databricks Linear_regression 09-10-2023 Notebook 2023-10-09 11:05:48

(https://databricks.com)

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('lin_reg').getOrCreate()
```

```
# import
from pyspark.ml.regression import LinearRegression
```

```
# Load Dataset
df = spark.read.format("csv").option("header", "true").load("dbfs:/FileStore/shared_uploads/devjethva234@gmail.com/Linear_regression_d
```

```
# validate the size of the data
print((df.count(), len(df.columns)))
```

```
(1232, 6)
```

```
# explore the data
df.printSchema()
```

```
root
 |-- var_1: integer (nullable = true)
 |-- var_2: integer (nullable = true)
 |-- var_3: integer (nullable = true)
 |-- var_4: double (nullable = true)
 |-- var_5: double (nullable = true)
 |-- output: double (nullable = true)
```

```
# view statistical mesures of the data
df.describe().show(5,False)
```

```
+-------+-----------------+-----------------+------------------+--------------------+--------------------+-------------------+
|summary|var_1            |var_2            |var_3             |var_4               |var_5               |output             |
+-------+-----------------+-----------------+------------------+--------------------+--------------------+-------------------+
|count  |1232             |1232             |1232              |1232                |1232                |1232               |
|mean   |715.0819805194806|715.0819805194806|80.90422077922078 |0.3263311688311693  |0.25927272727272715 |0.39734172077922014|
|stddev |91.5342940441652 |93.07993263118064|11.458139049993724|0.015012772334166148|0.012907228928000298|0.03326689862173776|
|min    |463              |472              |40                |0.277               |0.214               |0.301              |
|max    |1009             |1103             |116               |0.373               |0.294               |0.491              |
+-------+-----------------+-----------------+------------------+--------------------+--------------------+-------------------+
```

```
# sneak into the dataset
df.head(3)
```

```
Out[78]: [Row(var_1=734, var_2=688, var_3=81, var_4=0.328, var_5=0.259, output=0.418),
 Row(var_1=700, var_2=600, var_3=94, var_4=0.32, var_5=0.247, output=0.389),
 Row(var_1=712, var_2=705, var_3=93, var_4=0.311, var_5=0.247, output=0.417)]
```

```
# import corr function from pyspark functions
from pyspark.sql.functions import corr
```

```
# check for correlation & how to add variable using corr functions
df.select(corr('var_1', 'output')).show()
```

```
+-------------------+
|corr(var_1, output)|
+-------------------+
| 0.9187399607627283|
+-------------------+
```

```
# import
from pyspark.ml.linalg import Vector
from pyspark.ml.feature import VectorAssembler
```

```
# select the columns to create the input vector
df.columns
```

```
Out[82]: ['var_1', 'var_2', 'var_3', 'var_4', 'var_5', 'output']
```

```
# create the vector assembler
vec_assembler = VectorAssembler(inputCols = ['var_1', 'var_2', 'var_3', 'var_4', 'var_5'], outputCol='features')
```

```
# transform
features_df = vec_assembler.transform(df)
```

```
features_df.printSchema()
```

```
root
 |-- var_1: integer (nullable = true)
 |-- var_2: integer (nullable = true)
 |-- var_3: integer (nullable = true)
 |-- var_4: double (nullable = true)
 |-- var_5: double (nullable = true)
 |-- output: double (nullable = true)
 |-- features: vector (nullable = true)
```

```
features_df.select('features').show(5,False)
```

```
+------------------------------+
|features                      |
+------------------------------+
|[734.0,688.0,81.0,0.328,0.259]|
|[700.0,600.0,94.0,0.32,0.247] |
|[712.0,705.0,93.0,0.311,0.247]|
|[734.0,806.0,69.0,0.315,0.26] |
|[613.0,759.0,61.0,0.302,0.24] |
+------------------------------+
only showing top 5 rows
```

```
model_df = features_df.select('features', 'output')
```

```
model_df.show(5,False)
```

```
+------------------------------+------+
|features                      |output|
+------------------------------+------+
|[734.0,688.0,81.0,0.328,0.259]|0.418 |
|[700.0,600.0,94.0,0.32,0.247] |0.389 |
|[712.0,705.0,93.0,0.311,0.247]|0.417 |
|[734.0,806.0,69.0,0.315,0.26] |0.415 |
|[613.0,759.0,61.0,0.302,0.24] |0.378 |
+------------------------------+------+
only showing top 5 rows
```

```
print((model_df.count(), len(model_df.columns)))
```

```
(1232, 2)
```

Spilt data - Train & Test Data

```
# sec
```

```
train_df, test_df = model_df.randomSplit([0.7,0.3])
```

```
print((train_df.count(), len(train_df.columns)))
```

```
(880, 2)
```

```
print((test_df.count(), len(test_df.columns)))
```

```
(352, 2)
```

```
train_df.describe().show()
```

```
+-------+--------------------+
|summary|              output|
+-------+--------------------+
|  count|                 880|
|   mean|  0.3979874999999994|
| stddev|0.033438290887800266|
|    min|               0.301|
|    max|               0.491|
+-------+--------------------+
```

```
# Build Linear Regression Model
```

```
lin_reg = LinearRegression(labelCol='output')
```

```
# fit the linear model on training dataset
lr_model = lin_reg.fit(train_df)
```

```
lr_model.intercept
```

```
Out[100]: 0.18626961441250367
```

```
print(lr_model.coefficients)
```

```
[0.0003371220458572444,5.991498706752434e-05,0.0002556814872303228,-0.6686443805619972,0.48044283582971575]
```

```
training_predictions = lr_model.evaluate(train_df)
```

```
training_predictions.meanSquaredError
```

```
Out[103]: 0.00013704630021768094
```

```
training_predictions.r2
```

```
Out[104]: 0.8772919740056858
```

```
# make predictions on the test data
test_results= lr_model.evaluate(test_df)
```

```
# view the residual errors based on the predictions
test_results.residuals.show(10)
```

```
+--------------------+
|           residuals|
+--------------------+
|0.009429260836439135|
|-9.42564848742444...|
|0.013713981224977412|
|-0.01210437809228...|
|-0.00679164860545...|
|0.010048355854590463|
| 8.97412483862492E-4|
|  -0.012396549858743|
|-0.01181377619641...|
|-0.00117635064512...|
+--------------------+
only showing top 10 rows
```

```
# coefficient of determination value for model
test_results.r2
```

```
Out[107]: 0.8476131884871057
```

```
test_results.rootMeanSquaredError
```

```
Out[108]: 0.012796007916146983
```

```
Out[109]: 0.00016373781859009624
```