PysparkDecisionTree_Classifier.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

RAM Disk Colab AI

[1] `!pip install pyspark`

```
Collecting pyspark
  Downloading pyspark-3.5.1.tar.gz (317.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 317.0/317.0 MB 3.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.5.1-py2.py3-none-any.whl size=317488493 sha256=f957e2e088194f2caf153154c275a577cd876de187f462ff48cc251375d33f6f
  Stored in directory: /root/.cache/pip/wheels/80/1d/60/2c256ed38dddce2fdd93be545214a63e02fbd8d74fb0b7f3a6
Successfully built pyspark
Installing collected packages: pyspark
Successfully installed pyspark-3.5.1
```

Double-click (or enter) to edit

```
import pandas as pd
import matplotlib.pyplot as plt
```

[3]
```
import pyspark as ps
spark =ps.sql.SparkSession.builder.master("local").appName("demo3").getOrCreate()
```

Double-click (or enter) to edit

[4]
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ] Start coding or generate with AI.

[5]
```
dataset_df =spark.read.csv("/content/drive/My Drive/Colab Notebooks/Dataset/Iris.csv",header=True,infer
```

[6]
```
dataset_df.show()
```

```
+---+-------------+------------+-------------+------------+-----------+
| Id|SepalLengthCm|SepalWidthCm|PetalLengthCm|PetalWidthCm|    Species|
+---+-------------+------------+-------------+------------+-----------+
|  1|          5.1|         3.5|          1.4|         0.2|Iris-setosa|
|  2|          4.9|         3.0|          1.4|         0.2|Iris-setosa|
|  3|          4.7|         3.2|          1.3|         0.2|Iris-setosa|
|  4|          4.6|         3.1|          1.5|         0.2|Iris-setosa|
|  5|          5.0|         3.6|          1.4|         0.2|Iris-setosa|
|  6|          5.4|         3.9|          1.7|         0.4|Iris-setosa|
|  7|          4.6|         3.4|          1.4|         0.3|Iris-setosa|
|  8|          5.0|         3.4|          1.5|         0.2|Iris-setosa|
|  9|          4.4|         2.9|          1.4|         0.2|Iris-setosa|
| 10|          4.9|         3.1|          1.5|         0.1|Iris-setosa|
| 11|          5.4|         3.7|          1.5|         0.2|Iris-setosa|
| 12|          4.8|         3.4|          1.6|         0.2|Iris-setosa|
| 13|          4.8|         3.0|          1.4|         0.1|Iris-setosa|
| 14|          4.3|         3.0|          1.1|         0.1|Iris-setosa|
| 15|          5.8|         4.0|          1.2|         0.2|Iris-setosa|
| 16|          5.7|         4.4|          1.5|         0.4|Iris-setosa|
| 17|          5.4|         3.9|          1.3|         0.4|Iris-setosa|
| 18|          5.1|         3.5|          1.4|         0.3|Iris-setosa|
| 19|          5.7|         3.8|          1.7|         0.3|Iris-setosa|
| 20|          5.1|         3.8|          1.5|         0.3|Iris-setosa|
+---+-------------+------------+-------------+------------+-----------+
only showing top 20 rows
```

[7]
```
dataset_df.printSchema()
```

```
root
 |-- Id: integer (nullable = true)
 |-- SepalLengthCm: double (nullable = true)
 |-- SepalWidthCm: double (nullable = true)
 |-- PetalLengthCm: double (nullable = true)
 |-- PetalWidthCm: double (nullable = true)
 |-- Species: string (nullable = true)
```

Input variables: SepalLengthCm,SepalWidthCm,PetalLengthCm and PetalWidthCm

OutPut Variables: Species

[10]
```
# import pandas as pd
# pd.DataFrame(dataset_df.take(5), columns=dataset_df.columns).transpose()
```

	0	1	2	3	4
Id	1	2	3	4	5
SepalLengthCm	5.1	4.9	4.7	4.6	5.0
SepalWidthCm	3.5	3.0	3.2	3.1	3.6
PetalLengthCm	1.4	1.4	1.3	1.5	1.4
PetalWidthCm	0.2	0.2	0.2	0.2	0.2
Species	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa

```
dataset_df.groupby('Species').count().toPandas()
```

	Species	count
0	Iris-virginica	50
1	Iris-setosa	50
2	Iris-versicolor	50

Statistics Summary

```
numeric_features = [t[0] for t in dataset_df.dtypes if t[1] == 'int']
dataset_df.select(numeric_features).describe().toPandas().transpose()
```

	0	1	2	3	4
summary	count	mean	stddev	min	max
Id	150	75.5	43.445367992456916	1	150

Double-click (or enter) to edit

```
dataset_df.columns
```

```
['Id',
 'SepalLengthCm',
 'SepalWidthCm',
 'PetalLengthCm',
 'PetalWidthCm',
 'Species']
```

The pyspark.mllib.linalg module focused on linear algebra operations.

```
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
```

```
assemblerInputs =['SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm']
vector_assembler= VectorAssembler(inputCols=assemblerInputs,outputCol="fetaures")
assembler_temp =vector_assembler.transform(dataset_df)
```

```
assembler_temp.show(5)
```

```
+---+-------------+------------+-------------+------------+-----------+----------------+
| Id|SepalLengthCm|SepalWidthCm|PetalLengthCm|PetalWidthCm|    Species|        fetaures|
+---+-------------+------------+-------------+------------+-----------+----------------+
|  1|          5.1|         3.5|          1.4|         0.2|Iris-setosa|[5.1,3.5,1.4,0.2]|
|  2|          4.9|         3.0|          1.4|         0.2|Iris-setosa|[4.9,3.0,1.4,0.2]|
|  3|          4.7|         3.2|          1.3|         0.2|Iris-setosa|[4.7,3.2,1.3,0.2]|
|  4|          4.6|         3.1|          1.5|         0.2|Iris-setosa|[4.6,3.1,1.5,0.2]|
|  5|          5.0|         3.6|          1.4|         0.2|Iris-setosa|[5.0,3.6,1.4,0.2]|
+---+-------------+------------+-------------+------------+-----------+----------------+
only showing top 5 rows
```

```
assembler =assembler_temp.drop('SepalLengthCm','SepalWidthCm','PetalLengthCm','PetalWidthCm')
assembler.show(5)
```

```
+---+-----------+----------------+
| Id|    Species|        fetaures|
+---+-----------+----------------+
|  1|Iris-setosa|[5.1,3.5,1.4,0.2]|
|  2|Iris-setosa|[4.9,3.0,1.4,0.2]|
|  3|Iris-setosa|[4.7,3.2,1.3,0.2]|
|  4|Iris-setosa|[4.6,3.1,1.5,0.2]|
|  5|Iris-setosa|[5.0,3.6,1.4,0.2]|
+---+-----------+----------------+
only showing top 5 rows
```

Correlations between independent variables

```
numeric_features = [t[0] for t in dataset_df.dtypes if t[1] == 'int']
dataset_df.select(numeric_features).describe().toPandas().transpose()
```

	0	1	2	3	4
summary	count	mean	stddev	min	max
Id	150	75.5	43.445367992456916	1	150

Data preparation and feature engineering

In this part, we will remove unnecessary columns and fill the missing values. Finally, selecting features for machine learning models. These features will be divided into two parts train and test.

Lets starting mission 🐵

Missing Data Handling:

[23]
```
from pyspark.sql.functions import isnull, when, count, col

dataset_df.select([count(when(isnull(c), c)).alias(c) for c in dataset_df.columns]).show()
```

```
+---+-------------+------------+-------------+------------+-------+
| Id|SepalLengthCm|SepalWidthCm|PetalLengthCm|PetalWidthCm|Species|
+---+-------------+------------+-------------+------------+-------+
|  0|            0|           0|            0|           0|      0|
+---+-------------+------------+-------------+------------+-------+
```

Unnecessary columns dropping

[ ]
```
#Not used in this Implemenation but explained how to dropped
dataset_df = dataset_df.drop('SepalLengthCm')
dataset_df.show()
```

Features Convert into Vector

VectorAssembler — a feature transformer that merges multiple columns into a vector column.

[24]
```
# Assemble all the features with VectorAssembler
required_features = ['SepalLengthCm',
                     'SepalWidthCm',
                     'PetalLengthCm',
                     'PetalWidthCm'
                    ]
from pyspark.ml.feature import VectorAssembler
assembler = VectorAssembler(inputCols=required_features, outputCol='features')
transformed_data = assembler.transform(dataset_df )
transformed_data.show()
```

```
+---+-------------+------------+-------------+------------+-----------+-----------------+
| Id|SepalLengthCm|SepalWidthCm|PetalLengthCm|PetalWidthCm|    Species|         features|
+---+-------------+------------+-------------+------------+-----------+-----------------+
|  1|          5.1|         3.5|          1.4|         0.2|Iris-setosa|[5.1,3.5,1.4,0.2]|
|  2|          4.9|         3.0|          1.4|         0.2|Iris-setosa|[4.9,3.0,1.4,0.2]|
|  3|          4.7|         3.2|          1.3|         0.2|Iris-setosa|[4.7,3.2,1.3,0.2]|
|  4|          4.6|         3.1|          1.5|         0.2|Iris-setosa|[4.6,3.1,1.5,0.2]|
|  5|          5.0|         3.6|          1.4|         0.2|Iris-setosa|[5.0,3.6,1.4,0.2]|
|  6|          5.4|         3.9|          1.7|         0.4|Iris-setosa|[5.4,3.9,1.7,0.4]|
|  7|          4.6|         3.4|          1.4|         0.3|Iris-setosa|[4.6,3.4,1.4,0.3]|
|  8|          5.0|         3.4|          1.5|         0.2|Iris-setosa|[5.0,3.4,1.5,0.2]|
|  9|          4.4|         2.9|          1.4|         0.2|Iris-setosa|[4.4,2.9,1.4,0.2]|
| 10|          4.9|         3.1|          1.5|         0.1|Iris-setosa|[4.9,3.1,1.5,0.1]|
| 11|          5.4|         3.7|          1.5|         0.2|Iris-setosa|[5.4,3.7,1.5,0.2]|
| 12|          4.8|         3.4|          1.6|         0.2|Iris-setosa|[4.8,3.4,1.6,0.2]|
| 13|          4.8|         3.0|          1.4|         0.1|Iris-setosa|[4.8,3.0,1.4,0.1]|
| 14|          4.3|         3.0|          1.1|         0.1|Iris-setosa|[4.3,3.0,1.1,0.1]|
| 15|          5.8|         4.0|          1.2|         0.2|Iris-setosa|[5.8,4.0,1.2,0.2]|
| 16|          5.7|         4.4|          1.5|         0.4|Iris-setosa|[5.7,4.4,1.5,0.4]|
| 17|          5.4|         3.9|          1.3|         0.4|Iris-setosa|[5.4,3.9,1.3,0.4]|
| 18|          5.1|         3.5|          1.4|         0.3|Iris-setosa|[5.1,3.5,1.4,0.3]|
| 19|          5.7|         3.8|          1.7|         0.3|Iris-setosa|[5.7,3.8,1.7,0.3]|
| 20|          5.1|         3.8|          1.5|         0.3|Iris-setosa|[5.1,3.8,1.5,0.3]|
+---+-------------+------------+-------------+------------+-----------+-----------------+
only showing top 20 rows
```

Train and Test Split

Randomly split data into train and test sets, and set seed for reproducibility.

Double-click (or enter) to edit

[25]
```
# Split the data
(training_data, test_data) = transformed_data.randomSplit([0.8,0.2], seed =2020)
print("Training Dataset Count: " + str(training_data.count()))
print("Test Dataset Count: " + str(test_data.count()))
```

```
Training Dataset Count: 125
Test Dataset Count: 25
```

Random Forest Classifier

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets

the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

```
[26] from pyspark.ml.classification import RandomForestClassifier

rf = RandomForestClassifier(labelCol='Outcome',
                            featuresCol='features',
                            maxDepth=5)
model = rf.fit(training_data)
rf_predictions = model.transform(test_data)
```

Decision Tree Classifier

Decision trees are widely used since they are easy to interpret, handle categorical features, extend to the multiclass classification setting, do not require feature scaling, and are able to capture non-linearities and feature interactions.

Logistic Regression Model

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables. Logistic Regression is used when the dependent variable(target) is categorical.

```
#Gradient-boosted Tree classifier Model

#Gradient boosting is a machine learning technique for regression and
classification problems, which produces a prediction model in the form of an
ensemble of weak prediction models, typically decision trees.
```

Gradient-boosted Tree classifier Model

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees.

Double-click (or enter) to edit

```
[ ] assembler.distinct().show()
```

```
+---+---------------+-----------------+
| Id|        Species|         fetaures|
+---+---------------+-----------------+
| 60|Iris-versicolor|[5.2,2.7,3.9,1.4]|
| 36|    Iris-setosa|[5.0,3.2,1.2,0.2]|
| 63|Iris-versicolor|[6.0,2.2,4.0,1.0]|
| 71|Iris-versicolor|[5.9,3.2,4.8,1.8]|
| 76|Iris-versicolor|[6.6,3.0,4.4,1.4]|
|150| Iris-virginica|[5.9,3.0,5.1,1.8]|
| 80|Iris-versicolor|[5.7,2.6,3.5,1.0]|
| 62|Iris-versicolor|[5.9,3.0,4.2,1.5]|
|139| Iris-virginica|[6.0,3.0,4.8,1.8]|
| 96|Iris-versicolor|[5.7,3.0,4.2,1.2]|
|105| Iris-virginica|[6.5,3.0,5.8,2.2]|
|137| Iris-virginica|[6.3,3.4,5.6,2.4]|
| 61|Iris-versicolor|[5.0,2.0,3.5,1.0]|
| 32|    Iris-setosa|[5.4,3.4,1.5,0.4]|
|120| Iris-virginica|[6.0,2.2,5.0,1.5]|
|114| Iris-virginica|[5.7,2.5,5.0,2.0]|
| 81|Iris-versicolor|[5.5,2.4,3.8,1.1]|
| 78|Iris-versicolor|[6.7,3.0,5.0,1.7]|
| 85|Iris-versicolor|[5.4,3.0,4.5,1.5]|
| 89|Iris-versicolor|[5.6,3.0,4.1,1.3]|
+---+---------------+-----------------+
only showing top 20 rows
```

```
[ ] assembler.select("Species").distinct().show()
```

```
+---------------+
|        Species|
+---------------+
| Iris-virginica|
|    Iris-setosa|
|Iris-versicolor|
+---------------+
```

StringIndexer

StringIndexer maps a string column to a index column that will be treated as a categorical column by spark. The indices start with 0 and are ordered by label frequencies. If it is a numerical column, the column will first be casted to a string column and then indexed by StringIndexer.

There are three steps to implement the StringIndexer

Build the StringIndexer model: specify the input column and output column names.

Learn the StringIndexer model: fit the model with your data.

Execute the indexing: call the transform function to execute the indexing process.

```
from pyspark.ml.feature import StringIndexer
```

```
lable_indexer =StringIndexer(inputCol="Species",outputCol="lableIndex")
ml_dataset= lable_indexer.fit(assembler).transform(assembler)
ml_dataset.show(5)
```

```
+---+-----------+-----------------+----------+
| Id|    Species|        fetaures|lableIndex|
+---+-----------+-----------------+----------+
|  1|Iris-setosa|[5.1,3.5,1.4,0.2]|       0.0|
|  2|Iris-setosa|[4.9,3.0,1.4,0.2]|       0.0|
|  3|Iris-setosa|[4.7,3.2,1.3,0.2]|       0.0|
|  4|Iris-setosa|[4.6,3.1,1.5,0.2]|       0.0|
|  5|Iris-setosa|[5.0,3.6,1.4,0.2]|       0.0|
+---+-----------+-----------------+----------+
only showing top 5 rows
```

```
#After lable indexing, have to split dataset into training ans test data.
(trainDataset,testDataset) =(ml_dataset.randomSplit([0.8,0.2]))
```

```
#Everythings are set to develop our Ml Model Decision Three classifier
from pyspark.ml.classification import DecisionTreeClassifier
from pyspark.ml.feature import StringIndexer, VectorIndexer
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
```

```
dt_model =DecisionTreeClassifier(labelCol ="lableIndex",featuresCol="fetaures")
```

```
model =dt_model.fit(trainDataset)
```

```
model
```

```
DecisionTreeClassificationModel: uid=DecisionTreeClassifier_e7ba8b93f1ad, depth=5, numNodes=15, numClasses=3, numFeatures=4
```

```
predictions =model.transform(testDataset)
predictions.show(20)
```

```
+---+---------------+-----------------+----------+--------------+--------------------+----------+
| Id|        Species|         fetaures|lableIndex| rawPrediction|         probability|prediction|
+---+---------------+-----------------+----------+--------------+--------------------+----------+
|  5|    Iris-setosa|[5.0,3.6,1.4,0.2]|       0.0|[41.0,0.0,0.0]|       [1.0,0.0,0.0]|       0.0|
| 11|    Iris-setosa|[5.4,3.7,1.5,0.2]|       0.0|[41.0,0.0,0.0]|       [1.0,0.0,0.0]|       0.0|
| 13|    Iris-setosa|[4.8,3.0,1.4,0.1]|       0.0|[41.0,0.0,0.0]|       [1.0,0.0,0.0]|       0.0|
| 19|    Iris-setosa|[5.7,3.8,1.7,0.3]|       0.0|[41.0,0.0,0.0]|       [1.0,0.0,0.0]|       0.0|
| 22|    Iris-setosa|[5.1,3.7,1.5,0.4]|       0.0|[41.0,0.0,0.0]|       [1.0,0.0,0.0]|       0.0|
| 23|    Iris-setosa|[4.6,3.6,1.0,0.2]|       0.0|[41.0,0.0,0.0]|       [1.0,0.0,0.0]|       0.0|
| 27|    Iris-setosa|[5.0,3.4,1.6,0.4]|       0.0|[41.0,0.0,0.0]|       [1.0,0.0,0.0]|       0.0|
| 40|    Iris-setosa|[5.1,3.4,1.5,0.2]|       0.0|[41.0,0.0,0.0]|       [1.0,0.0,0.0]|       0.0|
| 46|    Iris-setosa|[4.8,3.0,1.4,0.3]|       0.0|[41.0,0.0,0.0]|       [1.0,0.0,0.0]|       0.0|
| 55|Iris-versicolor|[6.5,2.8,4.6,1.5]|       1.0|[0.0,38.0,1.0]|[0.0,0.9743589743...|       1.0|
| 59|Iris-versicolor|[6.6,2.9,4.6,1.3]|       1.0|[0.0,38.0,1.0]|[0.0,0.9743589743...|       1.0|
| 66|Iris-versicolor|[6.7,3.1,4.4,1.4]|       1.0|[0.0,38.0,1.0]|[0.0,0.9743589743...|       1.0|
| 70|Iris-versicolor|[5.6,2.5,3.9,1.1]|       1.0|[0.0,38.0,1.0]|[0.0,0.9743589743...|       1.0|
| 80|Iris-versicolor|[5.7,2.6,3.5,1.0]|       1.0|[0.0,38.0,1.0]|[0.0,0.9743589743...|       1.0|
| 94|Iris-versicolor|[5.0,2.3,3.3,1.0]|       1.0|[0.0,38.0,1.0]|[0.0,0.9743589743...|       1.0|
| 96|Iris-versicolor|[5.7,3.0,4.2,1.2]|       1.0|[0.0,38.0,1.0]|[0.0,0.9743589743...|       1.0|
| 98|Iris-versicolor|[6.2,2.9,4.3,1.3]|       1.0|[0.0,38.0,1.0]|[0.0,0.9743589743...|       1.0|
| 99|Iris-versicolor|[5.1,2.5,3.0,1.1]|       1.0|[0.0,38.0,1.0]|[0.0,0.9743589743...|       1.0|
|100|Iris-versicolor|[5.7,2.8,4.1,1.3]|       1.0|[0.0,38.0,1.0]|[0.0,0.9743589743...|       1.0|
|101| Iris-virginica|[6.3,3.3,6.0,2.5]|       2.0|[0.0,0.0,35.0]|       [0.0,0.0,1.0]|       2.0|
+---+---------------+-----------------+----------+--------------+--------------------+----------+
only showing top 20 rows
```

```
predictions.select("prediction","lableIndex").show(5)
```

```
+----------+----------+
|prediction|lableIndex|
+----------+----------+
|       0.0|       0.0|
|       0.0|       0.0|
|       0.0|       0.0|
|       0.0|       0.0|
|       0.0|       0.0|
+----------+----------+
```

```
only showing top 5 rows
```

```
[ ] evaluator = MulticlassClassificationEvaluator(labelCol ="lableIndex",predictionCol ="prediction")
    accuracy =evaluator.evaluate(predictions)
    print("Accuracy:",accuracy)
    print("test Error=%g " %(1.0- accuracy))
```

```
Exception ignored in: <function JavaWrapper.__del__ at 0x7deeb922e3b0>
Traceback (most recent call last):
  File "/usr/local/lib/python3.10/dist-packages/pyspark/ml/wrapper.py", line 53, in __del__
    if SparkContext._active_spark_context and self._java_obj is not None:
AttributeError: 'MulticlassClassificationEvaluator' object has no attribute '_java_obj'
Accuracy: 0.9666666666666666
test Error=0.0333333
```