Regression Model

Import Data

```
# File location and type
file_location = "/FileStore/tables/GOOGLE.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)

display(df.select("Adj Close").summary())
```

We re use the **exponential moving average** (EMA) to have our true values

```
def exponential_moving_average(df, column_name, period, smooth=2):
    prices = df.select(column_name).rdd.flatMap(lambda x: x).collect()
    ema = [sum(prices[:period]) / period]
    for price in prices[period:]:
        ema.append((price * (smooth / (1 + period))) + ema[-1] * (1 -
(smooth / (1 + period))))

    return ema
```

We create a new DataFrame containing only EMA for Closing price

```
from pyspark.sql.types import DoubleType
df = df.drop('High','Open','Low','Adj Close','Volume','company_name')
new_df =
spark.createDataFrame(exponential_moving_average(df,df['Close'],10),
DoubleType())
new_df = new_df.withColumnRenamed("value",'EMA')
columns = ['EMA']
vals = [0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0]
nr = spark.createDataFrame(vals, DoubleType())
nr = nr.withColumnRenamed("value",'EMA')
```

```
new_df = nr.union(new_df)
display(new_df)
```

We add an ID column

```
from pyspark.sql.functions import monotonically_increasing_id

DF1 = df.withColumn("row_id", monotonically_increasing_id())
display(DF1)
```

Finally, our dataset for the regression model is a DataFrame containing for each value in the Close price column its EMA calculated

```
from pyspark.sql.functions import col

DF2 = new_df.rdd.zipWithIndex().toDF()
DF2 = DF2.select(col("_1.*"),col("_2").alias('row_id'))
display(DF2)
result_df = DF1.join(DF2, ("row_id")).drop("row_id")
display(result_df)
```

```
final_df = spark.createDataFrame(result_df.tail(df.count()-9),
result_df.schema)
display(final_df)
```

We split this DataFrame to obtain a train and a test DataFrame

```
trainDF, testDF = final_df.randomSplit([.8, .2], seed=42)
trainDF.count()
```

```
Out[8]: 782
```

We use the MLlib form Pyspark to create a regression model and use it on our DataFrame

```
from pyspark.sql.functions import col, log, exp
from pyspark.ml import Pipeline
from pyspark.ml.feature import RFormula
from pyspark.ml.regression import LinearRegression
from pyspark.ml.evaluation import RegressionEvaluator

logTrainDF = trainDF.withColumn("log_EMA", log(col("EMA")))
logTestDF = testDF.withColumn("log_EMA", log(col("EMA")))

rFormula = RFormula(formula="log_EMA ~ . - EMA",
featuresCol="features", labelCol="log_EMA", handleInvalid="skip")

lr = LinearRegression(labelCol="log_EMA", predictionCol="log_pred")

pipeline = Pipeline(stages = [rFormula, lr])
pipelineModel = pipeline.fit(logTrainDF)
predDF = pipelineModel.transform(logTestDF)
```

```
expDF = predDF.withColumn("prediction", exp(col("log_pred")))
display(expDF)
```

At the end, we evalue our results.

```
regressionEvaluator = RegressionEvaluator(labelCol="EMA",
predictionCol="prediction")
rmse = regressionEvaluator.setMetricName("rmse").evaluate(expDF)
r2 = regressionEvaluator.setMetricName("r2").evaluate(expDF)
print(f"RMSE is {rmse}")
print(f"R2 is {r2}")
```

```
RMSE is 39.44250331548785
R2 is 0.9623560385420282
```