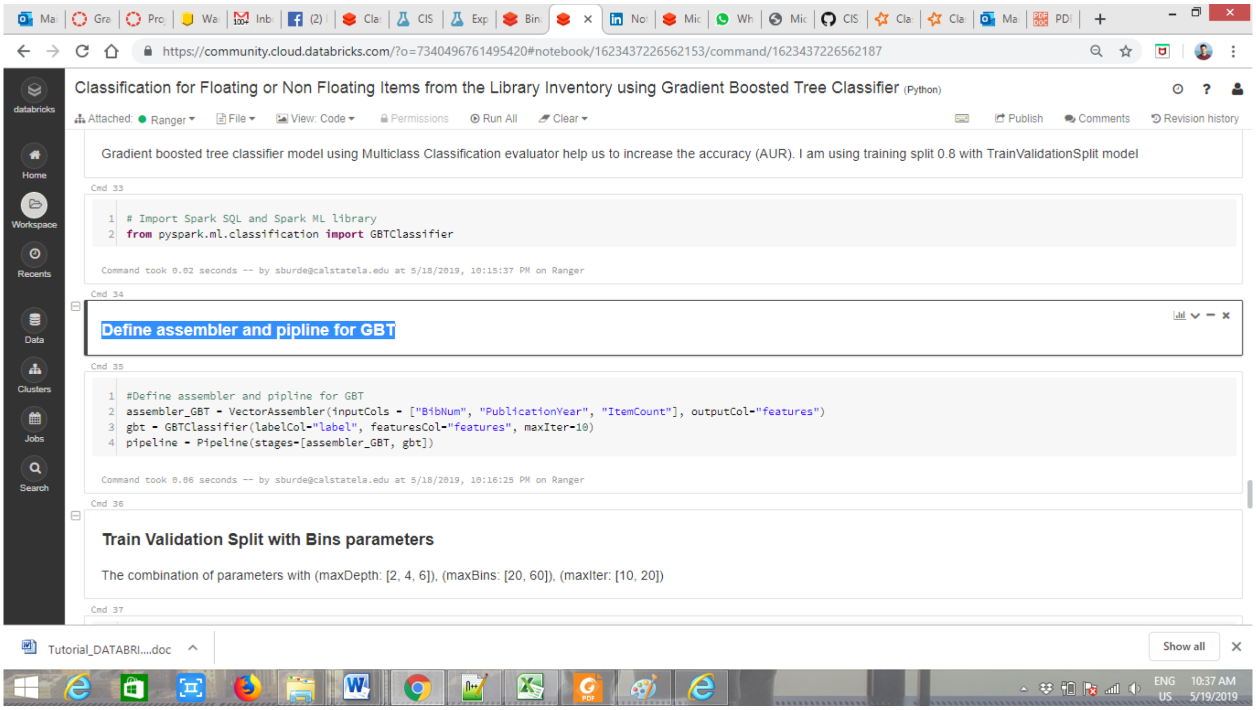
Cmd 31

Question 6 (25%): Improve the AUR

Step 9: Improve AUC: Gradient Boosted Tree Classifier

Gradient boosted tree classifier model using Multiclass Classification evaluator help us to increase the accuracy (AUR). I am using training split 0.8 with TrainValidationSplit model

Step 9.a Define assembler and pipeline for GBT



https://community.cloud.databricks.com/?o=7340496761495420#notebook/1623437226562153/command/1623437226562187

Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)

Gradient boosted tree classifier model using Multiclass Classification evaluator help us to increase the accuracy (AUR). I am using training split 0.8 with TrainValidationSplit model

Cmd 33

```
# Import Spark SQL and Spark ML library
from pyspark.ml.classification import GBTClassifier
```

Command took 0.02 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:15:37 PM on Ranger

Cmd 34

Define assembler and pipline for GBT

Cmd 35

```
#Define assembler and pipline for GBT
assembler_GBT = VectorAssembler(inputCols = ["BibNum", "PublicationYear", "ItemCount"], outputCol="features")
gbt = GBTClassifier(labelCol="label", featuresCol="features", maxIter=10)
pipeline = Pipeline(stages=[assembler_GBT, gbt])
```

Command took 0.06 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:16:25 PM on Ranger

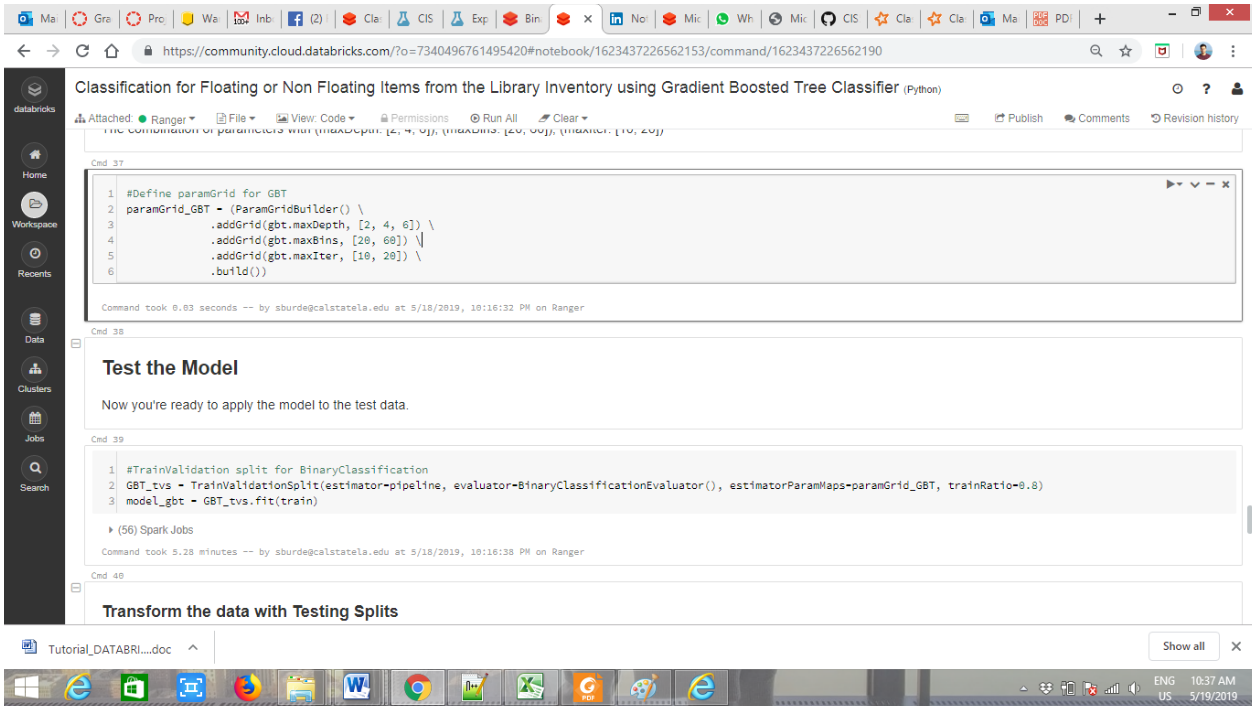
Cmd 36

Train Validation Split with Bins parameters

The combination of parameters with (maxDepth: [2, 4, 6]), (maxBins: [20, 60]), (maxIter: [10, 20])

Cmd 37

Step 9.b Train Validation Split with Bins parameters



https://community.cloud.databricks.com/?o=7340496761495420#notebook/1623437226562153/command/1623437226562190

Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)

The combination of parameters with (maxDepth: [2, 4, 6]), (maxBins: [20, 60]), (maxIter: [10, 20])

Cmd 37

```
#Define paramGrid for GBT
paramGrid_GBT = (ParamGridBuilder() \
            .addGrid(gbt.maxDepth, [2, 4, 6]) \
            .addGrid(gbt.maxBins, [20, 60]) \
            .addGrid(gbt.maxIter, [10, 20]) \
            .build())
```

```
Command took 0.03 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:16:32 PM on Ranger
```

Cmd 38

Test the Model

Now you're ready to apply the model to the test data.

Cmd 39

```
#TrainValidation split for BinaryClassification
GBT_tvs = TrainValidationSplit(estimator=pipeline, evaluator=BinaryClassificationEvaluator(), estimatorParamMaps=paramGrid_GBT, trainRatio=0.8)
model_gbt = GBT_tvs.fit(train)
```

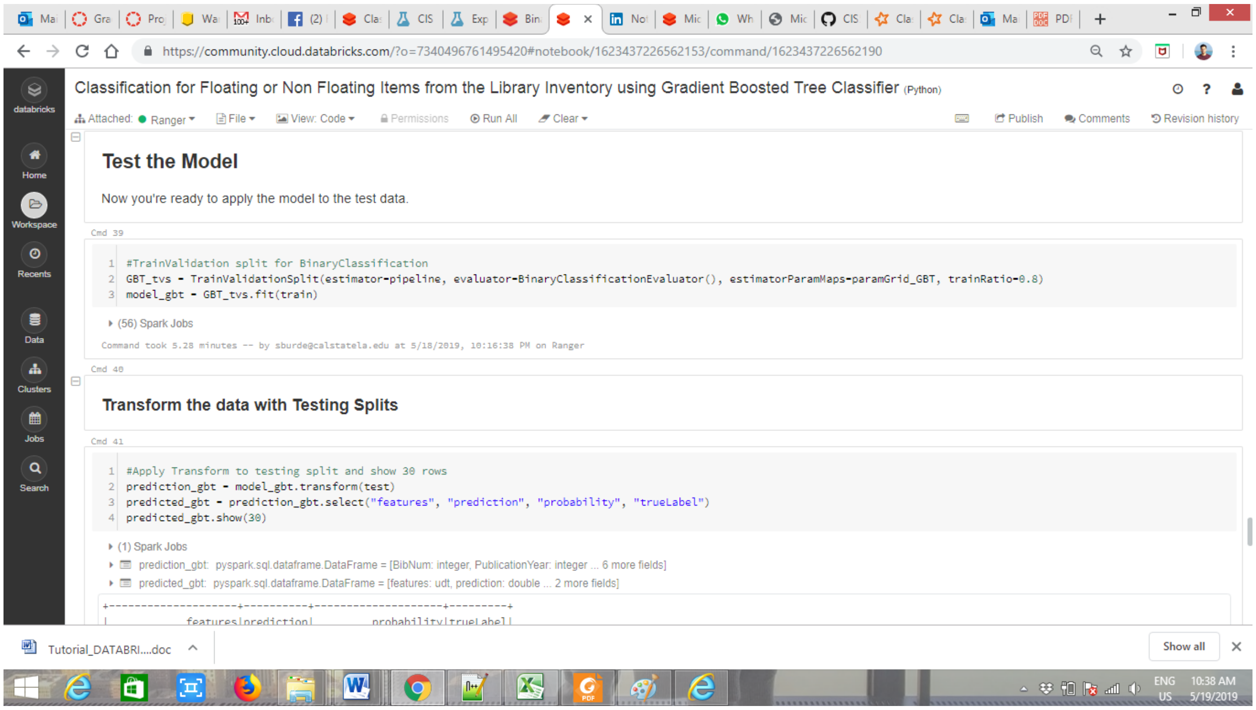
▸ (56) Spark Jobs

```
Command took 5.28 minutes -- by sburde@calstatela.edu at 5/18/2019, 10:16:38 PM on Ranger
```

Cmd 40

Transform the data with Testing Splits

Step 9.b Test the Model



Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)

Test the Model

Now you're ready to apply the model to the test data.

Cmd 39

```
#TrainValidation split for BinaryClassification
GBT_tvs = TrainValidationSplit(estimator=pipeline, evaluator=BinaryClassificationEvaluator(), estimatorParamMaps=paramGrid_GBT, trainRatio=0.8)
model_gbt = GBT_tvs.fit(train)
```

(56) Spark Jobs

Command took 5.28 minutes -- by sburde@calstatela.edu at 5/18/2019, 10:16:38 PM on Ranger

Cmd 40

Transform the data with Testing Splits

Cmd 41

```
#Apply Transform to testing split and show 30 rows
prediction_gbt = model_gbt.transform(test)
predicted_gbt = prediction_gbt.select("features", "prediction", "probability", "trueLabel")
predicted_gbt.show(30)
```

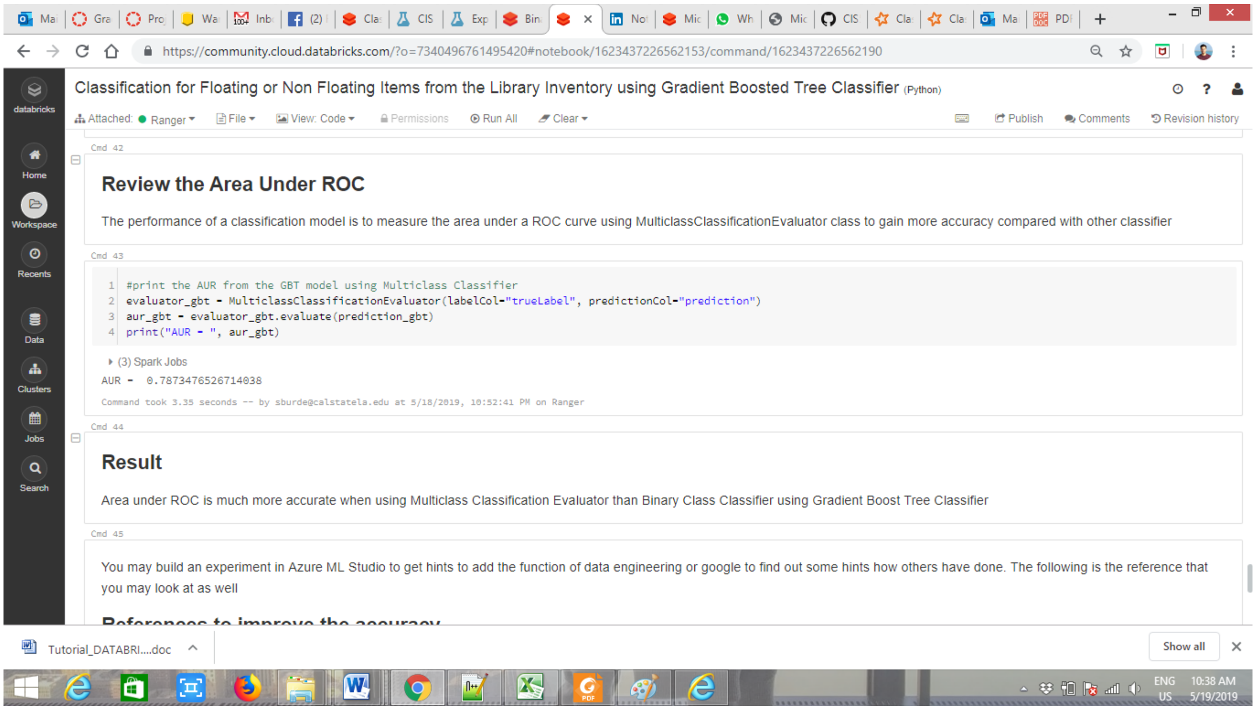
(1) Spark Jobs

prediction_gbt: pyspark.sql.dataframe.DataFrame = [BibNum: integer, PublicationYear: integer ... 6 more fields]

predicted_gbt: pyspark.sql.dataframe.DataFrame = [features: udt, prediction: double ... 2 more fields]

```
+--------------------+----------+--------------------+---------+
|            features|prediction|         probability|trueLabel|
```

Step 9. c Review the Area under ROC



Classification for Floating or Non Floating Items from the Library Inventory using Gradient Boosted Tree Classifier (Python)

Cmd 42

Review the Area Under ROC

The performance of a classification model is to measure the area under a ROC curve using MulticlassClassificationEvaluator class to gain more accuracy compared with other classifier

Cmd 43

```
#print the AUR from the GBT model using Multiclass Classifier
evaluator_gbt = MulticlassClassificationEvaluator(labelCol="trueLabel", predictionCol="prediction")
aur_gbt = evaluator_gbt.evaluate(prediction_gbt)
print("AUR = ", aur_gbt)
```

(3) Spark Jobs

```
AUR =  0.7873476526714038
```

Command took 3.35 seconds -- by sburde@calstatela.edu at 5/18/2019, 10:52:41 PM on Ranger

Cmd 44

Result

Area under ROC is much more accurate when using Multiclass Classification Evaluator than Binary Class Classifier using Gradient Boost Tree Classifier

Cmd 45

You may build an experiment in Azure ML Studio to get hints to add the function of data engineering or google to find out some hints how others have done. The following is the reference that you may look at as well

References to improve the accuracy

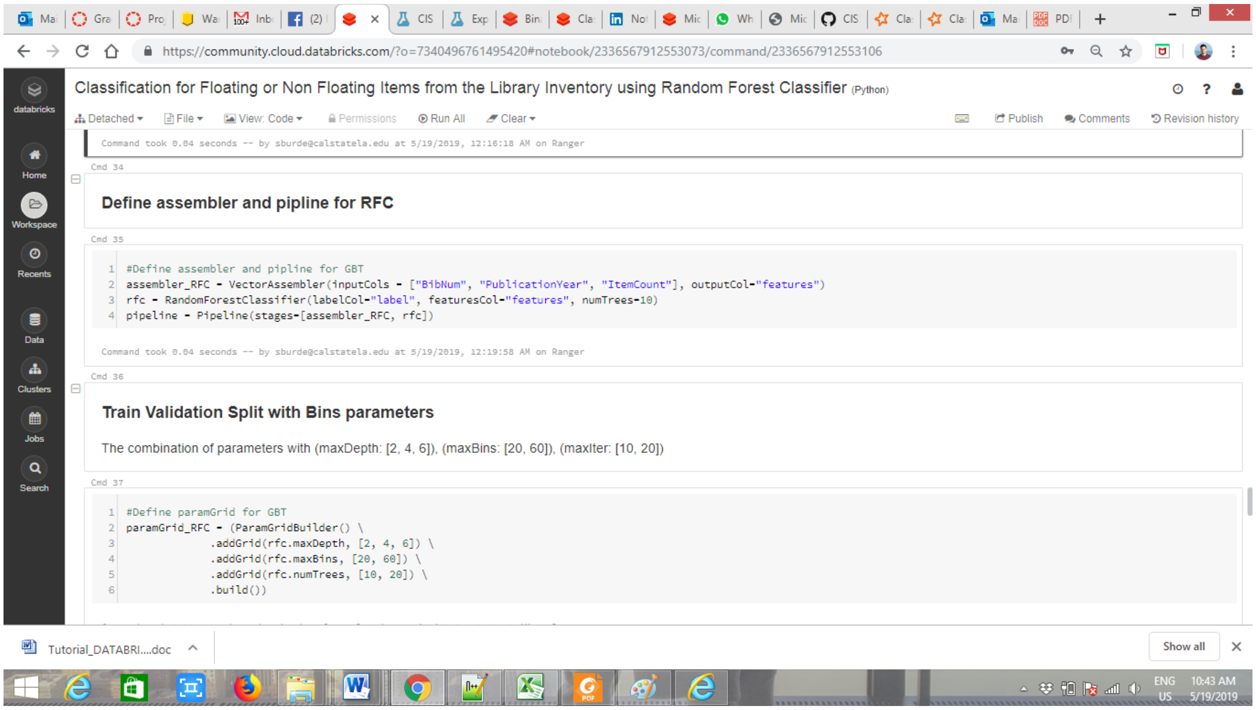
Result

Area under ROC is much more accurate when using Multiclass Classification Evaluator than Binary Class Classifier using Gradient Boost Tree Classifier is 0.787.

Step 10. Random Forest Classifier

Random Forest classifier model using Multiclass Classification evaluator help us to increase the accuracy (AUR). I am using training split 0.8 with TrainValidationSplit model

Define assembler and pipline for RFC



Classification for Floating or Non Floating Items from the Library Inventory using Random Forest Classifier (Python)

```
Command took 0.04 seconds -- by sburde@calstatela.edu at 5/19/2019, 12:16:18 AM on Ranger
```

Cmd 34

Define assembler and pipline for RFC

Cmd 35

```
#Define assembler and pipline for GBT
assembler_RFC = VectorAssembler(inputCols = ["BibNum", "PublicationYear", "ItemCount"], outputCol="features")
rfc = RandomForestClassifier(labelCol="label", featuresCol="features", numTrees=10)
pipeline = Pipeline(stages=[assembler_RFC, rfc])
```

```
Command took 0.04 seconds -- by sburde@calstatela.edu at 5/19/2019, 12:19:58 AM on Ranger
```

Cmd 36

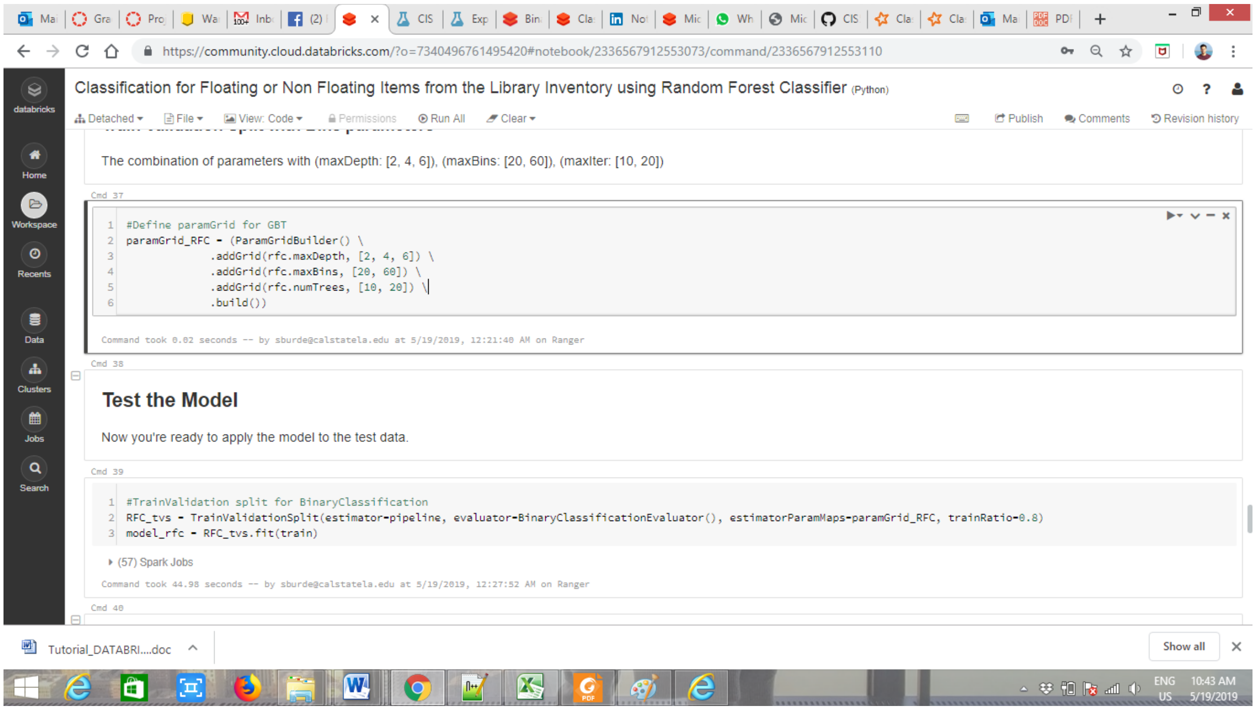
Train Validation Split with Bins parameters

The combination of parameters with (maxDepth: [2, 4, 6]), (maxBins: [20, 60]), (maxIter: [10, 20])

Cmd 37

```
#Define paramGrid for GBT
paramGrid_RFC = (ParamGridBuilder() \
            .addGrid(rfc.maxDepth, [2, 4, 6]) \
            .addGrid(rfc.maxBins, [20, 60]) \
            .addGrid(rfc.numTrees, [10, 20]) \
            .build())
```

Train Validation Split with Bins parameters



https://community.cloud.databricks.com/?o=7340496761495420#notebook/2336567912553073/command/2336567912553110
Classification for Floating or Non Floating Items from the Library Inventory using Random Forest Classifier (Python)
Detached
File
View: Code
Permissions
Run All
Clear
Publish
Comments
Revision history
databricks
Home
Workspace
Recents
Data
Clusters
Jobs
Search
Tutorial_DATABRI....doc
Show all
ENG
US
10:43 AM
5/19/2019

The combination of parameters with (maxDepth: [2, 4, 6]), (maxBins: [20, 60]), (maxIter: [10, 20])

Cmd 37

```
#Define paramGrid for GBT
paramGrid_RFC = (ParamGridBuilder() \
            .addGrid(rfc.maxDepth, [2, 4, 6]) \
            .addGrid(rfc.maxBins, [20, 60]) \
            .addGrid(rfc.numTrees, [10, 20]) \
            .build())
```

```
Command took 0.02 seconds -- by sburde@calstatela.edu at 5/19/2019, 12:21:40 AM on Ranger
```

Cmd 38

Test the Model

Now you're ready to apply the model to the test data.

Cmd 39

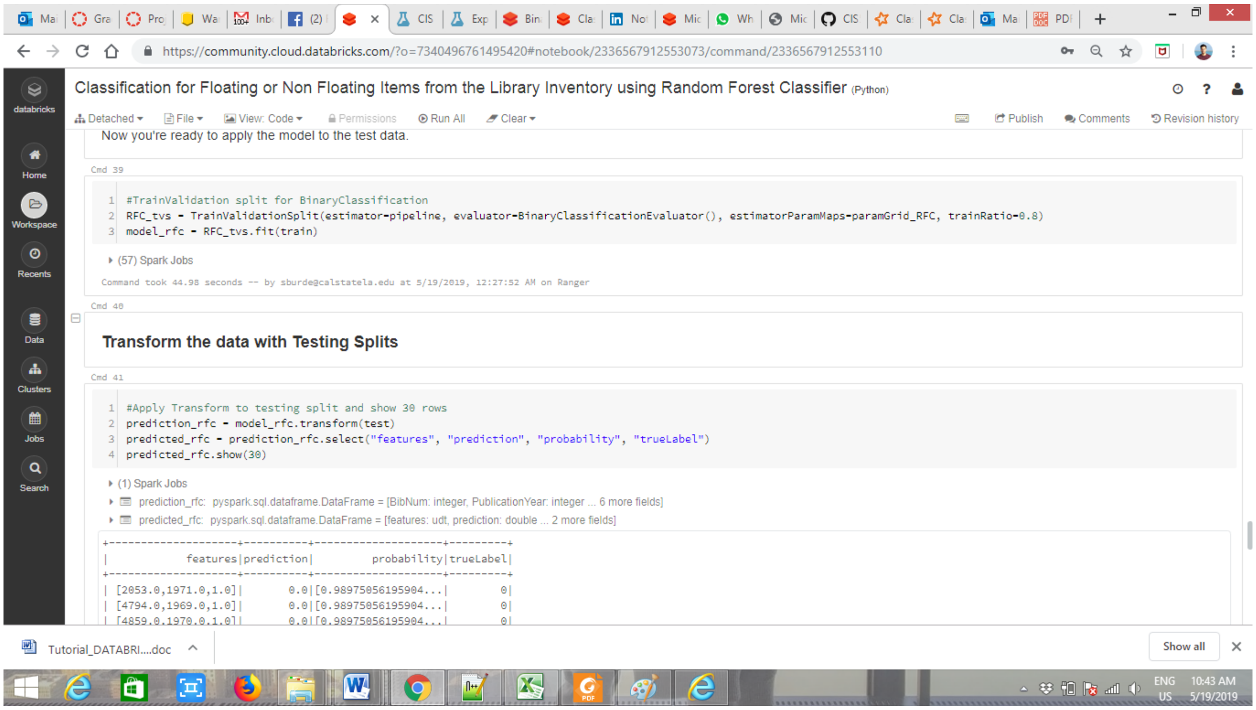
```
#TrainValidation split for BinaryClassification
RFC_tvs = TrainValidationSplit(estimator=pipeline, evaluator=BinaryClassificationEvaluator(), estimatorParamMaps=paramGrid_RFC, trainRatio=0.8)
model_rfc = RFC_tvs.fit(train)
```

▸ (57) Spark Jobs

```
Command took 44.98 seconds -- by sburde@calstatela.edu at 5/19/2019, 12:27:52 AM on Ranger
```

Cmd 40

Test the Model



https://community.cloud.databricks.com/?o=7340496761495420#notebook/2336567912553073/command/2336567912553110
Classification for Floating or Non Floating Items from the Library Inventory using Random Forest Classifier (Python)
Detached
File
View: Code
Permissions
Run All
Clear
Publish
Comments
Revision history
databricks
Home
Workspace
Recents
Data
Clusters
Jobs
Search
Tutorial_DATABRI....doc
Show all
ENG
US
10:43 AM
5/19/2019

Now you're ready to apply the model to the test data.

Cmd 39

```
#TrainValidation split for BinaryClassification
RFC_tvs = TrainValidationSplit(estimator=pipeline, evaluator=BinaryClassificationEvaluator(), estimatorParamMaps=paramGrid_RFC, trainRatio=0.8)
model_rfc = RFC_tvs.fit(train)
```

▸ (57) Spark Jobs

```
Command took 44.98 seconds -- by sburde@calstatela.edu at 5/19/2019, 12:27:52 AM on Ranger
```

Cmd 40

Transform the data with Testing Splits

Cmd 41

```
#Apply Transform to testing split and show 30 rows
prediction_rfc = model_rfc.transform(test)
predicted_rfc = prediction_rfc.select("features", "prediction", "probability", "trueLabel")
predicted_rfc.show(30)
```

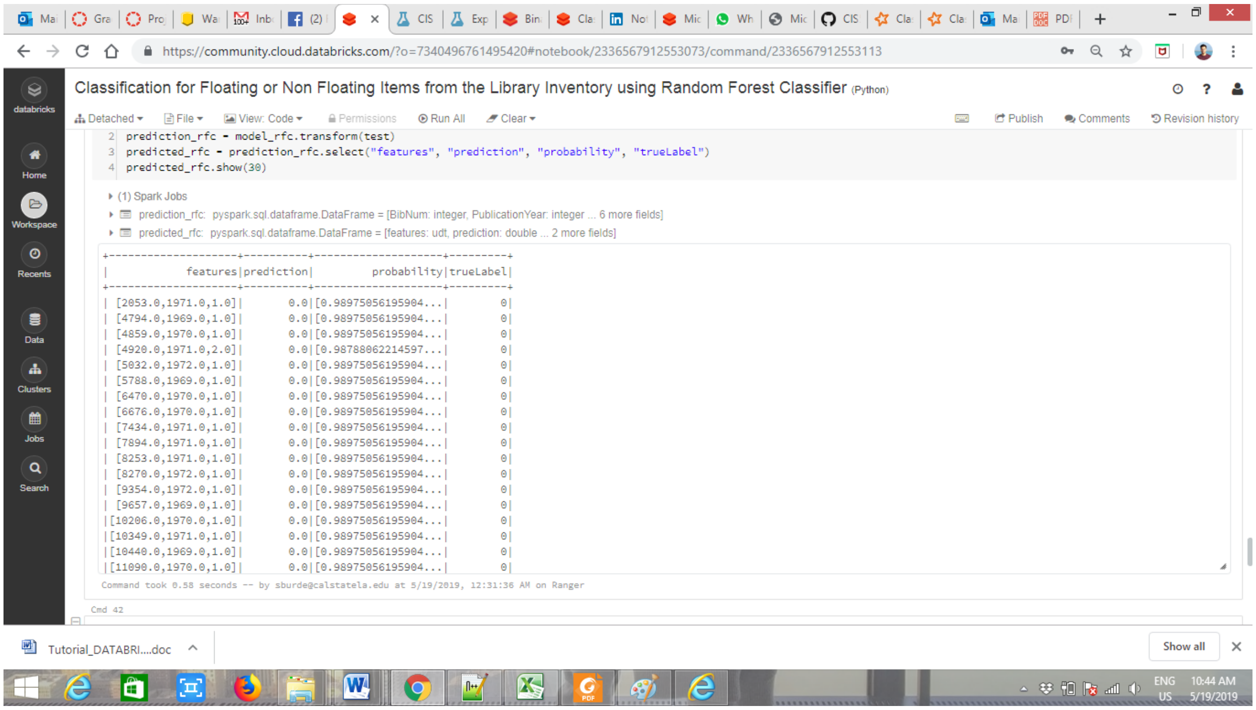
▸ (1) Spark Jobs

▸ prediction_rfc: pyspark.sql.dataframe.DataFrame = [BibNum: integer, PublicationYear: integer ... 6 more fields]

▸ predicted_rfc: pyspark.sql.dataframe.DataFrame = [features: udt, prediction: double ... 2 more fields]

```
+--------------------+----------+--------------------+---------+
|            features|prediction|         probability|trueLabel|
+--------------------+----------+--------------------+---------+
| [2053.0,1971.0,1.0]|       0.0|[0.98975056195904...|        0|
| [4794.0,1969.0,1.0]|       0.0|[0.98975056195904...|        0|
| [4859.0,1970.0,1.0]|       0.0|[0.98975056195904...|        0|
```

Transform the data with Testing Splits



Classification for Floating or Non Floating Items from the Library Inventory using Random Forest Classifier (Python)

```
prediction_rfc = model_rfc.transform(test)
predicted_rfc = prediction_rfc.select("features", "prediction", "probability", "trueLabel")
predicted_rfc.show(30)
```

(1) Spark Jobs

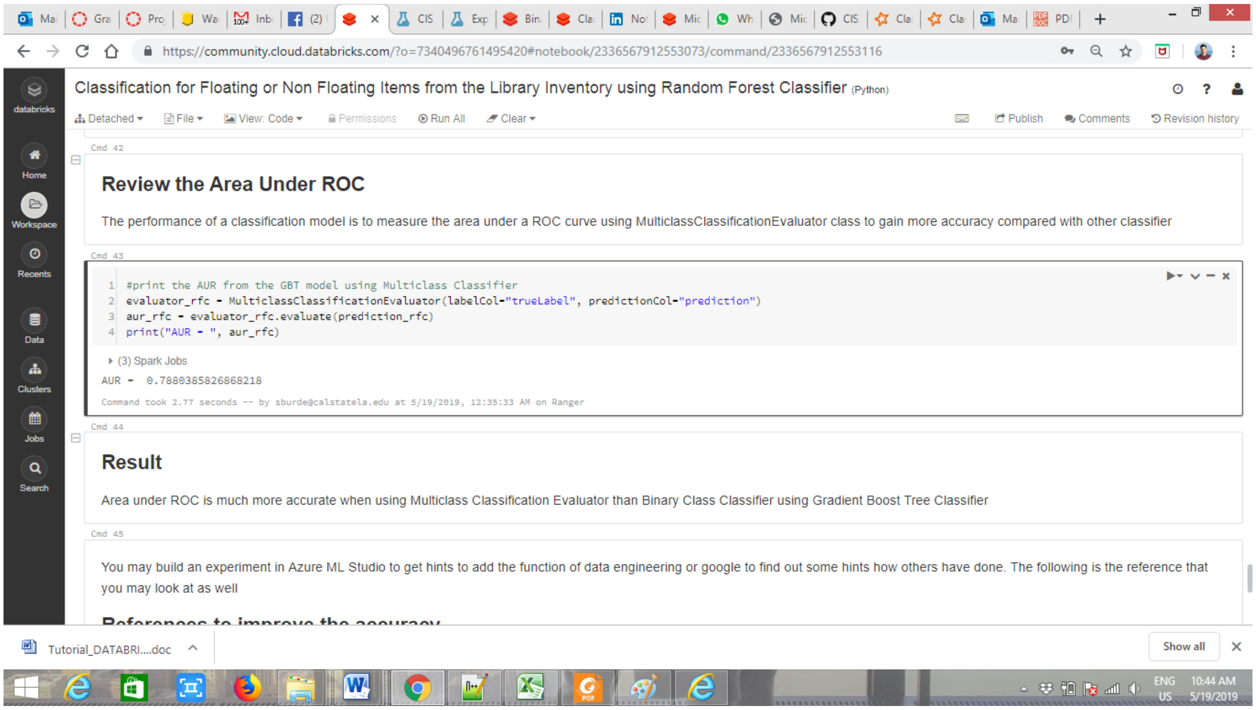
prediction_rfc: pyspark.sql.dataframe.DataFrame = [BibNum: integer, PublicationYear: integer ... 6 more fields]

predicted_rfc: pyspark.sql.dataframe.DataFrame = [features: udt, prediction: double ... 2 more fields]

```
+--------------------+----------+--------------------+---------+
|            features|prediction|         probability|trueLabel|
+--------------------+----------+--------------------+---------+
| [2053.0,1971.0,1.0]|       0.0|[0.98975056195904...|        0|
| [4794.0,1969.0,1.0]|       0.0|[0.98975056195904...|        0|
| [4859.0,1970.0,1.0]|       0.0|[0.98975056195904...|        0|
| [4920.0,1971.0,2.0]|       0.0|[0.98788062214597...|        0|
| [5032.0,1972.0,1.0]|       0.0|[0.98975056195904...|        0|
| [5788.0,1969.0,1.0]|       0.0|[0.98975056195904...|        0|
| [6470.0,1970.0,1.0]|       0.0|[0.98975056195904...|        0|
| [6676.0,1970.0,1.0]|       0.0|[0.98975056195904...|        0|
| [7434.0,1971.0,1.0]|       0.0|[0.98975056195904...|        0|
| [7894.0,1971.0,1.0]|       0.0|[0.98975056195904...|        0|
| [8253.0,1971.0,1.0]|       0.0|[0.98975056195904...|        0|
| [8270.0,1972.0,1.0]|       0.0|[0.98975056195904...|        0|
| [9354.0,1972.0,1.0]|       0.0|[0.98975056195904...|        0|
| [9657.0,1969.0,1.0]|       0.0|[0.98975056195904...|        0|
|[10206.0,1970.0,1.0]|       0.0|[0.98975056195904...|        0|
|[10349.0,1971.0,1.0]|       0.0|[0.98975056195904...|        0|
|[10440.0,1969.0,1.0]|       0.0|[0.98975056195904...|        0|
|[11090.0,1970.0,1.0]|       0.0|[0.98975056195904...|        0|
```

Command took 0.58 seconds -- by sburde@calstatela.edu at 5/19/2019, 12:31:36 AM on Ranger

Review the Area Under ROC by Random Forest Classifier



Classification for Floating or Non Floating Items from the Library Inventory using Random Forest Classifier (Python)

Cmd 42

Review the Area Under ROC

The performance of a classification model is to measure the area under a ROC curve using MulticlassClassificationEvaluator class to gain more accuracy compared with other classifier

Cmd 43

```
#print the AUR from the GBT model using Multiclass Classifier
evaluator_rfc = MulticlassClassificationEvaluator(labelCol="trueLabel", predictionCol="prediction")
aur_rfc = evaluator_rfc.evaluate(prediction_rfc)
print("AUR = ", aur_rfc)
```

(3) Spark Jobs

```
AUR =  0.7880385826868218
Command took 2.77 seconds -- by sburde@calstatela.edu at 5/19/2019, 12:35:33 AM on Ranger
```

Cmd 44

Result

Area under ROC is much more accurate when using Multiclass Classification Evaluator than Binary Class Classifier using Gradient Boost Tree Classifier

Cmd 45

You may build an experiment in Azure ML Studio to get hints to add the function of data engineering or google to find out some hints how others have done. The following is the reference that you may look at as well

References to improve the accuracy

Result

Area under ROC is much more accurate when using Multiclass Classification Evaluator than Binary Class Classifier using Random Forest Classifier is 0.788.

References to improve the accuracy

1. Extracting, transforming and selecting features, https://spark.apache.org/docs/latest/ml-features.html
2. Basic data preparation in Pyspark — Capping, Normalizing and Scaling, http://bit.ly/2Ihs6Wa
3. Machine Learning with PySpark and MLlib — Solving a Binary Classification Problem, http://bit.ly/2Zb20tg