

ADVANCED DBMS - TAKE AWAY CAT

DATE: 28.11.2023

GROUP MEMBERS

NICKSON KIPROTICH - SCT222-0037/2016

ELVIS - SCT222-0307/2020

BALDWIN - SCT222-0164/2020

KIMUTAI KELVIN - 0149/2019

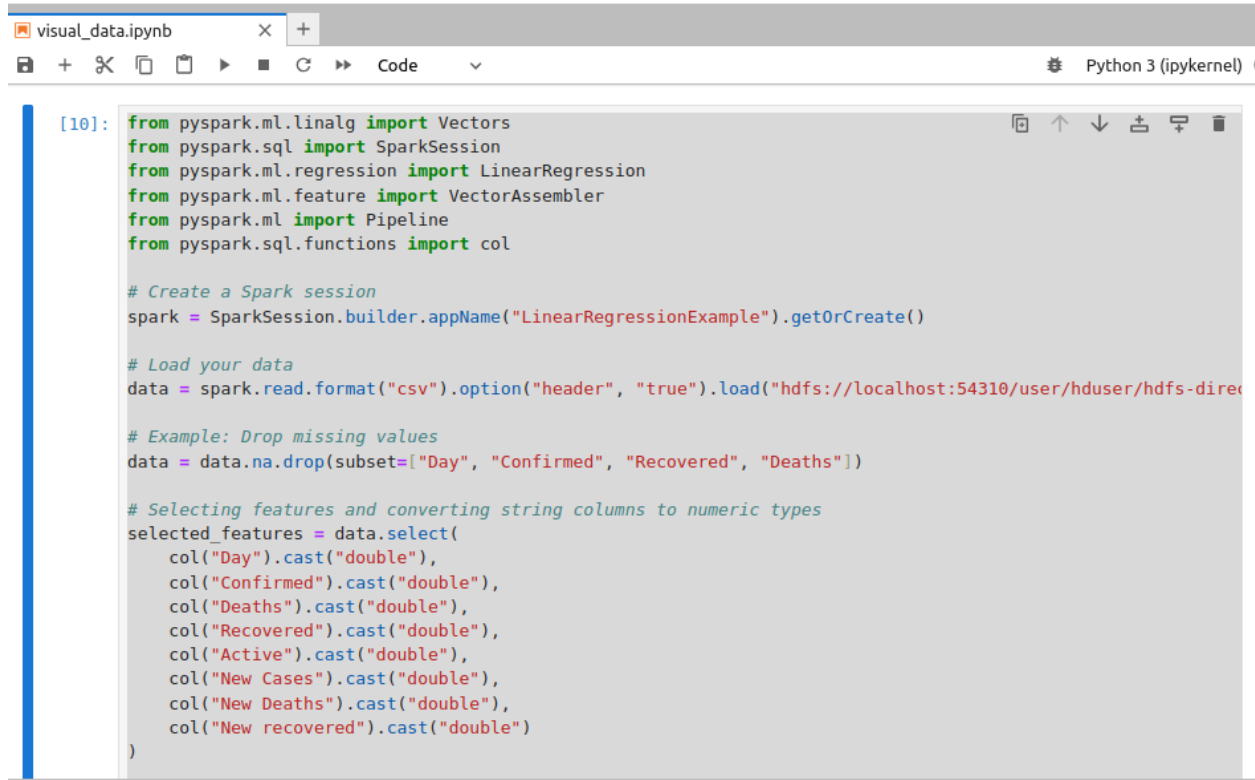
Ingest the data to hadoop.

```
hduser@rono-HP-15:~$ hdfs dfs -copyFromLocal /home/rono/Documents/BBC/DBMS/daily_file.csv /user/hduser/hdfs-directory/  
hduser@rono-HP-15:~$
```

- Added /user prefix property to \$HADOOP_HOME/etc/hadoop/core-site.xml
- Create directory /user/hduser
- `hdfs dfs -copyFromLocal /path/to/csv /user/hduser/hdfs-directory/`

The command is copying a file from the local file system to HDFS. Once the file is in HDFS, it becomes part of the Hadoop ecosystem and can be utilized by various distributed computing frameworks, such as Apache Spark (as demonstrated in our PySpark code) or MapReduce, for further processing and analysis.

1. Use pyspark package to extract the data from the data lake



```
[10]: from pyspark.ml.linalg import Vectors
from pyspark.sql import SparkSession
from pyspark.ml.regression import LinearRegression
from pyspark.ml.feature import VectorAssembler
from pyspark.ml import Pipeline
from pyspark.sql.functions import col

# Create a Spark session
spark = SparkSession.builder.appName("LinearRegressionExample").getOrCreate()

# Load your data
data = spark.read.format("csv").option("header", "true").load("hdfs://localhost:54310/user/hduser/hdfs-directo

# Example: Drop missing values
data = data.na.drop(subset=["Day", "Confirmed", "Recovered", "Deaths"])

# Selecting features and converting string columns to numeric types
selected_features = data.select(
    col("Day").cast("double"),
    col("Confirmed").cast("double"),
    col("Deaths").cast("double"),
    col("Recovered").cast("double"),
    col("Active").cast("double"),
    col("New Cases").cast("double"),
    col("New Deaths").cast("double"),
    col("New recovered").cast("double")
)
```

2. Choose appropriate techniques to Pre- process the extracted data

Dropped rows with empty values

```
# Data preprocessing (function) subset: Any
data = data.na.drop(subset=["Day", "Confirmed", "Recovered", "Deaths"])
```

Rows with missing values in columns "Day," "Confirmed," "Recovered," and "Deaths" are dropped.

Feature engineering

```
28
29 # Assemble features into a single vector column and overwrite the existing column
30 assembler = VectorAssembler(inputCols=["Day", "Confirmed", "Recovered", "Deaths"], outputCol="features")
31 data_assembled = assembler.transform(selected_features).select("Day", "Confirmed", "Recovered",
32 "Deaths", "features")
```

Added a new column "features" containing a vector of input features (Day, Confirmed, Recovered) for the linear regression model. Placing the data in vector format is good practice for preparing features for machine learning.

Splitting the data into training and test data

```
32
33 # Split the data into training and test sets
34 (training_data, test_data) = data_assembled.randomSplit([0.8, 0.2], seed=123)
35
```

3. Apply one predictive analytics technique to generate a model for predicting any of the following cases:

- a) Number of Death cases or Mortality rate**
- b) Number of confirmed cases**
- c) Number of recovery cases or Recovery rate**

Technique: We chose to apply Linear Regression

Justification: We chose Linear Regression due to its simplicity, interpretability, and effectiveness in predicting numeric values.

```

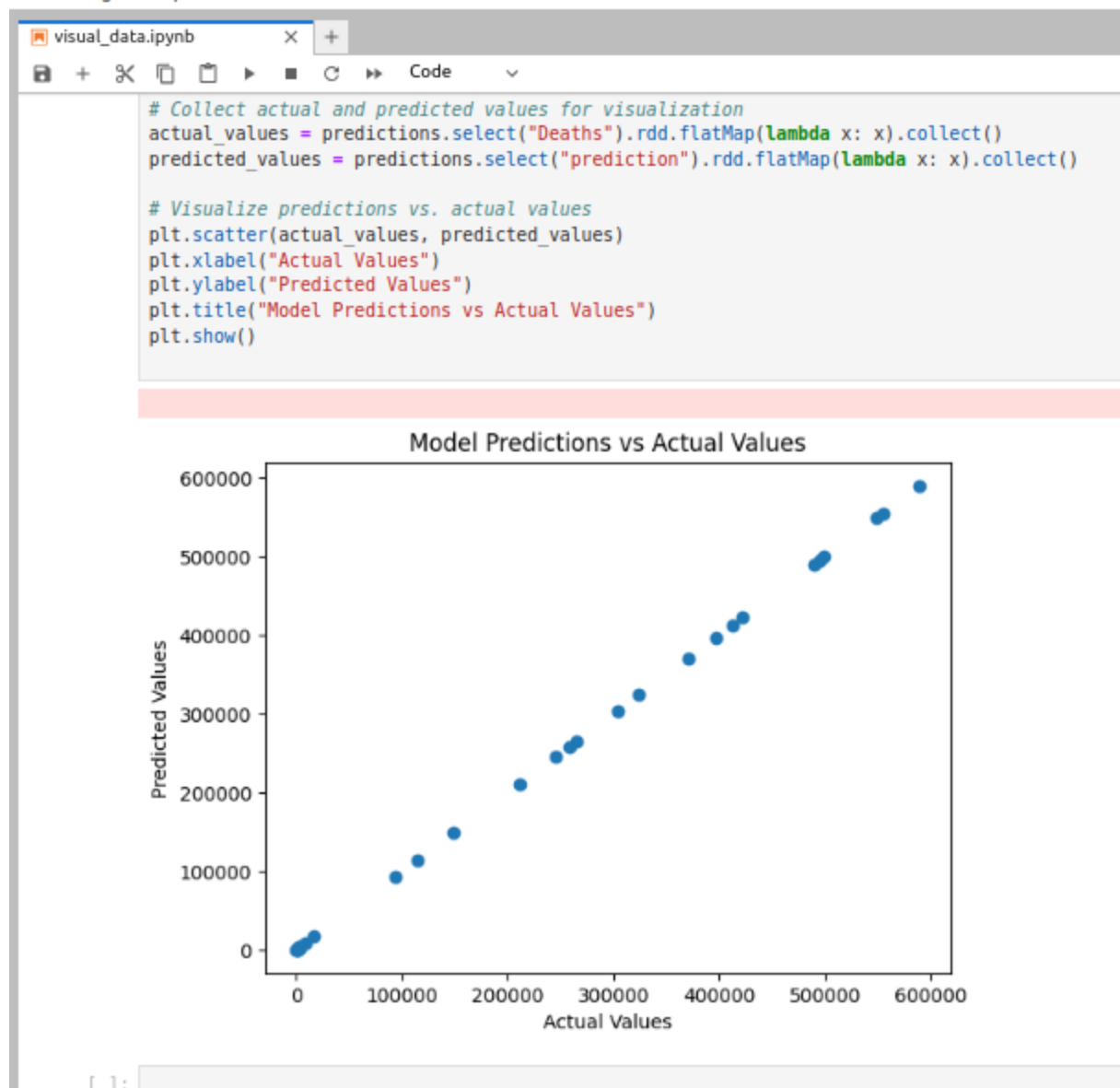
36 # Create a linear regression model
37 lr = LinearRegression(featuresCol="features", labelCol="Deaths")
38
39 # Create a pipeline
40 pipeline = Pipeline(stages=[lr])
41
42 # Train the model
43 model = pipeline.fit(training_data)
44
45 # Make predictions on the test data
46 predictions = model.transform(test_data)
47
48 # Display predictions vs. actual values
49 predictions.select("Deaths", "prediction").show()
50
51

```

Do you want to
extension

Linear regression is applied to predict the number of deaths based on features such as "Death," "Confirmed," and "Recovered." The chosen features are assembled into a vector column named "features" using the VectorAssembler, and the model is trained on the training dataset. Linear regression is particularly well-suited for this task as it is a suitable technique for predicting numeric values, such as the number of deaths in this case. The resulting model is then used to make predictions on the test dataset.

4. Visualize the model



5. Test the model

```
visual_data.ipynb x +
+ + ✂ 📄 ▶ ■ ↺ ⏩ Code v
23/11/30 21:49:03 WARN Instrumentation: [73f9c
y and overfitting.

+-----+-----+
| Deaths| prediction|
+-----+-----+
| 26.0|25.99999999978794|
| 131.0| 130.999999999559|
| 426.0| 425.999999999429|
| 492.0|491.9999999994554|
| 906.0| 905.999999999379|
| 1523.0|1522.999999999527|
| 2246.0| 2245.999999999869|
| 2250.0|2249.9999999998595|
| 2627.0| 2626.999999999806|
| 2707.0| 2706.999999999787|
| 2767.0|2766.9999999997694|
| 2936.0|2935.9999999997162|
| 3079.0| 3078.999999999686|
| 3981.0|3980.9999999996126|
| 7948.0| 7947.999999999662|
| 16748.0|16747.99999999956|
| 93650.0| 93650.0000000007|
|114620.0|114620.00000000067|
|148591.0|148591.00000000058|
|210862.0|210862.00000000038|
+-----+-----+
only showing top 20 rows
```

6. Validate the model

The result table displays the actual number of deaths (Deaths) alongside the corresponding predictions (prediction) generated by the linear regression model. Upon initial inspection with emphasis on the first 20 rows, it appears that the model has produced predictions that closely align with the actual death counts. The predicted values exhibit a high level of precision, appearing to be very close to the actual values. The closeness of the predictions to the actual values suggests that the linear regression model is performing well on the test dataset.

Potential Applications of the Interpreted Results:

The interpreted results, if validated and deemed reliable, hold significant potential applications. In the context of predicting the number of death cases based on features like "Deaths," "Confirmed," and "Recovered," the model could be employed for:

Early Warning Systems: Predicting potential spikes in death cases to enable timely responses.

Resource Allocation: Assisting health authorities in allocating resources such as medical personnel and supplies based on predicted mortality rates.

Policy Decision Support: Providing insights for policymakers to make informed decisions regarding public health interventions.

Github link: https://github.com/onornick/DBMS/blob/main/visual_data.ipynb