Professor Pete Alonzi

DS 6003

Introduction to Spark

Jan 28 2019

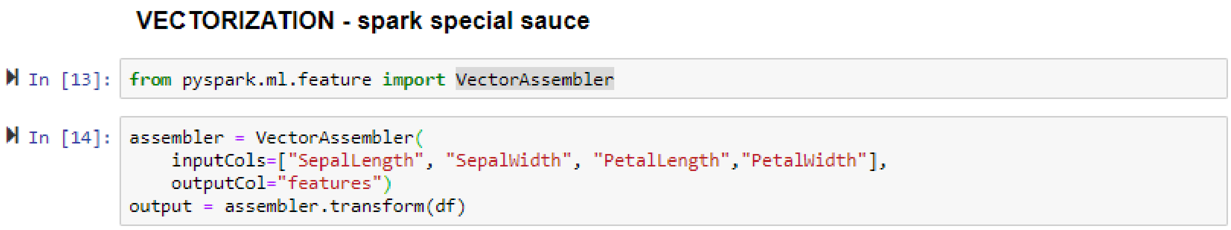
Luke Kang(sk5be)

Motivation

The motivation of this project is to perform machine learning analysis on the Iris dataset in Spark environment. The dataset contains the following columns: SepalLength, SepalWidth, PetalLength, PetalWidth and Species. Species is a response value and it originally contains 3 classes, which are setosa, versicolor and virginicia. In order to perform binary classification on the data, we would remove virginicia-related data out of the dataset. Also, we converted setosa to 1, as well as versicolor to 2. In specific, logistic regression and decision trees methods were performed. For the methods, MLlib in Spark was used. Along with it, Amazon S3 was used for storage service since S3 had relatively easy to use management features and it fulfilled the analysis requirements for this project. Also, Amazon Sagemaker was used since it is designed to allow machine learn models to run effectively. As we would like to build a full pipeline in Spark, first, a Spark context was created and it was wired up with S3. After the dataset was loaded from S3 to Sagemaker, it also was saved as parquet. The analysis was performed by using MLlib including VectorAssembler, LogisticRegression, BinaryClassificationEvaluator and DecisionTreeClassifier.

Code snippet and explanation

Before models actually were built, the dataset was vectorized as follows in order to accelerate computation speed.

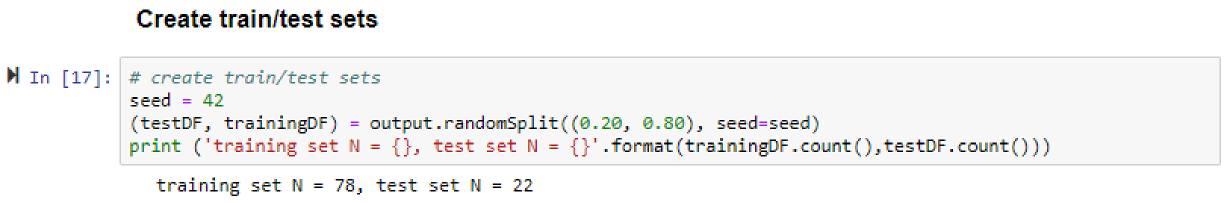


VECTORIZATION - spark special sauce

```
In [13]: from pyspark.ml.feature import VectorAssembler
```

```
In [14]: assembler = VectorAssembler(
             inputCols=["SepalLength", "SepalWidth", "PetalLength","PetalWidth"],
             outputCol="features")
         output = assembler.transform(df)
```

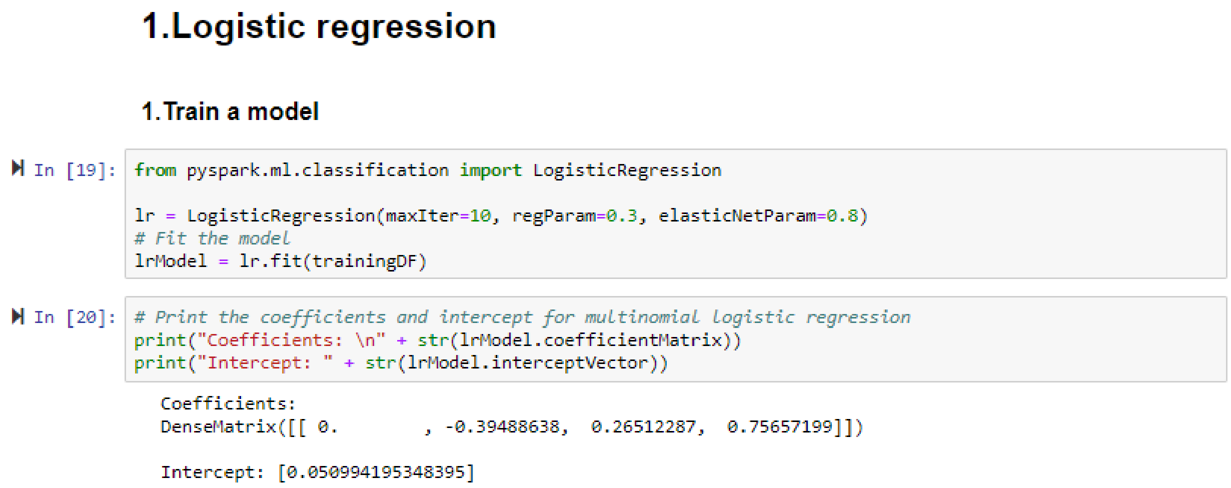
Moreover, the dataset was split into train/test sets as follows.



Create train/test sets

```
In [17]: # create train/test sets
         seed = 42
         (testDF, trainingDF) = output.randomSplit((0.20, 0.80), seed=seed)
         print ('training set N = {}, test set N = {}'.format(trainingDF.count(),testDF.count()))

          training set N = 78, test set N = 22
```

Logistic regression model was built on the training set first and its coefficients and intercept were obtained.



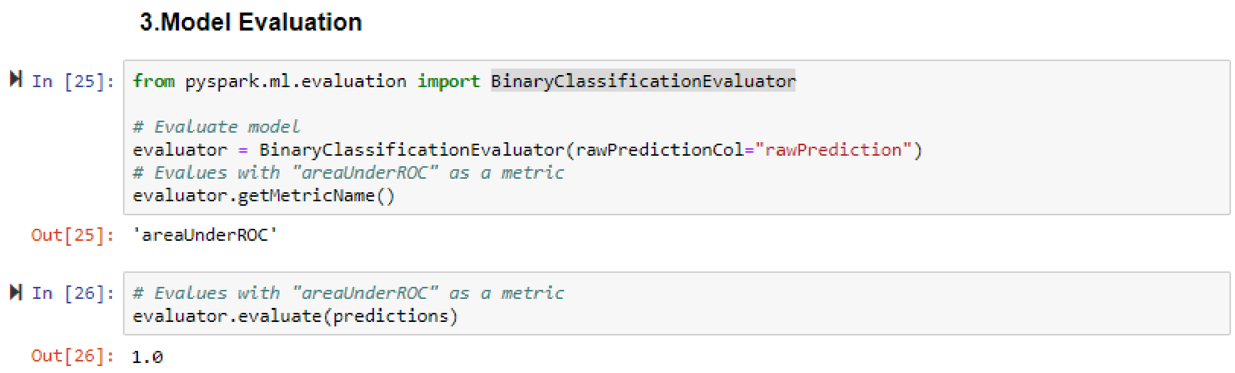
1.Logistic regression

1.Train a model

```
In [19]: from pyspark.ml.classification import LogisticRegression

         lr = LogisticRegression(maxIter=10, regParam=0.3, elasticNetParam=0.8)
         # Fit the model
         lrModel = lr.fit(trainingDF)
```

```
In [20]: # Print the coefficients and intercept for multinomial logistic regression
         print("Coefficients: \n" + str(lrModel.coefficientMatrix))
         print("Intercept: " + str(lrModel.interceptVector))

          Coefficients: 
          DenseMatrix([[ 0.        , -0.39488638,  0.26512287,  0.75657199]])

          Intercept: [0.050994195348395]
```

Later the model used for prediction and it was evaluated by ROC metric. The model returns 1.0, which means it returns 100% accuracy.



3.Model Evaluation

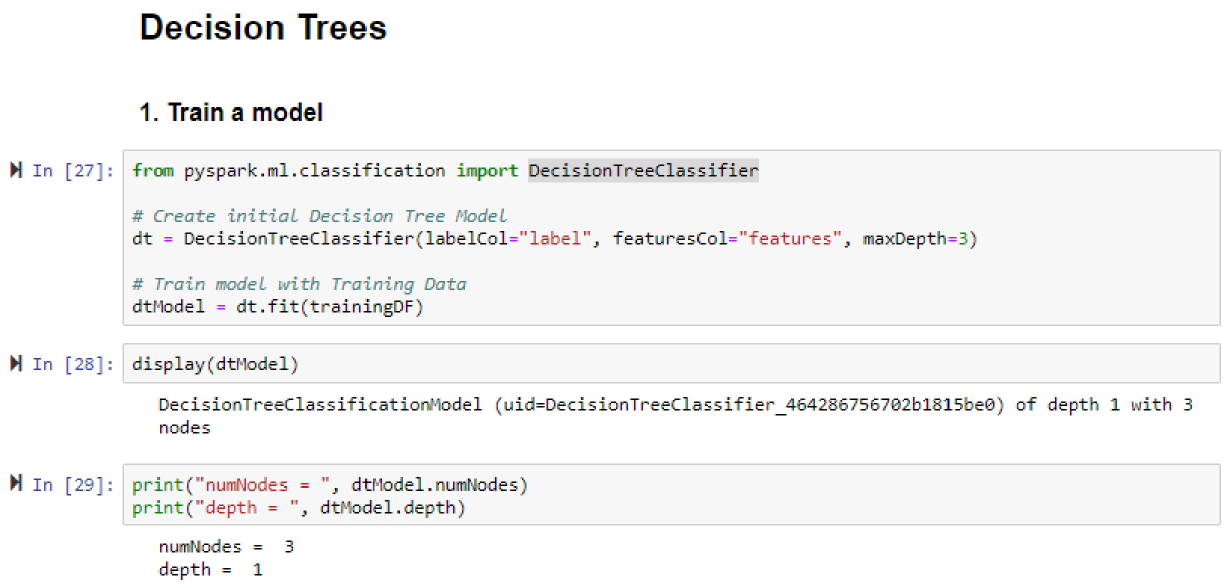
In [25]:
```
from pyspark.ml.evaluation import BinaryClassificationEvaluator

# Evaluate model
evaluator = BinaryClassificationEvaluator(rawPredictionCol="rawPrediction")
# Evalues with "areaUnderROC" as a metric
evaluator.getMetricName()
```

Out[25]:
```
'areaUnderROC'
```

In [26]:
```
# Evalues with "areaUnderROC" as a metric
evaluator.evaluate(predictions)
```

Out[26]:
```
1.0
```

As the 2nd model, decision trees model was built with 3 nodes.



Decision Trees

1. Train a model

In [27]:
```
from pyspark.ml.classification import DecisionTreeClassifier

# Create initial Decision Tree Model
dt = DecisionTreeClassifier(labelCol="label", featuresCol="features", maxDepth=3)

# Train model with Training Data
dtModel = dt.fit(trainingDF)
```

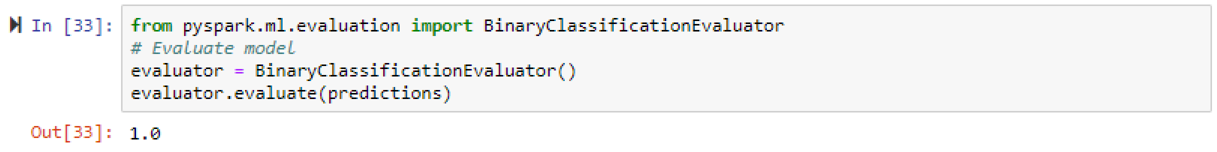
In [28]:
```
display(dtModel)
```

```
DecisionTreeClassificationModel (uid=DecisionTreeClassifier_464286756702b1815be0) of depth 1 with 3
nodes
```

In [29]:
```
print("numNodes = ", dtModel.numNodes)
print("depth = ", dtModel.depth)
```

```
numNodes =  3
depth =  1
```

Again, the model was evaluated as follows and it also returns 100% accuracy.

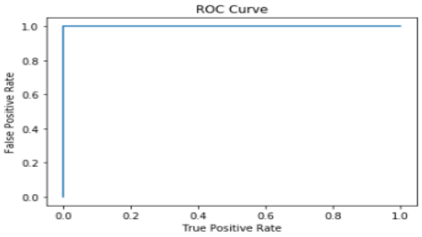


In [33]:
```
from pyspark.ml.evaluation import BinaryClassificationEvaluator
# Evaluate model
evaluator = BinaryClassificationEvaluator()
evaluator.evaluate(predictions)
```

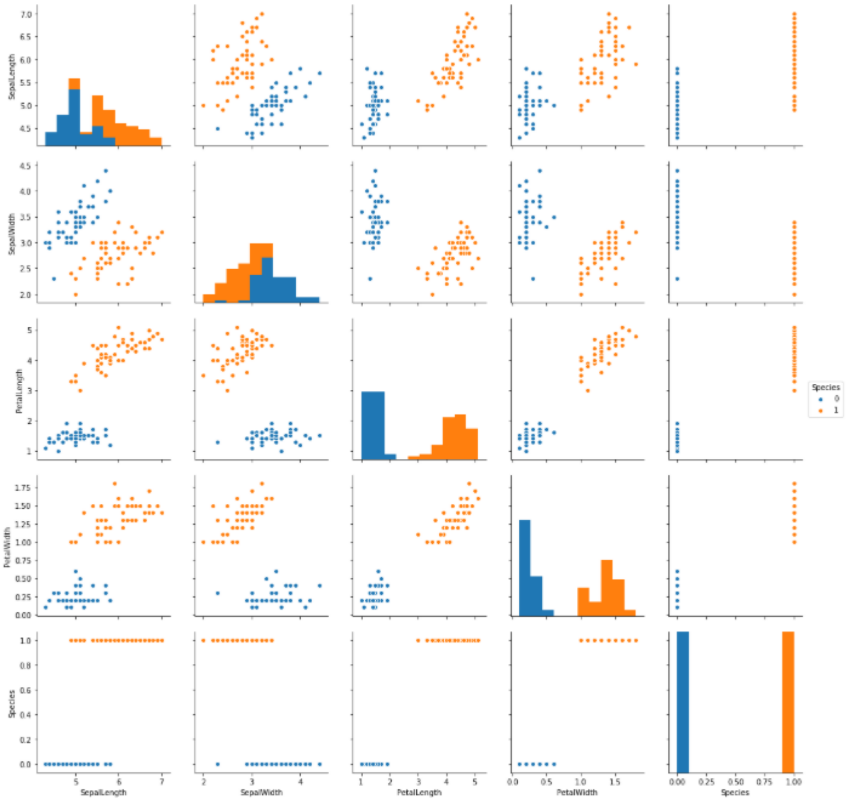
Out[33]:
```
1.0
```

Visualization

ROC curve was built for the logistic model.



By visualizing the data, we could see the dataset was originally well-seperated. Accordingly, the logistic regression model and decision trees model could easily classify the dataset into two classes


SepalLength
SepalWidth
PetalLength
PetalWidth
Species
Species