databricks Linear_Regression

```
# File location and type
file_location = "/FileStore/tables/exportcol.csv"
file_type = "csv"

# CSV options
infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

# The applied options are for CSV files. For other file types, these will be ignored.
df = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(file_location)

display(df)
```

product_id	product_description_lenght	product_photos_qty	product_weight_g	product_length_cm
6782d593f63105318f46bbf7633279bf	487	1	200	16
e95ee6822b66ac6058e2e4aff656071a	1153	1	180	17
e9a69340883a438c3f91739d14d3a56d	1912	5	3000	33
036734b5a58d5d4f46b0616ddc047ced	751	5	300	17
b1434a8f79cb3528540d9b21e686e823	184	1	13500	55
d86a6c48f83b045cbba6df84926a1f25	1150	2	3900	45
aa8d88eb4b9cb38894e33fa624c4287f	335	4	250	16
aa6746e94490239d3d9ee6ab89779aba	90	1	500	16

Showing the first 1000 rows.

```
# Create a view or table

temp_table_name = "linreg_csv"

df.createOrReplaceTempView(temp_table_name)
```

```
%sql

/* Query the created temp table in a SQL cell */

select * from `linreg_csv`
```

product_id	product_description_lenght	product_photos_qty	product_weight_g	product_length_cm
6782d593f63105318f46bbf7633279bf	487	1	200	16
e95ee6822b66ac6058e2e4aff656071a	1153	1	180	17
e9a69340883a438c3f91739d14d3a56d	1912	5	3000	33
036734b5a58d5d4f46b0616ddc047ced	751	5	300	17
b1434a8f79cb3528540d9b21e686e823	184	1	13500	55
d86a6c48f83b045cbba6df84926a1f25	1150	2	3900	45
aa8d88eb4b9cb38894e33fa624c4287f	335	4	250	16
aa6746e94490239d3d9ee6ab89779aba	90	1	500	16
5ea739ddd646d1ba53eea4e3c099a953	892	2	1200	52

Showing the first 1000 rows.

```
# With this registered as a temp view, it will only be available to this particular notebook. If you'd like other users to be able to query this table, you can also create a table from the DataFrame.
# Once saved, this table will persist across cluster restarts as well as allow various users across different notebooks to query this data.
# To do so, choose your table name and uncomment the bottom line.

permanent_table_name = "linreg_csv"

# df.write.format("parquet").saveAsTable(permanent_table_name)
```

```
dfLinReg = sqlContext.sql("select product_description_lenght, product_photos_qty, product_weight_g, product_length_cm, product_height_cm, product_width_cm, freight_value from linreg_csv")
display(dfLinReg)
```

product_description_lenght	product_photos_qty	product_weight_g	product_length_cm	product_he
487	1	200	16	14
1153	1	180	17	11
1912	5	3000	33	12
751	5	300	17	4
184	1	13500	55	25
1150	2	3900	45	33
335	4	250	16	2
90	1	500	16	35
892	2	1200	52	13

Showing the first 1000 rows.

```
dfLinReg = dfLinReg.dropna(how='any')
```

```
Out[19]: 0
```

```
from pyspark.sql.types import IntegerType

dfLinReg1 = dfLinReg.withColumn("freight_value", dfLinReg["freight_value"].cast(IntegerType()))
```

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

dfAssemblerFeature =  VectorAssembler(
                inputCols=["product_description_lenght", "product_photos_qty", "product_weight_g", "product_length_cm",
"product_height_cm", "product_width_cm"],
                outputCol="features")

transformed2 = dfAssemblerFeature.transform(dfLinReg1)
# dfLinReg.show()
```

```
(training, test) = transformed2.randomSplit([0.8, 0.2])
```

```
# training.show()
```

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

```
lr = LinearRegression( maxIter=10, regParam=0.3, elasticNetParam=0.8, featuresCol="features", labelCol="freight_value")
# Fit the model
lrModel = lr.fit(training)

# Print the coefficients and intercept for linear regression
print("Coefficients: %s" % str(lrModel.coefficients))
print("Intercept: %s" % str(lrModel.intercept))
```

```
Coefficients: [0.00107671160962,0.0,0.00230846925771,0.0256164301797,0.0562146962413,0.0123737770178]
Intercept: 11.899274513653877
```

```
dfRatingCount = lrModel.transform(test)
dfRatingCount.show(10)
```

```
+--------------------------+------------------+----------------+------------------+------------------+-----------------+-------------+--------------------+------------------+
|product_description_lenght|product_photos_qty|product_weight_g|product_length_cm|product_height_cm|product_width_cm|freight_value|            features|        prediction|
+--------------------------+------------------+----------------+------------------+------------------+-----------------+-------------+--------------------+------------------+
|                        23|                 1|             200|                16|                 2|               11|           15|[23.0,1.0,200.0,1...|13.044136554769029|
|                        27|                 1|             300|                21|                11|               13|            8|[27.0,1.0,300.0,2...|13.938052298083473|
|                        27|                 1|             300|                21|                11|               13|            8|[27.0,1.0,300.0,2...|13.938052298083473|
|                        27|                 1|             300|                21|                11|               13|           15|[27.0,1.0,300.0,2...|13.938052298083473|
|                        30|                 1|            1400|                16|                33|               18|           17|[30.0,1.0,1400.0,...| 17.65110866788902|
|                        30|                 1|            1400|                16|                33|               18|           17|[30.0,1.0,1400.0,...| 17.65110866788902|
|                        33|                 2|             400|                20|                20|               20|           15|[33.0,2.0,400.0,2...| 14.74229176862804|
|                        35|                 1|             300|                16|                 6|               16|            7|[35.0,1.0,300.0,1...|13.574631689909095|
|                        40|                 1|             562|                20|                13|               17|           11|[40.0,1.0,562.0,2...|14.693176564901815|
|                        41|                 1|             150|                20|                 3|               18|            7|[41.0,1.0,150.0,2...|13.193390756941028|
+--------------------------+------------------+----------------+------------------+------------------+-----------------+-------------+--------------------+------------------+
only showing top 10 rows
```

```
# Summarize the model over the training set and print out some metrics
trainingSummary = lrModel.summary
print("RMSE: %f" % trainingSummary.rootMeanSquaredError)
print("r2: %f" % trainingSummary.r2)
```

```
RMSE: 13.343608
r2: 0.353158
```

So from the predictions and rsme value we can see that the accuracy of the model is very bad.To increase the predictions of the model further analysis can be done through random forests.