CS4830 – Big Data Laboratory

Assignment 6-Lab 9

Ishan Chokshi – BE19B018

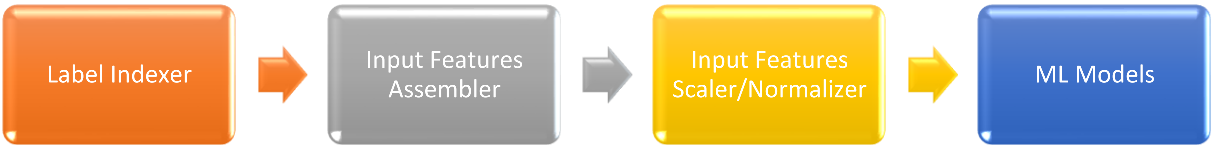
January-May 2023

Indian Institute of Technology Madras

Ishan Chokshi
BE19B018

1. **Download the dataset and upload it into your bucket.**
   Train an ML model on the dataset to predict the Outcome and report the accuracy for different preprocessing techniques and models. Provide the details of data exploration and feature engineering steps. You also need to submit the code along with the answers to the above questions.

   ML Model pipeline:



Two types of scaling techniques were used: Normalizer and Standard Scaler. **I have attached outputs (while submitting the job) only for Normalizer and the code submitted contains output for both types of scaling techniques**. The metric selected for model performance was accuracy. Summary of model outputs is shown in the table below:

Scaling Technique	ML Model	Accuracy
Normalizer	Logistic Regression	61.18%
	Decision Tree Classifier	64.47%
	Random Forest Classifier	65.79%
Standard Scaler	Logistic Regression	79.33%
	Decision Tree Classifier	71.33%
	Random Forest Classifier	78.67%

We notice that Standard Scaler technique gives a better accuracy than the Normalizer.
We know that in cases where meaningful information is found in the relationship between feature values from one sample to another sample, Standard Scaler and other scalers that work feature wise are preferred. In contrast, Normalizer and other scalers that work sample wise are preferred in cases where meaningful information is found in the relationship between feature values from one feature to another feature. So we can conclude that there is some meaningful information in the relationship between feature values from one sample to another. Screenshots are attached below:

Ishan Chokshi
BE19B018

[16]:
```
df.show()
```

```
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
|Pregnancies|Glucose|BloodPressure|SkinThickness|Insulin| BMI|DiabetesPedigreeFunction|Age|Outcome|
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
|          6|  148.0|           72|         35.0|      0|33.6|                       0| 50|      1|
|          1|   85.0|           66|         29.0|      0|26.6|                       0| 31|      0|
|          8|  183.0|           64|          0.0|      0|23.3|                       0| 32|      1|
|          1|   89.0|           66|         23.0|     94|28.1|                       0| 21|      0|
|          0|  137.0|           40|         35.0|    168|43.1|                       2| 33|      1|
|          5|  116.0|           74|          0.0|      0|25.6|                       0| 30|      0|
|          3|   78.0|           50|         32.0|     88|31.0|                       0| 26|      1|
|         10|  115.0|            0|          0.0|      0|35.3|                       0| 29|      0|
|          2|  197.0|           70|         45.0|    543|30.5|                       0| 53|      1|
|          8|  125.0|           96|          0.0|      0| 0.0|                       0| 54|      1|
|          4|  110.0|           92|          0.0|      0|37.6|                       0| 30|      0|
|         10|  168.0|           74|          0.0|      0|38.0|                       0| 34|      1|
|         10|  139.0|           80|          0.0|      0|27.1|                       1| 57|      0|
|          1|  189.0|           60|         23.0|    846|30.1|                       0| 59|      1|
|          5|  166.0|           72|         19.0|    175|25.8|                       0| 51|      1|
|          7|  100.0|            0|          0.0|      0|30.0|                       0| 32|      1|
|          0|  118.0|           84|         47.0|    230|45.8|                       0| 31|      1|
|          7|  107.0|           74|          0.0|      0|29.6|                       0| 31|      1|
|          1|  103.0|           30|         38.0|     83|43.3|                       0| 33|      0|
|          1|  115.0|           70|         30.0|     96|34.6|                       0| 32|      1|
+-----------+-------+-------------+-------------+-------+----+------------------------+---+-------+
only showing top 20 rows
```

[28]:
```
predictions = model.transform(testData)
evaluator = MulticlassClassificationEvaluator(labelCol="indexedLabel", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy for logistic regression  = %g" % (accuracy))
```

```
Test set accuracy for logistic regression  = 0.611842
```

[24]:
```
pipeline_rf = Pipeline(stages=[labelIndexer, assembler,scaler, rf])
pipeline_dtc = Pipeline(stages=[labelIndexer, assembler,scaler, dtc])
```

[25]:
```
model_rf = pipeline_rf.fit(trainingData)
model_dtc = pipeline_dtc.fit(trainingData)
```

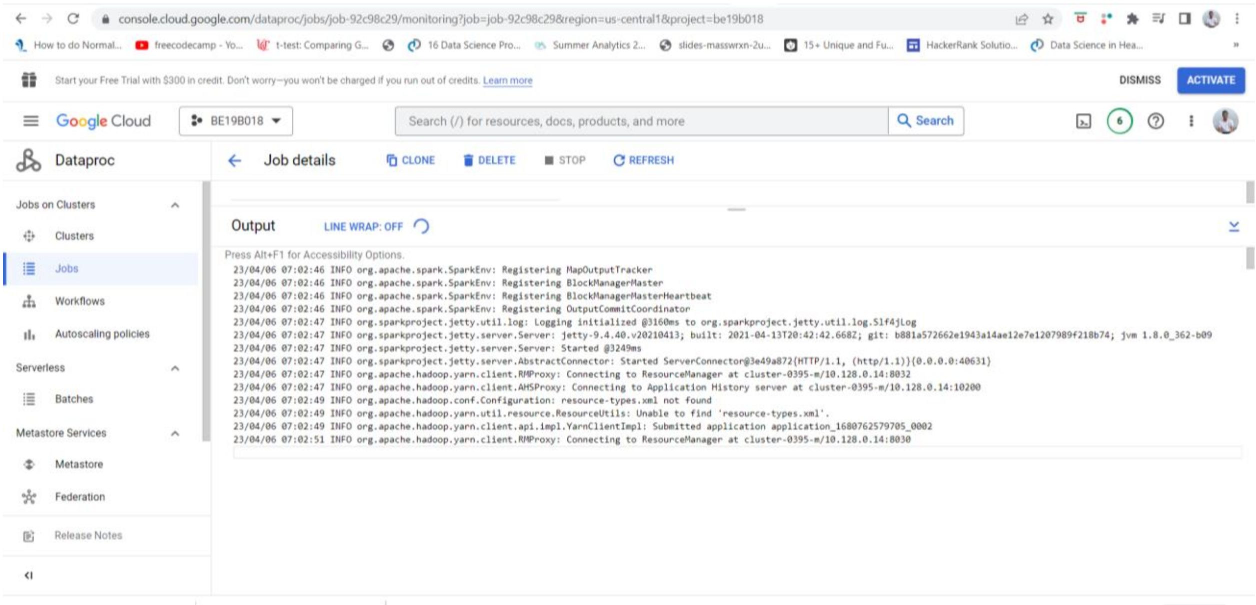
[26]:
```
predictions = model_rf.transform(testData)
evaluator = MulticlassClassificationEvaluator(labelCol="indexedLabel", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy for Random Forest Classifier  = %g" % (accuracy))
```

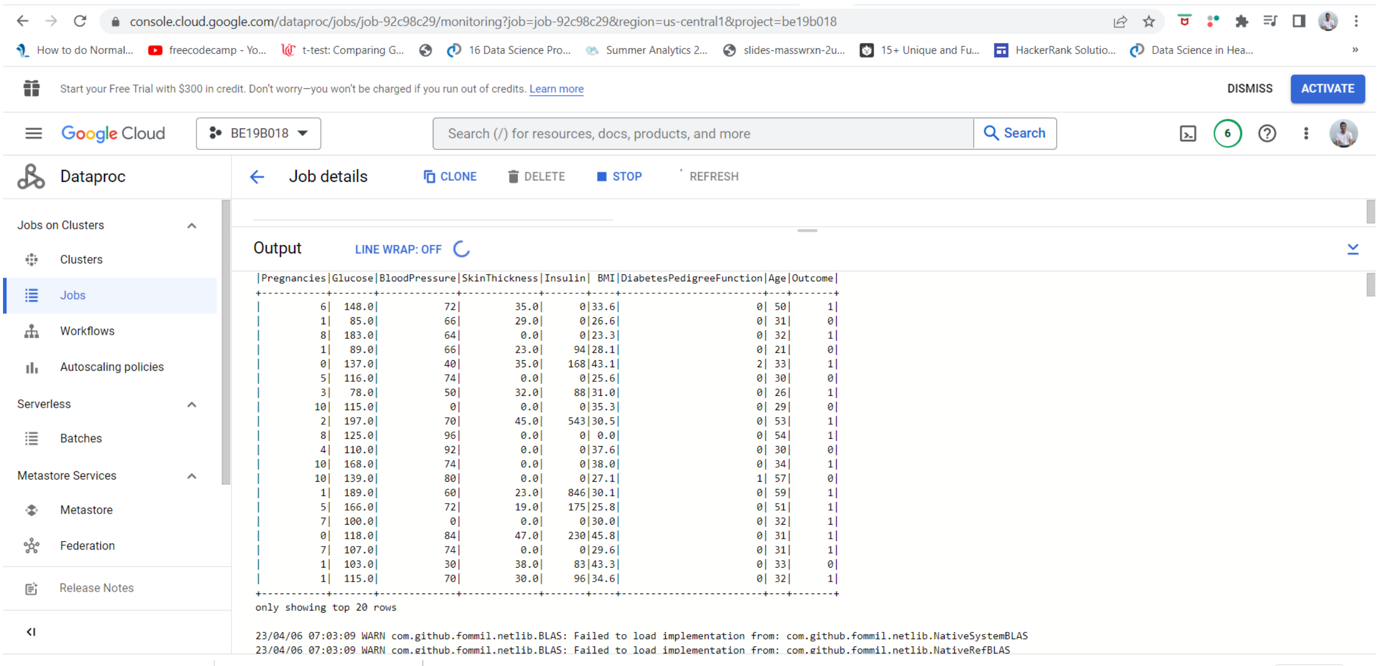
```
Test set accuracy for Random Forest Classifier  = 0.657895
```

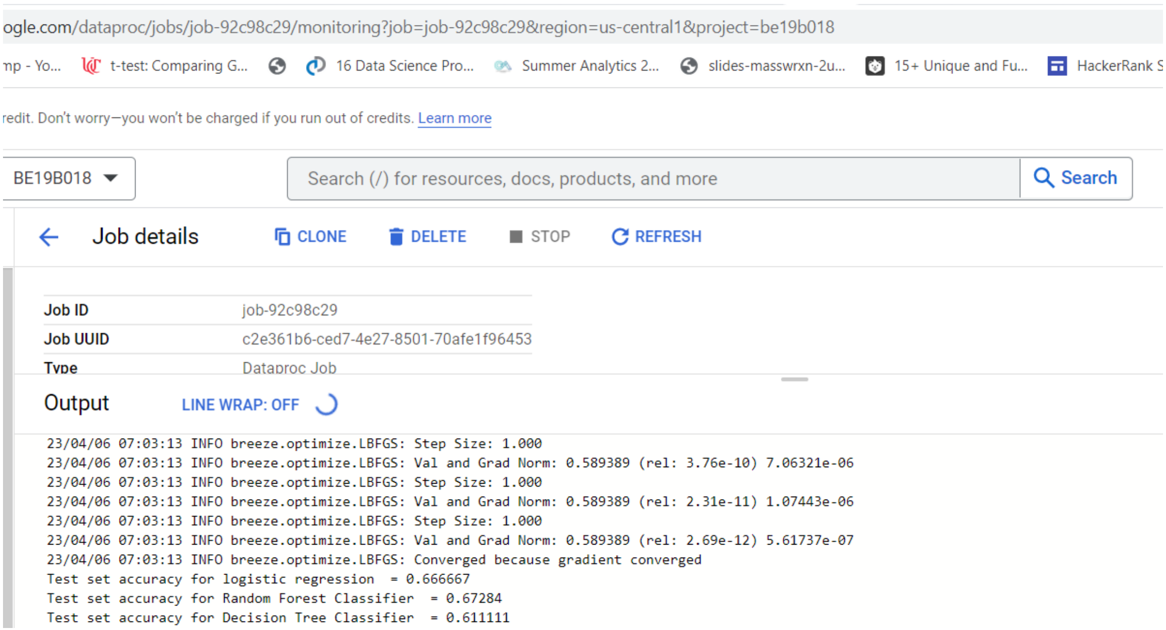
[27]:
```
predictions = model_dtc.transform(testData)
evaluator = MulticlassClassificationEvaluator(labelCol="indexedLabel", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Test set accuracy for Decision Tree Classifier  = %g" % (accuracy))
```

```
Test set accuracy for Decision Tree Classifier  = 0.644737
```

Ishan Chokshi
BE19B018







2. The DL.ipynb file uploaded on moodle uses a pre-trained mobilenet model to run inference on the flowers dataset using Pyspark.

Modify the above code to run inference on CIFAR 10 dataset using Pyspark. Try a few different models pre-trained on Imagenet and report which works better.

The CIFAR 10 data set is provided by the University of Toronto (https://www.cs.toronto.edu/%7Ekriz/cifar-10-python.tar.gz). The dataset is split batchwise, hence some pre-processing is required for the data. It has to be converted into the format necessary for ImageNet. The final data frame looks as follows:

```
+--------------------+--------------------+------+--------------------+
|                path|    modificationTime|length|             content|
+--------------------+--------------------+------+--------------------+
|file:/content/cif...|2023-04-05 17:40:...|  1124|[FF D8 FF E0 00 1...|
|file:/content/cif...|2023-04-05 17:40:...|  1121|[FF D8 FF E0 00 1...|
|file:/content/cif...|2023-04-05 17:40:...|  1118|[FF D8 FF E0 00 1...|
|file:/content/cif...|2023-04-05 17:40:...|  1111|[FF D8 FF E0 00 1...|
|file:/content/cif...|2023-04-05 17:40:...|  1104|[FF D8 FF E0 00 1...|
+--------------------+--------------------+------+--------------------+
only showing top 5 rows
```

The classes used in the pre-trained models in ImageNet are different from CIFAR 10. In the code provided, only the flowers from the ImageNet dataset have been used. But for using CIFAR-10 dataset, I have included all the classes of the ImageNet dataset in order to get a better understanding of how well the model is performing, and to do a comparative analysis of different models later. The below image shows an example prediction output on our pretrained model for Imagenet. The predicted classes for CIFAR-10 dataset do not match the actual classes since this model has been trained on ImageNet dataset classes. Some predictions are not even closely related to the actual label. This shows that this pretrained model is not ideal for CIFAR-10 dataset.

Actual Class Label	Imagenet Prediction Label
airplane	chain_saw
automobile	moving_van
bird	fox_squirrel
cat	EntleBucher
deer	cardoon
dog	Japanese_spaniel
frog	rock_python
horse	sorrel
ship	moving_van
truck	moving_van

Now, I have compared the performance of 4 models – Densenet, Shufflenet, Alexnet, and Resnet50. For every class in the CIFAR dataset, I have done a valuecount() first for every predicted label and then assigned the most frequent label as the predicted label for that class. This way, I have compared the output for all the 4 models and displayed the output in a PySpark dataframe as shown below:

Actual Class	Densenet Prediction	Alexnet Prediction	Shufflenet Prediction	Resnet50 Prediction
airplane	airliner	panpipe	Windsor_tie	letter_opener
automobile	moving_van	moving_van	moving_van	moving_van
bird	limpkin	fox_squirrel	langur	limpkin
cat	fox_squirrel	English_foxhound	langur	fox_squirrel
deer	sorrel	Dandie_Dinmont	limpkin	toy_terrier
dog	Dandie_Dinmont	wire-haired_fox_t...	Japanese_spaniel	Japanese_spaniel
frog	fox_squirrel	fox_squirrel	fox_squirrel	tailed_frog
horse	sorrel	sorrel	Indian_elephant	sorrel
ship	speedboat	moving_van	Windsor_tie	moving_van
truck	moving_van	moving_van	moving_van	moving_van

From the above dataframe, we can conclude that Densenet is the best performing model since it closely predicts the actual class. Example: airliner for airplane, speedboat for ship, moving van for truck, limpkin for bird. All the models have predicted automobile and truck classes as moving van. This means that these classes are easier to predict on pretrained models. The horse class seems to be difficult to predict since most of the models have assigned it the sorrel class which is a plant. Note that I have used only the predictions of the first 1000 rows in order to assign the predicted label for each class. If we increase that number, the results might vary and it is possible that the predictions might improve.