2

FINISHED

```
//Using just one month data as a test to implement Machine learning prediction model
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._
import org.apache.spark.sql.expressions.Window

val spark=SparkSession.builder().appName("Spark dataframe eaxample").config("spark.some.config.option"
import spark.implicits._
var data=spark.read.option("sep",",").option("header","true").csv("C:/BigData/~data/FP-Data-2020/Drive
data.write.parquet("C:/BigData/~data/projectp.parquet")

val parquetProject = spark.read.parquet("C:/BigData/~data/projectp.parquet")
```

```
import org.apache.spark.sql.SparkSession
import org.apache.spark.sql.functions._
import org.apache.spark.sql.expressions.Window
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@2f8cae9b
import spark.implicits._
data: org.apache.spark.sql.DataFrame = [date: string, serial_number: string ... 129 more fields]
parquetProject: org.apache.spark.sql.DataFrame = [date: string, serial_number: string ... 129 more fie
lds]
```

Took 2 min 37 sec. Last updated by admin at May 04 2021, 7:54:16 AM.

FINISHED

```
import org.apache.spark.ml.linalg.Vectors
val data_useful = parquetProject.select(
    col("date"),
    col("serial_number"),
    col("model"),
    col("capacity_bytes"),
    col("failure"),
    col("smart_1_raw"),col("smart_5_raw"),col("smart_9_raw"),col("smart_187_raw"),
    col("smart_188_raw"),col("smart_194_raw"),col("smart_197_raw"),
    col("smart_198_raw"))
data_useful.show(5)
```

```
+----------+--------------+--------------------+--------------+-------+-----------+-----------+-----
------+-------------+-------------+-------------+-------------+-------------+
|      date| serial_number|               model|capacity_bytes|failure|smart_1_raw|smart_5_raw|smart
_9_raw|smart_187_raw|smart_188_raw|smart_194_raw|smart_197_raw|smart_198_raw|
+----------+--------------+--------------------+--------------+-------+-----------+-----------+-----
------+-------------+-------------+-------------+-------------+-------------+
|2020-06-01|      Z305B2QN|         ST4000DM000| 4000787030016|      0|175611984.0|        0.0|    3
9114.0|          0.0|          0.0|         22.0|          0.0|          0.0|
|2020-06-01|      ZJV0XJQ4|       ST12000NM0007|12000138625024|      0| 45037632.0|        0.0|    1
6130.0|          0.0|          0.0|         29.0|          0.0|          0.0|
|2020-06-01|      ZJV0XJQ3|       ST12000NM0007|12000138625024|      0|204656400.0|        0.0|    1
3186.0|          0.0|          0.0|         34.0|          0.0|          0.0|
|2020-06-01|      ZJV0XJQ0|       ST12000NM0007|12000138625024|      0| 24304936.0|        0.0|    1
6757.0|          0.0|          0.0|         26.0|          0.0|          0.0|
|2020-06-01|PL1331LAHG1S4H|HGST HMS5C4040ALE640| 4000787030016|      0|        0.0|        0.0|    2
9186.0|         null|         null|         29.0|          0.0|          0.0|
+----------+--------------+--------------------+--------------+-------+-----------+-----------+-----
------+-------------+-------------+-------------+-------------+-------------+
```

Took 0 sec. Last updated by admin at May 04 2021, 7:56:58 AM.

```
data_useful.printSchema
```

FINISHED

```
root
 |-- date: string (nullable = true)
 |-- serial_number: string (nullable = true)
 |-- model: string (nullable = true)
 |-- capacity_bytes: string (nullable = true)
 |-- failure: string (nullable = true)
 |-- smart_1_raw: string (nullable = true)
 |-- smart_5_raw: string (nullable = true)
 |-- smart_9_raw: string (nullable = true)
 |-- smart_187_raw: string (nullable = true)
 |-- smart_188_raw: string (nullable = true)
 |-- smart_194_raw: string (nullable = true)
 |-- smart_197_raw: string (nullable = true)
 |-- smart_198_raw: string (nullable = true)
```

Took 0 sec. Last updated by admin at May 04 2021, 7:57:02 AM.

FINISHED

```
val cast_df = data_useful.select(data_useful.columns.map {
    case column@"failure" =>
      col(column).cast("Integer").as(column)
    case column@"smart_1_raw" =>
      col(column).cast("Double").as(column)
    case column@"smart_5_raw" =>
      col(column).cast("Double").as(column)
    case column@"smart_9_raw" =>
      col(column).cast("Double").as(column)
    case column@"smart_187_raw" =>
      col(column).cast("Double").as(column)
    case column@"smart_188_raw" =>
      col(column).cast("Double").as(column)
    case column@"smart_194_raw" =>
      col(column).cast("Double").as(column)
    case column@"smart_197_raw" =>
      col(column).cast("Double").as(column)
    case column@"smart_198_raw" =>
      col(column).cast("Double").as(column)
    case column =>
      col(column)
  }: _*)

cast_df.printSchema()
```

```
root
 |-- date: string (nullable = true)
 |-- serial_number: string (nullable = true)
 |-- model: string (nullable = true)
 |-- capacity_bytes: string (nullable = true)
 |-- failure: integer (nullable = true)
 |-- smart_1_raw: double (nullable = true)
 |-- smart_5_raw: double (nullable = true)
 |-- smart_9_raw: double (nullable = true)
 |-- smart_187_raw: double (nullable = true)
 |-- smart_188_raw: double (nullable = true)
 |-- smart_194_raw: double (nullable = true)
 |-- smart_197_raw: double (nullable = true)
 |-- smart_198_raw: double (nullable = true)
```

```
cast_df: org.apache.spark.sql.DataFrame = [date: string, serial_number: string ... 11 more fields]
```

Took 2 sec. Last updated by admin at May 04 2021, 7:57:08 AM.

2

```
//handling null values
val df= cast_df.na.fill(0)
```

FINISHED

```
df: org.apache.spark.sql.DataFrame = [date: string, serial_number: string ... 11 more fields]
```

Took 0 sec. Last updated by admin at May 04 2021, 7:57:13 AM.

```
df.createOrReplaceTempView("check")
```

FINISHED

Took 0 sec. Last updated by admin at May 03 2021, 2:20:26 PM.

```
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.regression.{LinearRegression, LinearRegressionModel}
```

FINISHED

```
import org.apache.spark.ml.Pipeline
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.regression.{LinearRegression, LinearRegressionModel}
```

Took 0 sec. Last updated by admin at May 04 2021, 3:18:30 PM.

```
// Defining features
val features = new VectorAssembler()
  .setInputCols(Array("smart_5_raw","smart_187_raw","smart_188_raw","smart_197_raw","smart_198_raw"))
  .setOutputCol("features")
```

FINISHED

```
features: org.apache.spark.ml.feature.VectorAssembler = vecAssembler_a6f21a8a0e3e
```

Took 1 sec. Last updated by admin at May 04 2021, 3:18:37 PM.

```
// Define model to use
val lr = new LinearRegression().setLabelCol("failure")
```

FINISHED

```
lr: org.apache.spark.ml.regression.LinearRegression = linReg_9d6c8d359912
```

Took 0 sec. Last updated by admin at May 04 2021, 3:18:40 PM.

```
// Create a pipeline that associates the model with the data processing sequence
val pipeline = new Pipeline().setStages(Array(features, lr))
```

FINISHED

```
pipeline: org.apache.spark.ml.Pipeline = pipeline_4a1573d5e07f
```

Took 0 sec. Last updated by admin at May 04 2021, 8:05:09 AM.

```
// Run the Model
val model = pipeline.fit(df)
```

FINISHED

```
model: org.apache.spark.ml.PipelineModel = pipeline_4a1573d5e07f
```

Took 4 min 12 sec. Last updated by admin at May 04 2021, 8:09:25 AM.

```
val linRegModel = model.stages(1).asInstanceOf[LinearRegressionModel]
```

FINISHED

```
linRegModel: org.apache.spark.ml.regression.LinearRegressionModel = linReg_1a59360b44cf
```

Took 0 sec. Last updated by admin at May 04 2021, 8:09:28 AM.

```
println(s"RMSE:  ${linRegModel.summary.rootMeanSquaredError}")
println(s"r2:    ${linRegModel.summary.r2}")
println(s"Model: Y = ${linRegModel.coefficients(3)} * X + ${linRegModel.intercept}")
```

FINISHED

```
RMSE:  0.004458851698142961
r2:    0.001506107215420438
Model: Y = 1.2228264550008065E-5 * X + 1.1841480163685294E-5
```

Took 0 sec. Last updated by admin at May 04 2021, 8:09:32 AM.

```
linRegModel.summary.residuals.show()
```

FINISHED

```
+--------------------+
|           residuals|
+--------------------+
|-1.18414801636852...|
|-1.18414801636852...|
|-1.18414801636852...|
|-1.18414801636852...|
|-1.18414801636852...|
|-1.18414801636852...|
|-1.18414801636852...|
|-1.18414801636852...|
|-9.38136879710314...|
|-1.18414801636852...|
|-1.18414801636852...|
|-1.18414801636852...|
|-1.18414801636852...|
|-1.18414801636852...|
|-1.18414801636852...|
```

Took 1 sec. Last updated by admin at May 04 2021, 8:09:39 AM.

```
val result = model.transform(df).select("smart_5_raw","smart_187_raw","smart_188_raw","smart_197_raw",
result.show(1000)
```

FINISHED

```
+-----------+-------------+---------------+-------------+-------+--------------------+--------------
------+
|smart_5_raw|smart_187_raw|  smart_188_raw|smart_197_raw|failure|          prediction|            fe
atures|
+-----------+-------------+---------------+-------------+-------+--------------------+--------------
------+
|        0.0|          0.0|            0.0|          0.0|      0|1.184148016368529...|          (5,
[],[])|
|        0.0|          0.0|            0.0|          0.0|      0|1.184148016368529...|          (5,
[],[])|
|        0.0|          0.0|            0.0|          0.0|      0|1.184148016368529...|          (5,
```

```
[],[])|
|       0.0|          0.0|           0.0|          0.0|      0|1.184148016368529...|          (5,
[],[])|
|       0.0|          0.0|           0.0|          0.0|      0|1.184148016368529...|          (5,
[],[])|
|       0.0|          0.0|           0.0|          0.0|      0|1.184148016368529...|          (5,
```

2

Output is truncated to 102400 bytes. Learn more about **ZEPPELIN_INTERPRETER_OUTPUT_LIMIT**

Took 1 sec. Last updated by admin at May 04 2021, 8:10:11 AM.

READY