Published in Analytics Vidhya

Debpriyo Roy Follow

Dec 20, 2019 · 4 min read · Listen

Save

Linear Regression and Decision Tree Implementation using Pyspark

We will be building a simple Linear regression and Decision tree to help you get started with pyspark. The data set taken into consideration is a small cars data set.

The first thing which needs to be done is to find spark.

```
import findspark
findspark.init()
findspark.find()
import pyspark
findspark.find()
```

The next step is to create a Spark session.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from numpy import polyfit
from pyspark.sql import SparkSession
from pyspark import SparkContext,SparkConf,SQLContext

conf = SparkConf().setMaster('local').setAppName('ML_learning')
sc = SparkContext(conf=conf)
```

```
sqlcontext = SQLContext(sc)
```

Loading the data which is a csv file in this case.

```
data = sqlcontext.read.csv(path='C:\AAI\week 3\Small_Car_Data.csv',
header = True, inferSchema = True)

data.show()
```

```
+---+------------+---------+------------+----------+-------------+--------------------+----------+----+-------+------+
|_c0|Acceleration|Cylinders|Displacement|Horsepower| Manufacturer|               Model|Model_Year| MPG| Origin|Weight|
+---+------------+---------+------------+----------+-------------+--------------------+----------+----+-------+------+
|  1|        12.0|        8|         307|       130|chevrolet    |chevrolet chevell...|        70|18.0|USA    |  3504|
|  2|        11.5|        8|         350|       165|buick        |buick skylark 320...|        70|15.0|USA    |  3693|
|  3|        11.0|        8|         318|       150|plymouth     |plymouth satellit...|        70|18.0|USA    |  3436|
|  4|        12.0|        8|         304|       150|amc          |amc rebel sst    ...|        70|16.0|USA    |  3433|
|  5|        10.5|        8|         302|       140|ford         |ford torino      ...|        70|17.0|USA    |  3449|
|  6|        10.0|        8|         429|       198|ford         |ford galaxie 500 ...|        70|15.0|USA    |  4341|
|  7|         9.0|        8|         454|       220|chevrolet    |chevrolet impala ...|        70|14.0|USA    |  4354|
|  8|         8.5|        8|         440|       215|plymouth     |plymouth fury iii...|        70|14.0|USA    |  4312|
|  9|        10.0|        8|         455|       225|pontiac      |pontiac catalina ...|        70|14.0|USA    |  4425|
| 10|         8.5|        8|         390|       190|amc          |amc ambassador dp...|        70|15.0|USA    |  3850|
| 11|        17.5|        4|         133|       115|citroen      |citroen ds-21 pal...|        70|14.0|France |  3090|
| 12|        11.5|        8|         350|       165|chevrolet    |chevrolet chevell...|        70|16.0|USA    |  4142|
| 13|        11.0|        8|         351|       153|ford         |ford torino (sw) ...|        70|17.0|USA    |  4034|
| 14|        10.5|        8|         383|       175|plymouth     |plymouth satellit...|        70|19.0|USA    |  4166|
| 15|        11.0|        8|         360|       175|amc          |amc rebel sst (sw...|        70|20.0|USA    |  3850|
| 16|        10.0|        8|         383|       170|dodge        |dodge challenger ...|        70|15.0|USA    |  3563|
| 17|         8.0|        8|         340|       160|plymouth     |plymouth 'cuda 34...|        70|14.0|USA    |  3609|
| 18|         8.0|        8|         302|       140|ford         |ford mustang boss...|        70|16.0|USA    |  3353|
| 19|         9.5|        8|         400|       150|chevrolet    |chevrolet monte c...|        70|15.0|USA    |  3761|
| 20|        10.0|        8|         455|       225|buick        |buick estate wago...|        70|14.0|USA    |  3086|
+---+------------+---------+------------+----------+-------------+--------------------+----------+----+-------+------+
only showing top 20 rows
```

small cars dataset

Now, import the required modules for implementing linear regression.

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml.evaluation import RegressionEvaluator
```

Initially performing linear regression on single variable and plotting the result for better understanding.

Splitting the data for training and testing purposes.

```
data2 =data.select(data.Displacement,data.Horsepower.alias('label'))
train, test = data2.randomSplit([0.9,0.1])
```

Vector Assembler is used for creating a vector of all the individual features.

```
assembler=VectorAssembler().setInputCols(['Displacement',])\.setOutput
Col('features')
train01 = assembler.transform(train)

train02 = train01.select("features","label")
train02.show(5)
```

```
+--------+-----+
|features|label|
+--------+-----+
|  [85.0]|   52|
|  [85.0]|   70|
|  [90.0]|   70|
|  [91.0]|   53|
|  [91.0]|   67|
+--------+-----+
only showing top 5 rows
```

Features: Vector of all the individual features.

Label: Target Variable.

In this case since initially we are building a uni variate linear regression model, only one feature ‘Displacement’ is converted to a vector.

Now, building the model.

```
lr = LinearRegression()
model = lr.fit(train02)
```

Testing our model against unseen data.

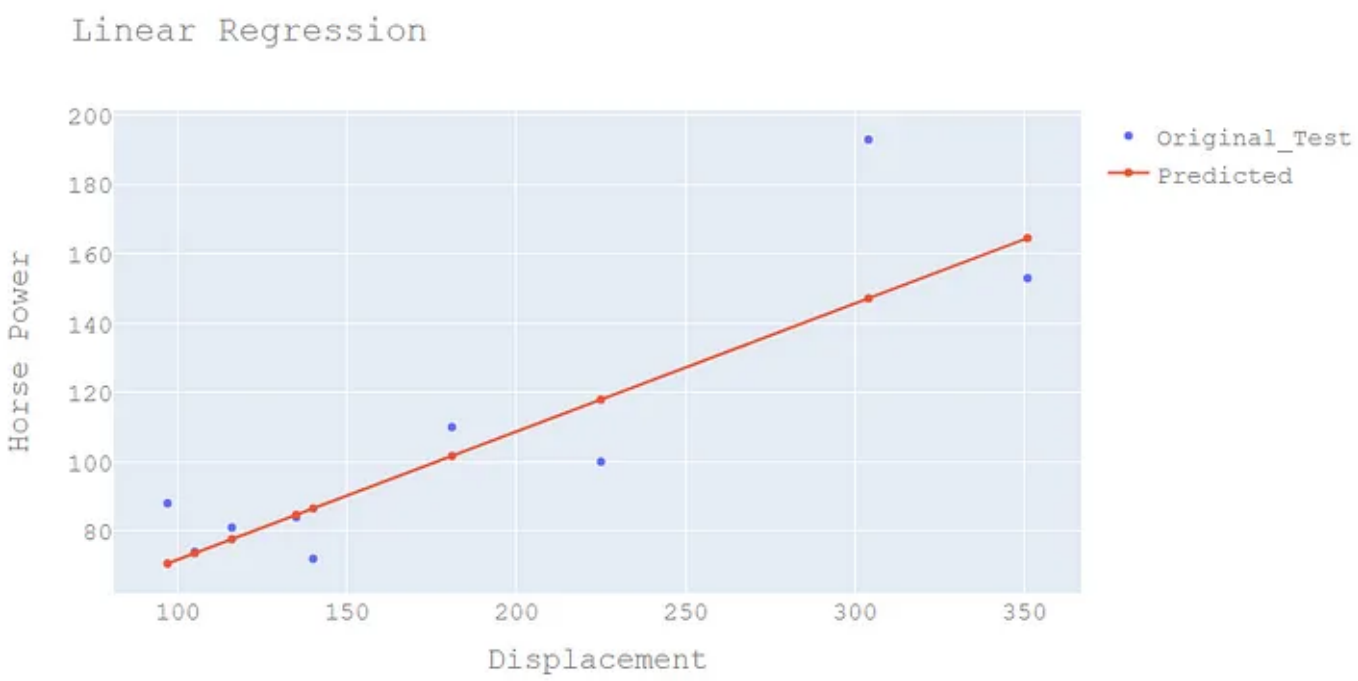
```
test01 = assembler.transform(test)
test02 = test01.select('features', 'label')
test03 = model.transform(test02)
test03.show(5)
```

```
+--------+-----+-----------------+
|features|label|       prediction|
+--------+-----+-----------------+
|  [97.0]|   88|70.62220050293436|
| [105.0]|   74|73.58076048288027|
| [116.0]|   81|77.64878045530585|
| [135.0]|   84|84.67536040767735|
| [140.0]|   72|86.52446039514352|
+--------+-----+-----------------+
only showing top 5 rows
```

Plotting the linear fit line.

```
import chart_studio.plotly as py
import plotly.graph_objects as go
fig = go.Figure()

fig.add_trace(
    go.Scatter(
        x=x,
        y=y,
        mode='markers',
        name='Original_Test',
    ))

fig.add_trace(
    go.Scatter(
        x=x,
        y=y_pred,
        name='Predicted'
    ))
fig.update_layout(
    title="Linear Regression",
    xaxis_title="Displacement",
    yaxis_title="Horse Power",
    font=dict(
        family="Courier New, monospace",
        size=18,
        color="#7f7f7f"
    )
)
```

```
fig.show()
```



Evaluation of model.

```
evaluator = RegressionEvaluator()
print(evaluator.evaluate(test03,
{evaluator.metricName: "r2"})
)

print(evaluator.evaluate(test03,
{evaluator.metricName: "mse"})
)
print(evaluator.evaluate(test03,
{evaluator.metricName: "rmse"})
)
print(evaluator.evaluate(test03,
{evaluator.metricName: "mae"})
)
```

Now, let's start building multivariate Linear Regression and Decision Tree model.

Selecting the columns relevant to model building.

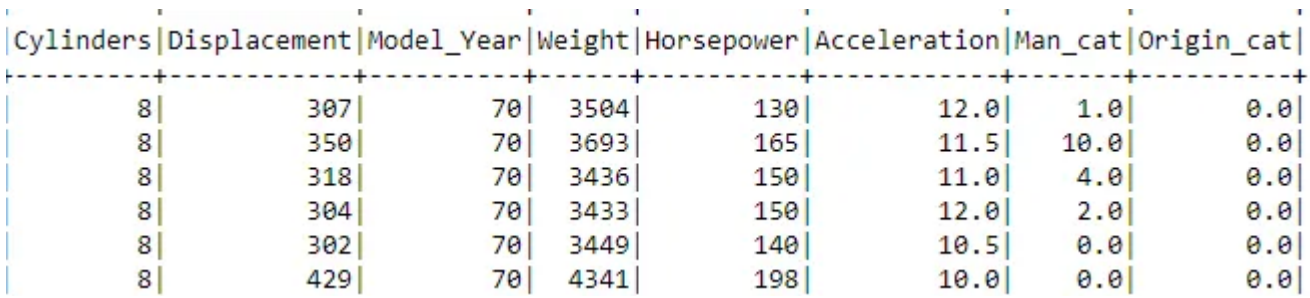
```
small_cars=data.select(data._c0,data.Cylinders,data.Displacement,data.
Manufacturer,data.Model_Year,data.Origin,data.Weight,data.Horsepower,d
ata.Acceleration)
small_cars.show()
```

```
+---+---------+------------+-------------+----------+-------+------+----------+------------+
|_c0|Cylinders|Displacement| Manufacturer|Model_Year| Origin|Weight|Horsepower|Acceleration|
+---+---------+------------+-------------+----------+-------+------+----------+------------+
|  1|        8|         307|chevrolet    |        70|USA    |  3504|       130|        12.0|
|  2|        8|         350|buick        |        70|USA    |  3693|       165|        11.5|
|  3|        8|         318|plymouth     |        70|USA    |  3436|       150|        11.0|
|  4|        8|         304|amc          |        70|USA    |  3433|       150|        12.0|
|  5|        8|         302|ford         |        70|USA    |  3449|       140|        10.5|
|  6|        8|         429|ford         |        70|USA    |  4341|       198|        10.0|
|  7|        8|         454|chevrolet    |        70|USA    |  4354|       220|         9.0|
|  8|        8|         440|plymouth     |        70|USA    |  4312|       215|         8.5|
|  9|        8|         455|pontiac      |        70|USA    |  4425|       225|        10.0|
| 10|        8|         390|amc          |        70|USA    |  3850|       190|         8.5|
| 11|        4|         133|citroen      |        70|France |  3090|       115|        17.5|
| 12|        8|         350|chevrolet    |        70|USA    |  4142|       165|        11.5|
| 13|        8|         351|ford         |        70|USA    |  4034|       153|        11.0|
| 14|        8|         383|plymouth     |        70|USA    |  4166|       175|        10.5|
| 15|        8|         360|amc          |        70|USA    |  3850|       175|        11.0|
| 16|        8|         383|dodge        |        70|USA    |  3563|       170|        10.0|
| 17|        8|         340|plymouth     |        70|USA    |  3609|       160|         8.0|
| 18|        8|         302|ford         |        70|USA    |  3353|       140|         8.0|
| 19|        8|         400|chevrolet    |        70|USA    |  3761|       150|         9.5|
| 20|        8|         455|buick        |        70|USA    |  3086|       225|        10.0|
+---+---------+------------+-------------+----------+-------+------+----------+------------+
only showing top 20 rows
```

Before building the model, some pre-processing of the dataset is required. The variables such as ‘ORIGIN’ and ‘MANUFACTURER’ have categorical values which needs to be converted.

```
from pyspark.ml.feature import StringIndexer

indexer = StringIndexer(inputCol="Origin", outputCol="Origin_cat")

indexed = indexer.fit(small_cars).transform(small_cars)

origin_cat=indexed.select(indexed._c0,indexed.Origin_cat)

indexer_1 = StringIndexer(inputCol="Manufacturer",
outputCol="Man_cat")

indexed_1 = indexer_1.fit(small_cars).transform(small_cars)
man_cat=indexed_1.select(indexed_1._c0,indexed_1.Man_cat)
```

```
inner_join = small_cars.join(man_cat, small_cars._c0 ==
man_cat._c0).drop(man_cat._c0)

inner_join_1=inner_join.join(origin_cat,inner_join._c0==origin_cat._c0
).drop('_c0','Manufacturer','Origin')

inner_join_1.show(5)
```



Cylinders	Displacement	Model_Year	Weight	Horsepower	Acceleration	Man_cat	Origin_cat
8	307	70	3504	130	12.0	1.0	0.0
8	350	70	3693	165	11.5	10.0	0.0
8	318	70	3436	150	11.0	4.0	0.0
8	304	70	3433	150	12.0	2.0	0.0
8	302	70	3449	140	10.5	0.0	0.0
8	429	70	4341	198	10.0	0.0	0.0

Splitting the data for training and testing.

```
train, test = inner_join_1.randomSplit([0.9,0.1])
```

Using Vector Assembler to create a vector for all the individual features.

```
assembler=VectorAssembler().setInputCols(['Displacement','Cylinders','
Model_Year','Weight','Man_cat','Origin_cat'])\
.setOutputCol('features')

train_a = assembler.transform(train)

train_b=train_a.select("features",train_a.Horsepower.alias('label'))

train_b.show(truncate=False)
```

```
+-------------------------------+-----+
|features                       |label|
+-------------------------------+-----+
|[85.0,4.0,76.0,1990.0,9.0,1.0] |70   |
|[85.0,4.0,76.0,2035.0,1.0,0.0] |52   |
|[90.0,4.0,76.0,1937.0,6.0,2.0] |70   |
|[91.0,4.0,76.0,1795.0,8.0,1.0] |53   |
|[91.0,4.0,82.0,1965.0,8.0,1.0] |67   |
|[91.0,4.0,82.0,1965.0,8.0,1.0] |67   |
|[91.0,4.0,82.0,1970.0,12.0,1.0]|68   |
|[91.0,4.0,82.0,1995.0,9.0,1.0] |67   |
|[97.0,4.0,70.0,1835.0,6.0,2.0] |46   |
|[97.0,4.0,70.0,2130.0,9.0,1.0] |88   |
|[97.0,4.0,76.0,1825.0,6.0,2.0] |71   |
|[97.0,4.0,76.0,2155.0,5.0,1.0] |75   |
|[97.0,4.0,82.0,2130.0,6.0,2.0] |52   |
|[98.0,4.0,76.0,2164.0,1.0,0.0] |60   |
|[98.0,4.0,76.0,2255.0,3.0,0.0] |79   |
|[98.0,4.0,82.0,2125.0,18.0,0.0]|70   |
|[101.0,4.0,76.0,2202.0,19.0,3.0]|83  |
|[104.0,4.0,70.0,2375.0,16.0,4.0]|95  |
|[105.0,4.0,82.0,1980.0,6.0,2.0]|74   |
|[105.0,4.0,82.0,2125.0,4.0,0.0]|63   |
+-------------------------------+-----+
only showing top 20 rows
```

```
lr = LinearRegression()
model = lr.fit(train_b)
test_a = assembler.transform(test)
test_b = test_a.select('features', test_a.Horsepower.alias('label'))
test_c = model.transform(test_b)
test_c.show(truncate=False)
```

```
+-------------------------------+-----+------------------+
|features                       |label|prediction        |
+-------------------------------+-----+------------------+
|[91.0,4.0,82.0,2025.0,12.0,1.0]|68   |62.48024404320351 |
|[110.0,4.0,70.0,2672.0,11.0,3.0]|87  |94.34315983389409 |
|[120.0,4.0,76.0,3270.0,11.0,3.0]|88  |102.68364046209135|
|[121.0,4.0,70.0,2234.0,25.0,2.0]|113 |97.85901468053534 |
|[304.0,8.0,70.0,4732.0,14.0,0.0]|193 |171.09799520339473|
|[350.0,8.0,70.0,4142.0,1.0,0.0]|165  |167.38958988609176|
|[350.0,8.0,76.0,4055.0,1.0,0.0]|145  |159.9030313966342 |
|[350.0,8.0,76.0,4380.0,24.0,0.0]|180 |180.71432815350295|
|[383.0,8.0,70.0,3563.0,3.0,0.0]|170  |169.22965496027015|
|[454.0,8.0,70.0,4354.0,1.0,0.0]|220  |206.17891340929728|
+-------------------------------+-----+------------------+
```

Evaluating the model.

```
evaluator = RegressionEvaluator()
print(evaluator.evaluate(test_c,
{evaluator.metricName: "r2"})
)

print(evaluator.evaluate(test_c,
{evaluator.metricName: "mse"})
)
print(evaluator.evaluate(test_c,
{evaluator.metricName: "rmse"})
)
print(evaluator.evaluate(test_c,
{evaluator.metricName: "mae"})
```



```
from pyspark.ml.regression import DecisionTreeRegressor

dt = DecisionTreeRegressor()
model = dt.fit(train_b)
test_dt = model.transform(test_b)
test_dt.show(truncate=False)
```

```
+--------------------------------+-----+-----------------+
|features                        |label|prediction       |
+--------------------------------+-----+-----------------+
|[91.0,4.0,82.0,2025.0,12.0,1.0] |68   |66.36363636363636|
|[110.0,4.0,70.0,2672.0,11.0,3.0]|87   |117.5            |
|[120.0,4.0,76.0,3270.0,11.0,3.0]|88   |117.5            |
|[121.0,4.0,70.0,2234.0,25.0,2.0]|113  |85.0             |
|[304.0,8.0,70.0,4732.0,14.0,0.0]|193  |210.0            |
|[350.0,8.0,70.0,4142.0,1.0,0.0] |165  |165.0            |
|[350.0,8.0,76.0,4055.0,1.0,0.0] |145  |165.0            |
|[350.0,8.0,76.0,4380.0,24.0,0.0]|180  |215.0            |
|[383.0,8.0,70.0,3563.0,3.0,0.0] |170  |150.0            |
|[454.0,8.0,70.0,4354.0,1.0,0.0] |220  |215.0            |
+--------------------------------+-----+-----------------+
```

Evaluating the model

```
evaluator = RegressionEvaluator()
print(evaluator.evaluate(test_dt,
```

```
{evaluator.metricName: "r2"})
)

print(evaluator.evaluate(test_dt,
{evaluator.metricName: "mse"})
)
print(evaluator.evaluate(test_dt,
{evaluator.metricName: "rmse"})
)
print(evaluator.evaluate(test_dt,
{evaluator.metricName: "mae"})
)
```


This is how to build a simple Linear Regression and Decision Tree model in Pyspark.

Thanks

Machine Learning | Pyspark | Linear Regression | Decision Tree