Statistical Solutions to Big Data Problems using Python and Apache Spark

Christian Jordan
Student Number: 14061768

Plagiarism Declaration

With the exception of any statement to the contrary, all the material presented in this report is the result of my own efforts. In addition, no parts of this report are copied from other sources. I understand that any evidence of plagiarism and/or the use of unacknowledged third party materials will be dealt with as a serious matter.

Signed:

Department of Computing and Mathematics
Manchester Metropolitan University
United Kingdom
12th May 2023

1 Bike Rides Analysis

The basis of this project is to analyse a "Big Data" dataset using PySpark to give conclusive evidence for the research hypothesis:

"People ride for longer in Spring than in Autumn in London in the year 2014."

This research can be important for cycle hire companies, knowing how seasonal variation in demand for longer cycle hires can affect stock levels.

1.1 Terms of Hypothesis

To evaluate the hypothesis stated, first determine what constitutes "longer" for the difference in duration. For this experiment, "longer" is defined as the mean duration in Spring to be statistically greater than the mean duration in Autumn at the 5% significance level.

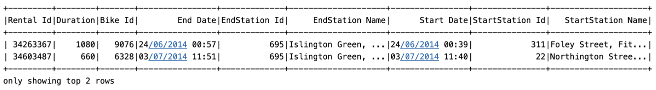
According to Met-Office (n.d.), meteorological Spring spans from 1st March - 31st May and Autumn from 1st September - 30th November, which has been used to compare seasons in this project.

1.2 Data Collection and Preparation

The dataset used for this research is provided by Transport for London courtesy of the UK Government and can be located on the url below (London n.d.).

`https://cycling.data.tfl.gov.uk/usage-stats/cyclehireusagestats-2014.zip`

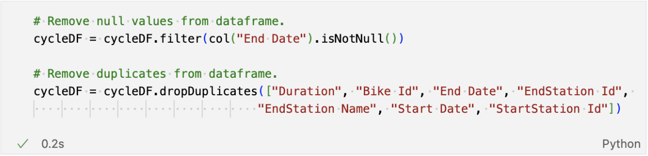
Initial exploration of the cycle hire dataset return the table shown in Figure 1.



```
+----------+--------+-------+----------------+--------------+--------------------+----------------+---------------+--------------------+
|Rental Id|Duration|Bike Id|        End Date|EndStation Id|     EndStation Name|      Start Date|StartStation Id|   StartStation Name|
+----------+--------+-------+----------------+--------------+--------------------+----------------+---------------+--------------------+
| 34263367|    1080|   9076|24/06/2014 00:57|           695|Islington Green, ...|24/06/2014 00:39|            311|Foley Street, Fit...|
| 34603487|     660|   6328|03/07/2014 11:51|           695|Islington Green, ...|03/07/2014 11:40|             22|Northington Stree...|
+----------+--------+-------+----------------+--------------+--------------------+----------------+---------------+--------------------+
only showing top 2 rows
```

Figure 1: First 2 rows of cycle hire dataframe.

The features of interest in relation to the hypothesis are "Duration" and "End Date", therefore observations that have missing values within these have been removed. Duplicate observations have also been removed if matched in all features except "Rental ID" as shown in Figure 2.

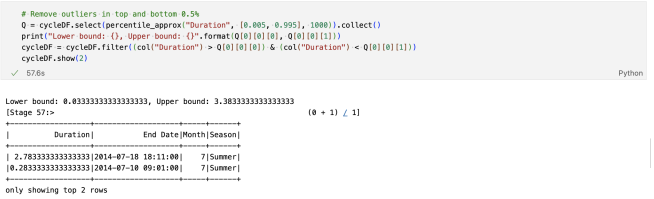


```
# Remove null values from dataframe.
cycleDF = cycleDF.filter(col("End Date").isNotNull())

# Remove duplicates from dataframe.
cycleDF = cycleDF.dropDuplicates(["Duration", "Bike Id", "End Date", "EndStation Id",
                                  "EndStation Name", "Start Date", "StartStation Id"])
```

Figure 2: Remove missing values and duplicates.

From the summary statistics and distribution plots in Section 1.3, there are some heavy outliers which may affect our analysis. Although some of them might be legitimate observations, they are not representative of the average population. Therefore, choose to remove extreme outliers in the top and bottom 0.5%, as shown in Figure 3.



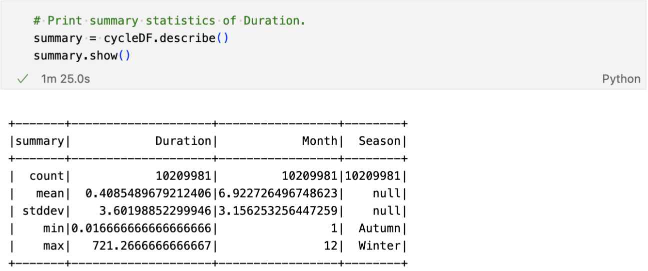
```
# Remove outliers in top and bottom 0.5%
Q = cycleDF.select(percentile_approx("Duration", [0.005, 0.995], 1000)).collect()
print("Lower bound: {}, Upper bound: {}".format(Q[0][0][0], Q[0][0][1]))
cycleDF = cycleDF.filter((col("Duration") > Q[0][0][0]) & (col("Duration") < Q[0][0][1]))
cycleDF.show(2)
```

```
Lower bound: 0.03333333333333333, Upper bound: 3.3833333333333333
[Stage 57:>                                                         (0 + 1) / 1]
+------------------+-------------------+-----+------+
|          Duration|           End Date|Month|Season|
+------------------+-------------------+-----+------+
| 2.783333333333333|2014-07-18 18:11:00|    7|Summer|
|0.2833333333333333|2014-07-10 09:01:00|    7|Summer|
+------------------+-------------------+-----+------+
only showing top 2 rows
```

Figure 3: Remove outliers from data.

1.3 Descriptive Analysis

From the dataframe summary in Figure 4, there are 10209981 total observations, with mean duration of approximately 0.408 hours. Interestingly, the standard deviation of the duration is large in comparison to the mean, suggesting potential for outliers and skewed distribution.



```
# Print summary statistics of Duration.
summary = cycleDF.describe()
summary.show()
```

```
+-------+--------------------+-----------------+--------+
|summary|            Duration|            Month|  Season|
+-------+--------------------+-----------------+--------+
|  count|            10209981|         10209981|10209981|
|   mean|  0.4085489679212406|6.927726496748623|    null|
| stddev|    3.60198852299946|3.156253256447259|    null|
|    min|0.016666666666666666|                1|  Autumn|
|    max|   721.2666666666667|               12|  Winter|
+-------+--------------------+-----------------+--------+
```

Figure 4: Summary statistics of dataset.

The skewed distribution is confirmed in Figure 5, with a large positive skew, that is, most observations lie at the lower end of duration.

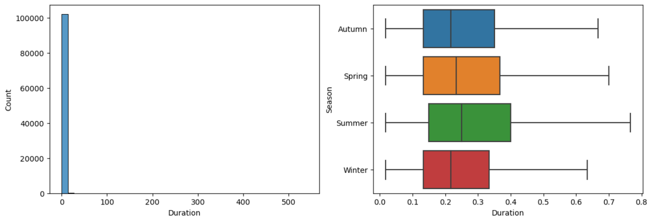


Figure 5: Distribution of duration and box plot per season.

With regards to the hypothesis, we are more interested in the distribution differences between Spring and Autumn, as shown in Figure 6. Observing that the distribution of duration in the two seasons is roughly similar, and that the means and IQR of the data differs, perform a statistical test in Section 1.4 to see if this difference is significant.

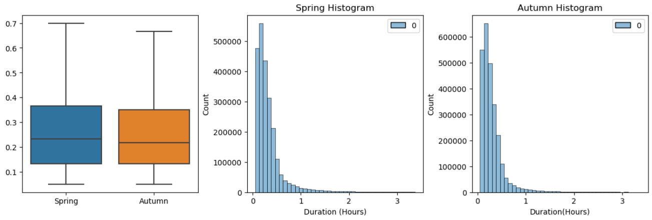


Figure 6: Distribution of Spring and Autumn and Box-Plot Comparison

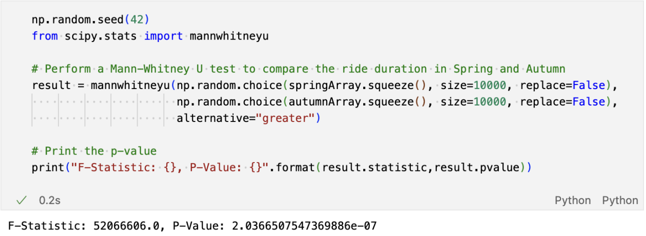
1.3.1 Choosing a Statistical Test

In Section 1.1, the terms for evaluating the hypothesis were set out as a difference in means, and a statistical test can be chosen in line with this. A paired T-test is not suitable since the data is not normally distributed (from Figure 6), therefore look to a non-parametric test such as the Mann-Whitney U Test, that only assumes observations in one group are independent of the other.

1.4 Results

Choosing a random sample of 10,000 observations from both Spring and Autumn, then running the Mann-Whitney U test where:

- H_0: The distribution of values in Spring are the same as the distribution in Autumn.
- H_1: The distribution of values in Spring is greater than the distribution in Autumn.



```
np.random.seed(42)
from scipy.stats import mannwhitneyu

# Perform a Mann-Whitney U test to compare the ride duration in Spring and Autumn
result = mannwhitneyu(np.random.choice(springArray.squeeze(), size=10000, replace=False),
                      np.random.choice(autumnArray.squeeze(), size=10000, replace=False),
                      alternative="greater")

# Print the p-value
print("F-Statistic: {}, P-Value: {}".format(result.statistic,result.pvalue))
```

```
F-Statistic: 52066606.0, P-Value: 2.0366507547369886e-07
```

Figure 7: Mann-Whitney U Test in Python.

Since the P-Value is less than 0.05, there is enough evidence at the 95% significance level to reject the null hypothesis and conclude that in 2014, people do ride for longer in Spring than in Autumn.

1.5 Conclusion

The Mann-Whitney U Test shows that the difference in means of cycle hire duration in Spring is greater than in Autumn, confirmed by the bar chart with error bars in Figure 8. Further works could determine if the season is the cause of the difference in means, or if the relationship is with weather conditions, for example.

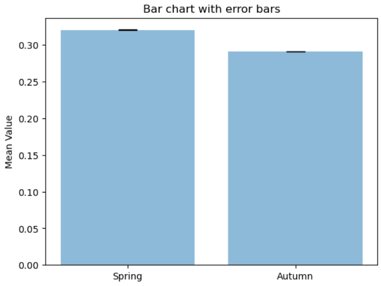


Figure 8: Spring VS. Autumn bar chart with error bars.

2 Research 2 Analysis

Following on from the research carried out in Section 1 of this report, the second research hypothesis question is:

”People ride for longer when there is no wind in London in the year 2014.”

This analysis can build on the previous hypothesis and help cycle hire companies plan for periods of higher/lower demand based on the wind speed forecast.

2.1 Terms of Hypothesis

To evaluate the hypothesis outlined above, find if there is a declining relationship between the duration of cycle hires and the corresponding wind speed on those dates in London in 2014.

2.2 Data Collection and Preparation

Utilising the same dataset for cycle hire that was outlined in the first hypothesis, found here. In addition to this, wind speeds need to be collected, and can be found at the following web location (Wunderground n.d.).

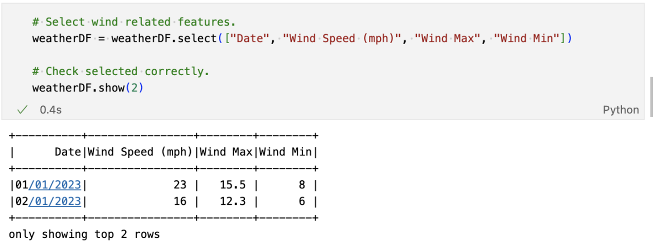
`https://www.wunderground.com/history/monthly/gb/london/EGLC/date`

Data from this location has been copied into Microsoft Excel for use in this project.

2.2.1 Descriptive Analysis

Starting with the cycle hire data, duplicate values have been removed if they are duplicated in all rows except ”Rental Id”, then the top and bottom 0.5% of duration outliers have been removed to increase the power of any statistical tests we deploy.

Exploring the weather dataset, find the following features extracted from the import in Figure 9.



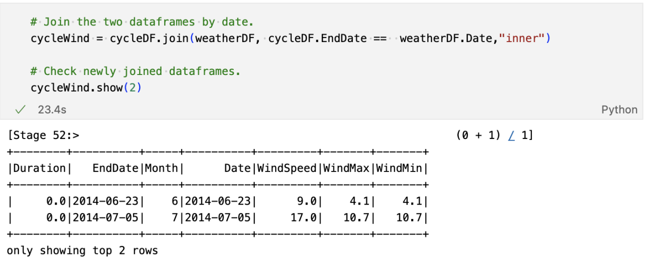
```
# Select wind related features.
weatherDF = weatherDF.select(["Date", "Wind Speed (mph)", "Wind Max", "Wind Min"])

# Check selected correctly.
weatherDF.show(2)
```

```
+----------+----------------+--------+--------+
|      Date|Wind Speed (mph)|Wind Max|Wind Min|
+----------+----------------+--------+--------+
|01/01/2023|              23 |   15.5 |      8 |
|02/01/2023|              16 |   12.3 |      6 |
+----------+----------------+--------+--------+
only showing top 2 rows
```

Figure 9: First rows of weather dataframe.

Joining the cycle hire and weather data provides the resulting dataframe in Figure 10.



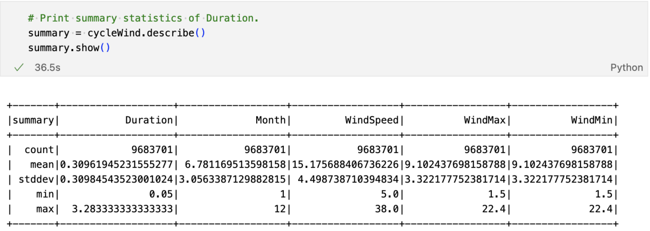
```
# Join the two dataframes by date.
cycleWind = cycleDF.join(weatherDF, cycleDF.EndDate ==  weatherDF.Date,"inner")

# Check newly joined dataframes.
cycleWind.show(2)
```

```
[Stage 52:>                                                         (0 + 1) / 1]
+--------+----------+-----+----------+---------+-------+-------+
|Duration|   EndDate|Month|      Date|WindSpeed|WindMax|WindMin|
+--------+----------+-----+----------+---------+-------+-------+
|     0.0|2014-06-23|    6|2014-06-23|      9.0|    4.1|    4.1|
|     0.0|2014-07-05|    7|2014-07-05|     17.0|   10.7|   10.7|
+--------+----------+-----+----------+---------+-------+-------+
only showing top 2 rows
```

Figure 10: Resulting dataframe of inner join.

The resulting table of summary statistics in Figure 11



```
# Print summary statistics of Duration.
summary = cycleWind.describe()
summary.show()
```

```
+-------+-------------------+------------------+------------------+-----------------+-----------------+
|summary|           Duration|             Month|         WindSpeed|          WindMax|          WindMin|
+-------+-------------------+------------------+------------------+-----------------+-----------------+
|  count|            9683701|           9683701|           9683701|          9683701|          9683701|
|   mean|0.30961945231555277| 6.781169513598158|15.175688406736226|9.102437698158788|9.102437698158788|
| stddev|0.30984543523001024|3.0563387129882815| 4.498738710394834|3.322177752381714|3.322177752381714|
|    min|               0.05|                 1|               5.0|              1.5|              1.5|
|    max|  3.283333333333333|                12|              38.0|             22.4|             22.4|
+-------+-------------------+------------------+------------------+-----------------+-----------------+
```

Figure 11: Summary statistics of joined data.

Plotting the distribution of duration, it is again clear that the data is positively skewed. Visualising how the wind speed affects the duration can be more difficult. In Figure 12, the data has been separated into 5 bins based on wind speed, showing how the mean duration can vary based on wind speed (for a 20% data sample).

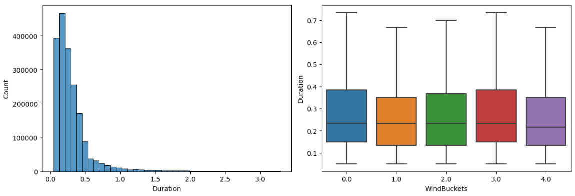
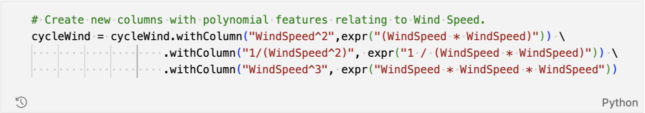


Figure 12: Distribution of duration and binned wind speed.

2.2.2 Choosing a Statistical Test

Describing relationships between two continuous variables can be completed using the Pearson's Correlation Coefficient Test, which only requires both variables to be numeric in nature.

As the test describes linear relationships between the variables, wind speed can be transformed using polynomials in order to test various non-linear relationships, as shown in Figure 13.



```python
# Create new columns with polynomial features relating to Wind Speed.
cycleWind = cycleWind.withColumn("WindSpeed^2",expr("(WindSpeed * WindSpeed)")) \
                    .withColumn("1/(WindSpeed^2)", expr("1 / (WindSpeed * WindSpeed)")) \
                    .withColumn("WindSpeed^3", expr("WindSpeed * WindSpeed * WindSpeed"))
```

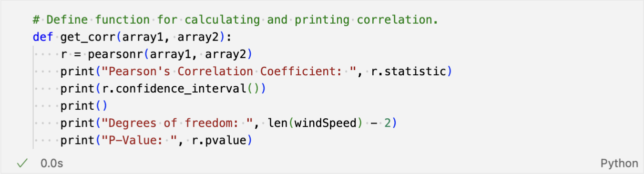
Figure 13: Code for transforming wind speed.

2.3 Results

Using a random sample of 10% of total data points, then running the Pearson's correlation coefficient test where:

- H_0: There is no significant linear relationship between WindSpeed^3 and Duration.
- H_1: There is a significant linear relationship between WindSpeed^3 and Duration.

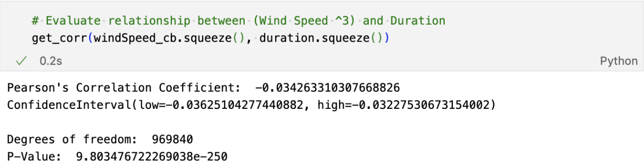
Writing a function to calculate Pearson's correlation coefficient and print the results including P-Value's for determining if results are statistically significant.



```python
# Define function for calculating and printing correlation.
def get_corr(array1, array2):
    r = pearsonr(array1, array2)
    print("Pearson's Correlation Coefficient: ", r.statistic)
    print(r.confidence_interval())
    print()
    print("Degrees of freedom: ", len(windSpeed) - 2)
    print("P-Value: ", r.pvalue)
```

Figure 14: Function to calculate and print Pearson's correlation results.

The result with the highest correlation coefficient is WindSpeed^3, with the results of the statistical test shown in Figure 15.



```python
# Evaluate relationship between (Wind Speed ^3) and Duration
get_corr(windSpeed_cb.squeeze(), duration.squeeze())
```

```
Pearson's Correlation Coefficient:  -0.034263310307668826
ConfidenceInterval(low=-0.03625104277440882, high=-0.03227530673154002)

Degrees of freedom:  969840
P-Value:  9.803476722269038e-250
```

Figure 15: Results for WindSpeed^3 vs Duration.

Since the P-Value is less than 0.05, there is sufficient evidence to reject the null hypothesis, and conclude that there is a linear relationship between WindSpeed^3 and Duration. Though the strength of the relationship is small at approximately -0.034, it is unlikely that this occurs due to chance.

This can be backed up by testing a linear regression model using WindSpeed^3 as the only predictor of Duration, such is shown in Appendix A. The results of this model are shown in Figure 16, where the predictions of WindSpeed^3 are on the left and the actual values on the right.

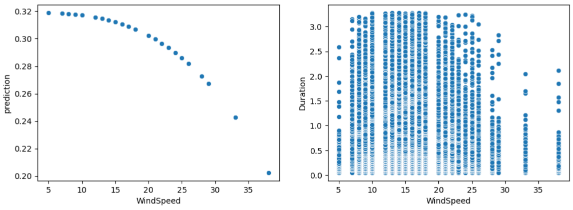


Figure 16: Predictions (left) and Actual (right) of Linear Model.

Although not at first clear, the actual values of duration appear to have higher cycle hire duration's when wind speed is lower, and tend towards a smaller maximum as wind speed increases.

2.4 Conclusion

To conclude, there is a statistically significant relationship between WindSpeed^3 and the Duration of cycle hires.

In Figure 16, cycle hire duration at the lowest wind speeds does not seem to follow the pattern that has been concluded by the Pearson correlation test, future works could look into what might be the reasoning for this.

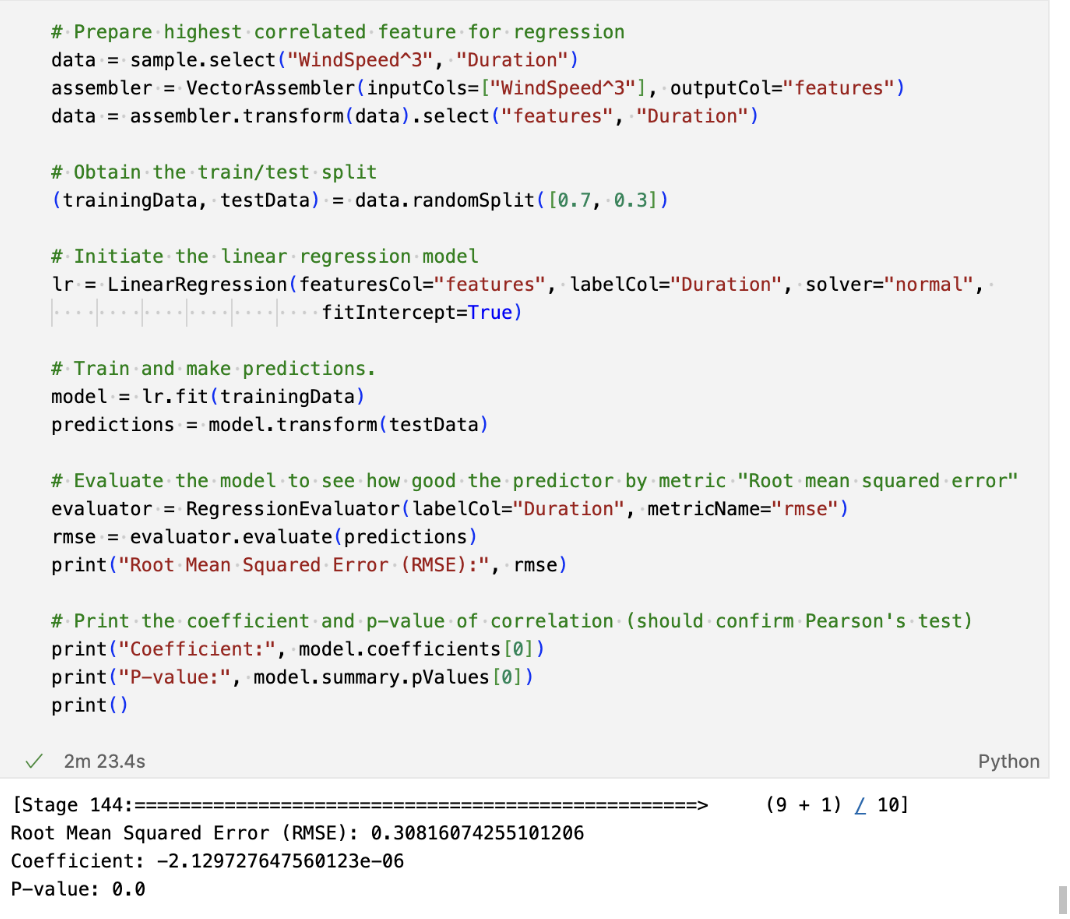
References

London, T. F. (n.d.), ‘cycling.data.tfl.gov.uk’.
URL: *https://cycling.data.tfl.gov.uk/*

Met-Office (n.d.), ‘When does spring start?’.
URL: *https://www.metoffice.gov.uk/weather/learn-about/weather/seasons/spring/when-does-spring-start*

Wunderground (n.d.), ‘London, england, united kingdom weather history’.
URL: *https://www.wunderground.com/history/monthly/gb/london/EGLC/date*

A Linear Model



```
# Prepare highest correlated feature for regression
data = sample.select("WindSpeed^3", "Duration")
assembler = VectorAssembler(inputCols=["WindSpeed^3"], outputCol="features")
data = assembler.transform(data).select("features", "Duration")

# Obtain the train/test split
(trainingData, testData) = data.randomSplit([0.7, 0.3])

# Initiate the linear regression model
lr = LinearRegression(featuresCol="features", labelCol="Duration", solver="normal", 
                        fitIntercept=True)

# Train and make predictions.
model = lr.fit(trainingData)
predictions = model.transform(testData)

# Evaluate the model to see how good the predictor by metric "Root mean squared error"
evaluator = RegressionEvaluator(labelCol="Duration", metricName="rmse")
rmse = evaluator.evaluate(predictions)
print("Root Mean Squared Error (RMSE):", rmse)

# Print the coefficient and p-value of correlation (should confirm Pearson's test)
print("Coefficient:", model.coefficients[0])
print("P-value:", model.summary.pValues[0])
print()
```

```
[Stage 144:=================================================>      (9 + 1) / 10]
Root Mean Squared Error (RMSE): 0.30816074255101206
Coefficient: -2.129727647560123e-06
P-value: 0.0
```