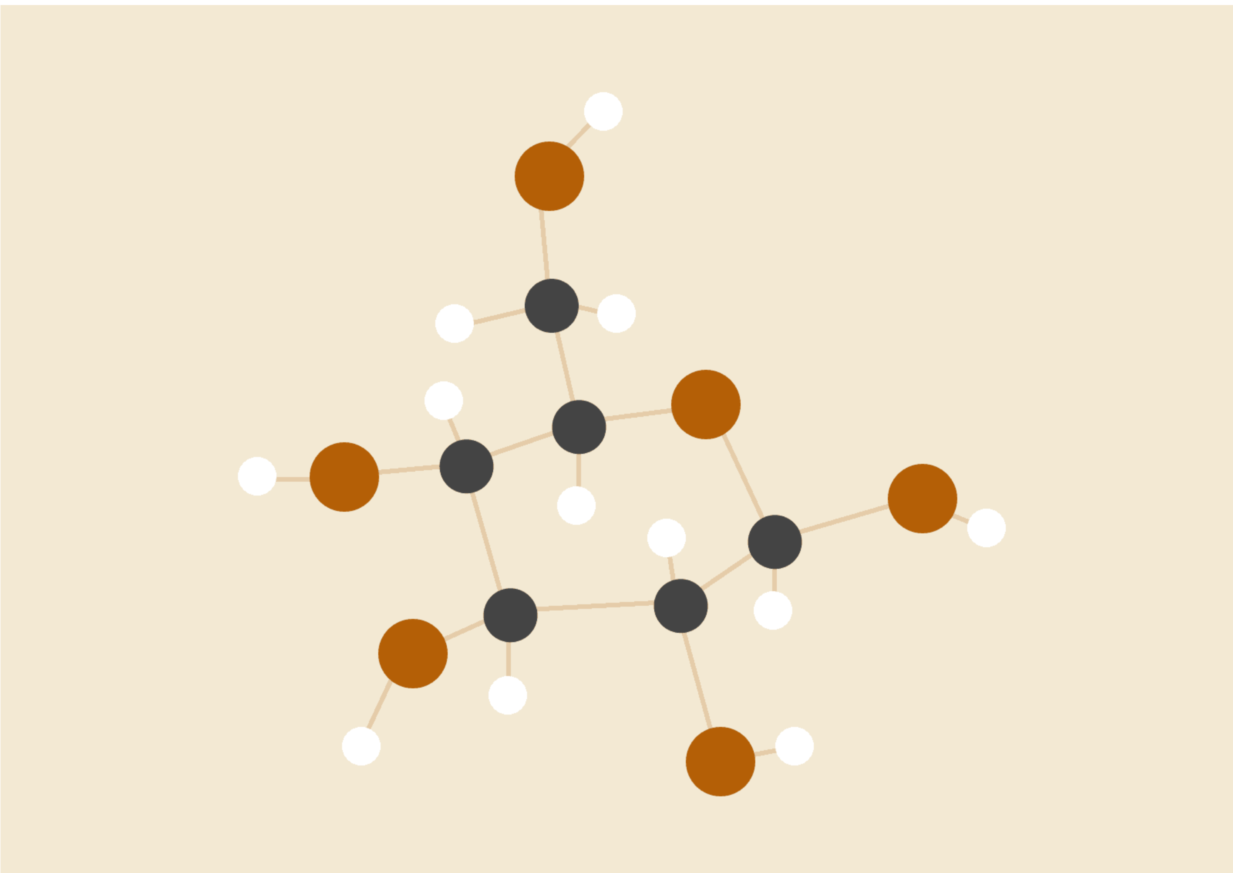
Plant Disease Classification using PySpark

CIE [427] Big Data Analytics - Fall 22'

Under the supervision of Dr. Elsayed Hemayed, Eng. Ahmed Wael, and Eng. Farah Sami.



Ahmed Saad - 201800251

Ehab Mansour - 201800506

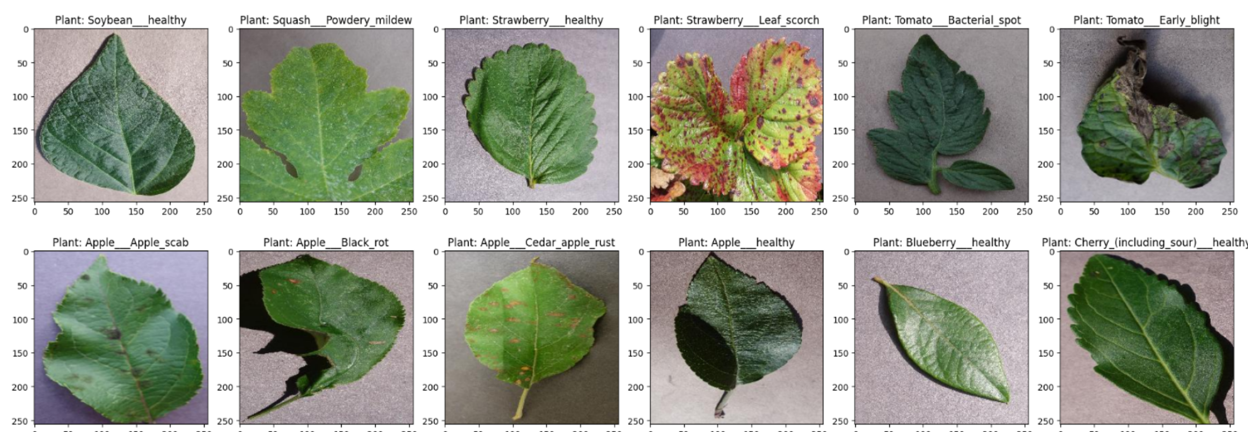
26.01.2023

Problem Statement

The increasing global population and the need for food security have highlighted the urgent need to improve crop yields. However, plant diseases continue to pose a significant threat to crop production, with an estimated 40% reduction in potential output. This problem is particularly severe in developing countries where farmers often experience yield losses of up to 100%. The widespread use of smartphones, with an estimated 5 billion in use worldwide by 2020, presents an opportunity to harness their potential as a tool for detecting and diagnosing plant diseases. By leveraging crowdsourcing and machine learning, mobile illness diagnostics can be developed and made accessible to farmers through their personal mobile phones or low-camera resolution devices. Such an approach can help to mitigate the impact of plant diseases and contribute to the goal of increasing food production to meet the needs of a growing population.

Analysis

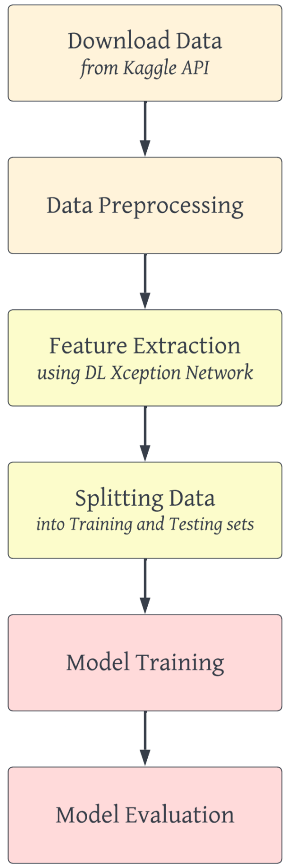
The dataset contains around 210k images of 38 types of plants as shown below. Each image is 256x256 pixels. There are 14 unique plants and 26 types of diseases.



Unique Plants are 14:

- Tomato - Grape - Orange - Soybean - Squash - Potato - Corn_(maize) - Strawberry - Peach - Apple - Blueberry - Cherry_(including_sour) - Pepper,_bell - Raspberry.

Pipeline



1. **Download Data:**

   Download the dataset from Kaggle using Kaggle API and initialize data and code folders to retrieve data.

2. **Data Preprocessing:**

   Changing categorical names of plants using ordinal encoding to train the model easily.

3. **Feature Extraction:**

   After setting the images dataset, We need to extract the features from each image and start applying Machine Learning multiclass models. We used pic2vec for feature extraction using Xception Network. The resulting output should be the label and features columns. Both are quantitative columns.

4. **Split Data:**

   Split our dataset into training and testing sets. With proportions 0.7 and 0. For training and testing sets by order. We made sure there is no overlapping between the two sets to avoid data leakage.

5. **Training Data:**

   The dataset is trained on Logistic Regression based on OneVsRest Classifier

6. **Evaluating Data:**

   The dataset is evaluated based on different metrics generated from the model summary like Accuracy, ROC curve, and F1-score.

7. **Hyperparameter Tuning:**

   We tried to perform hyper tuning but it takes too much time to retrieve results.

Trials

The following are the version of frameworks and libraries used in the project:

```
Spark 2.4.0
Hadoop 2.7
Java 8
jdk1.8.0_202
jre1.8.0_202
tensorflow==1.15.0
keras==2.2.5
pyspark==2.4.0
pic2vec==0.101.1
```

We faced many challenges while reading data or training models. We will show two main points of challenges:

1. Reading Images:
   a. We tried to read images from the ImageSchema module in spark:

```
AttributeError: '_ImageSchema' object has no attribute 'readImages'
```

   b. We tried to read from the IO module:

```
AttributeError: module 'sparkdl.image.imageIO' has no attribute 'readImagesWithCustomFn'
```

   c. We tried to read the image as an image `spark.read.format("image")`

```
Py4JJavaError: An error occurred while calling o38.collectToPython.
: org.apache.spark.SparkException: Job aborted due to stage failure:
java.io.FileNotFoundException:
```

   But it can’t see the images as image type, so We moved to read it as a binary file.

   d. Read images as binary files:

```
spark.read.format("binaryFile").load(f"{plant_path}/*.JPG")
```

   Actually this method worked, but the model cannot handle the binary format generated from binaryFiles format.

```
+--------------------+-------------------+------+--------------------+
|                path|   modificationTime|length|             content|
+--------------------+-------------------+------+--------------------+
|file:/content/dri...|2019-10-12 07:40:04| 29714|[FF D8 FF E0 00 1...|
|file:/content/dri...|2019-10-12 07:40:08| 28770|[FF D8 FF E0 00 1...|
|file:/content/dri...|2019-10-12 07:40:08| 28763|[FF D8 FF E0 00 1...|
|file:/content/dri...|2019-10-12 07:40:06| 28646|[FF D8 FF E0 00 1...|
|file:/content/dri...|2019-10-12 07:40:08| 28579|[FF D8 FF E0 00 1...|
```

   We tried to convert this format to integer or double format, but none of the many trials worked.

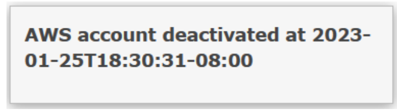
```python
from pyspark.sql.functions import unhex

from struct import unpack

from pyspark.sql.functions import udf

def binary_to_float(binary_data):

    return unpack('f', binary_data)[0]

binary_to_float_udf = udf(binary_to_float)

df = df.withColumn("content_float",
binary_to_float_udf(unhex(df["content"])))
```

2. Extracting Features Challenge:
    a. To extract features from images is to have an array of features for every image. And as long as we have the same image size for all images. We should have the same feature length.
    b. We tried the sparkdl library with feature extraction and evaluations using `DeepImageFeaturizer(inputCol="image", outputCol="features", modelName="InceptionV3")`

       `DeepImagePredictor(inputCol="image",outputCol="predictions")`

       But unfortunately, the library is incompatible with TF>1.x and Google Colab doesn't offer TF<2.x. Therefore, there is no ability to downgrade with Colab.

    c. We tried to download data on our system and downgrade to TensorFlow == 1.13 but **sparkdl** library threw java errors we couldn't solve.
    d. The last trial is with a library called **pic2vec**: which uses a machine learning model to get features of images as vectors. By using this, We could generate features for every image using Xception Network. We passed the CSV file to assemble features and label to start LogisticRegression Classifier.
3. AWS Challenge:
    a. We tried to make the nodes more than 3 nodes (1 master and 2 slaves) we tried to increase the number of slaves to 7, but after running the code it gave an error and then the account got deactivated and I couldn't open the lab again.



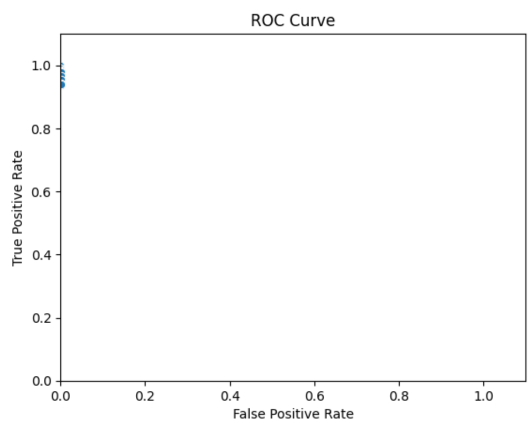
b. We used another account to run the code on AWS and It run successfully without errors

RESULTS

1. Local

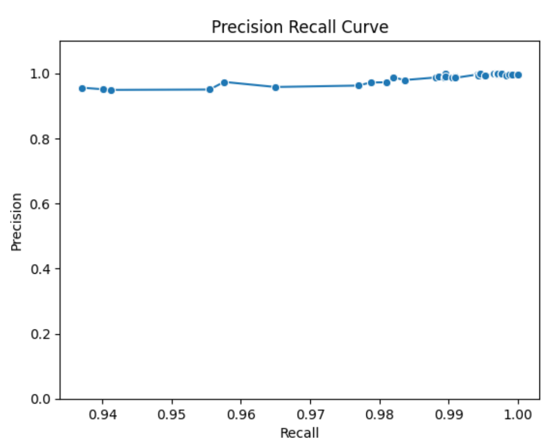
a. Accuracy: 98.49% for the training set and 97.81% for the testing set.

b. ROC curve:



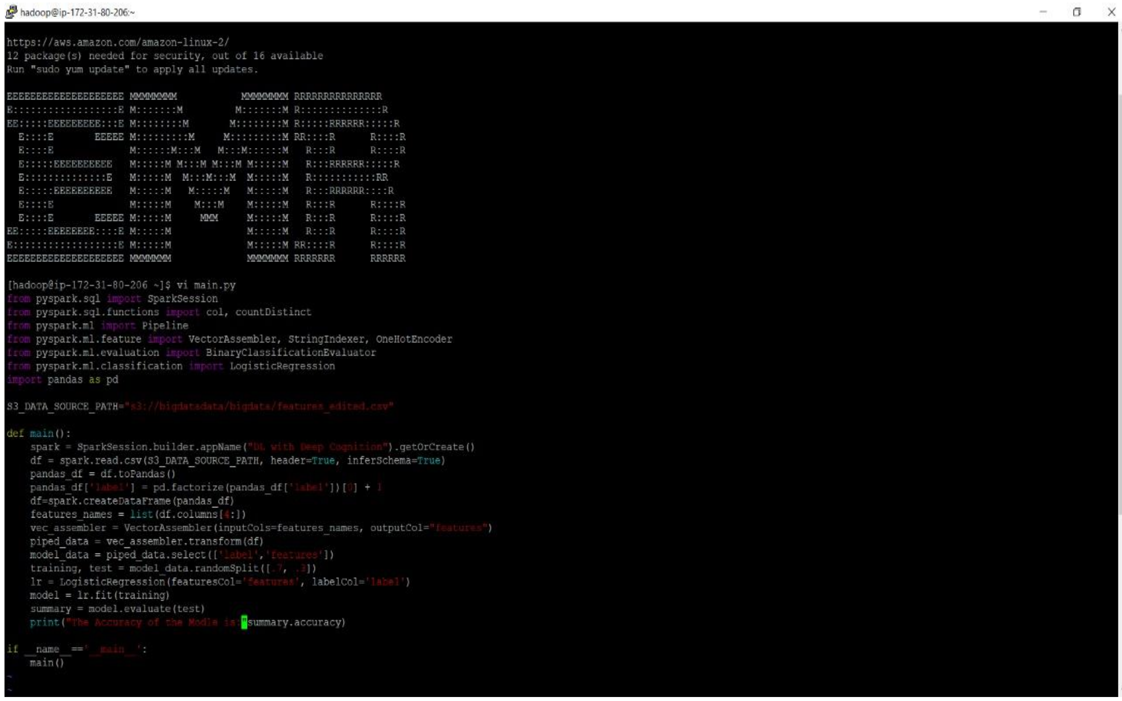
There are high values in True Positive Rate (TPR) and low values in False Positive Rate (FPR).

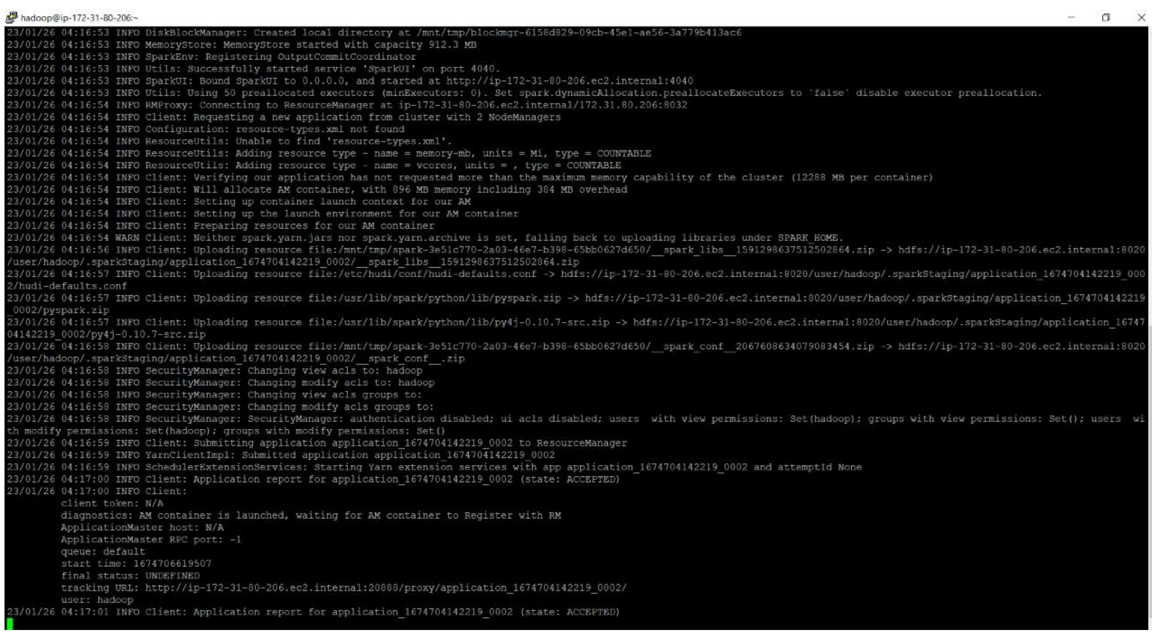
c. Precision and Recall Curve:



2. AWS

a. We upload half of the data to S3 socket at AWS and then apply the model on it and connect using putty and SSH to the AWS server to apply the model.





```
hadoop@ip-172-31-80-206:~
23/01/26 04:26:06 INFO BlockManagerInfo: Added broadcast_324_piece0 in memory on ip-172-31-80-206.ec2.internal:36349 (size: 2.2 KB, free: 912.1 MB)
23/01/26 04:26:06 INFO SparkContext: Created broadcast 324 from broadcast at DAGScheduler.scala:1297
23/01/26 04:26:06 INFO DAGScheduler: Submitting 8 missing tasks from ResultStage 216 (ShuffledRDD[470] at countByValue at MulticlassMetrics.scala:42) (first 15 tasks are for partitions Vect
or(0, 1, 2, 3, 4, 5, 6, 7))
23/01/26 04:26:06 INFO YarnScheduler: Adding task set 216.0 with 8 tasks
23/01/26 04:26:06 INFO TaskSetManager: Starting task 0.0 in stage 216.0 (TID 1091, ip-172-31-95-73.ec2.internal, executor 2, partition 0, NODE_LOCAL, 7975 bytes)
23/01/26 04:26:06 INFO TaskSetManager: Starting task 1.0 in stage 216.0 (TID 1092, ip-172-31-95-73.ec2.internal, executor 2, partition 1, PROCESS_LOCAL, 7834 bytes)
23/01/26 04:26:06 INFO TaskSetManager: Starting task 2.0 in stage 216.0 (TID 1093, ip-172-31-86-184.ec2.internal, executor 1, partition 2, PROCESS_LOCAL, 7834 bytes)
23/01/26 04:26:06 INFO TaskSetManager: Starting task 3.0 in stage 216.0 (TID 1094, ip-172-31-95-73.ec2.internal, executor 2, partition 3, PROCESS_LOCAL, 7834 bytes)
23/01/26 04:26:06 INFO TaskSetManager: Starting task 4.0 in stage 216.0 (TID 1095, ip-172-31-86-184.ec2.internal, executor 1, partition 4, PROCESS_LOCAL, 7834 bytes)
23/01/26 04:26:06 INFO TaskSetManager: Starting task 5.0 in stage 216.0 (TID 1096, ip-172-31-95-73.ec2.internal, executor 2, partition 5, PROCESS_LOCAL, 7834 bytes)
23/01/26 04:26:06 INFO TaskSetManager: Starting task 6.0 in stage 216.0 (TID 1097, ip-172-31-86-184.ec2.internal, executor 1, partition 6, PROCESS_LOCAL, 7834 bytes)
23/01/26 04:26:06 INFO TaskSetManager: Starting task 7.0 in stage 216.0 (TID 1098, ip-172-31-86-184.ec2.internal, executor 1, partition 7, PROCESS_LOCAL, 7834 bytes)
23/01/26 04:26:06 INFO BlockManagerInfo: Added broadcast_324_piece0 in memory on ip-172-31-95-73.ec2.internal:45899 (size: 2.2 KB, free: 4.8 GB)
23/01/26 04:26:06 INFO BlockManagerInfo: Added broadcast_324_piece0 in memory on ip-172-31-86-184.ec2.internal:38313 (size: 2.2 KB, free: 4.8 GB)
23/01/26 04:26:06 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 106 to 172.31.95.73:47422
23/01/26 04:26:06 INFO MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 106 to 172.31.86.184:44430
23/01/26 04:26:06 INFO TaskSetManager: Finished task 3.0 in stage 216.0 (TID 1094) in 13 ms on ip-172-31-95-73.ec2.internal (executor 2) (1/8)
23/01/26 04:26:06 INFO TaskSetManager: Finished task 5.0 in stage 216.0 (TID 1096) in 13 ms on ip-172-31-95-73.ec2.internal (executor 2) (2/8)
23/01/26 04:26:06 INFO TaskSetManager: Finished task 7.0 in stage 216.0 (TID 1098) in 14 ms on ip-172-31-86-184.ec2.internal (executor 1) (3/8)
23/01/26 04:26:06 INFO TaskSetManager: Finished task 6.0 in stage 216.0 (TID 1097) in 16 ms on ip-172-31-86-184.ec2.internal (executor 1) (4/8)
23/01/26 04:26:06 INFO TaskSetManager: Finished task 0.0 in stage 216.0 (TID 1091) in 17 ms on ip-172-31-95-73.ec2.internal (executor 2) (5/8)
23/01/26 04:26:06 INFO TaskSetManager: Finished task 4.0 in stage 216.0 (TID 1095) in 18 ms on ip-172-31-86-184.ec2.internal (executor 1) (6/8)
23/01/26 04:26:06 INFO TaskSetManager: Finished task 1.0 in stage 216.0 (TID 1092) in 19 ms on ip-172-31-95-73.ec2.internal (executor 2) (7/8)
23/01/26 04:26:06 INFO TaskSetManager: Finished task 2.0 in stage 216.0 (TID 1093) in 19 ms on ip-172-31-86-184.ec2.internal (executor 1) (8/8)
23/01/26 04:26:06 INFO YarnScheduler: Removed TaskSet 216.0, whose tasks have all completed, from pool
23/01/26 04:26:06 INFO DAGScheduler: ResultStage 216 (countByValue at MulticlassMetrics.scala:42) finished in 0.670 s
23/01/26 04:26:06 INFO DAGScheduler: Job 109 finished: countByValue at MulticlassMetrics.scala:42, took 4.365519 s
The Accuracy of the Modle is: 0.9762611275964391
23/01/26 04:26:07 INFO SparkContext: Invoking stop() from shutdown hook
23/01/26 04:26:07 INFO SparkUI: Stopped Spark web UI at http://ip-172-31-80-206.ec2.internal:4040
23/01/26 04:26:07 INFO YarnClientSchedulerBackend: Interrupting monitor thread
23/01/26 04:26:07 INFO YarnClientSchedulerBackend: Shutting down all executors
23/01/26 04:26:07 INFO YarnSchedulerBackend$YarnDriverEndpoint: Asking each executor to shut down
23/01/26 04:26:07 INFO SchedulerExtensionServices: Stopping SchedulerExtensionServices
(serviceOption=None,
 services=List(),
 started=false)
23/01/26 04:26:07 INFO YarnClientSchedulerBackend: Stopped
23/01/26 04:26:07 INFO MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
23/01/26 04:26:07 INFO MemoryStore: MemoryStore cleared
23/01/26 04:26:07 INFO BlockManager: BlockManager stopped
23/01/26 04:26:07 INFO BlockManagerMaster: BlockManagerMaster stopped
23/01/26 04:26:07 INFO OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stopped!
23/01/26 04:26:07 INFO SparkContext: Successfully stopped SparkContext
23/01/26 04:26:07 INFO ShutdownHookManager: Shutdown hook called
23/01/26 04:26:07 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-3e51c770-2a03-46e7-b398-65bb0627d650/pyspark-4111b10c-2bdc-4e14-ab48-5fe4bd5a3a08
23/01/26 04:26:07 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-feb0de4d-6700-43c4-b518-bcea72095bd8
23/01/26 04:26:07 INFO ShutdownHookManager: Deleting directory /mnt/tmp/spark-3e51c770-2a03-46e7-b398-65bb0627d650
[hadoop@ip-172-31-80-206 ~]$
```

It gave an accuracy of 97%, and it run about 108 task to finish processing

108	Succeeded	countByValue at MulticlassMetrics.scala:42	2023-01-26 06:09 (UTC+2)	4 s	2 / 2	16 / 16
107	Succeeded	collectAsMap at MulticlassMetrics.scala:48	2023-01-26 06:09 (UTC+2)	4 s	2 / 2	16 / 16
106	Succeeded	treeAggregate at RDDLossFunction.scala:61	2023-01-26 06:09 (UTC+2)	2 s	2 / 2	10 / 10
105	Succeeded	treeAggregate at RDDLossFunction.scala:61	2023-01-26 06:09 (UTC+2)	2 s	2 / 2	10 / 10
104	Succeeded	treeAggregate at RDDLossFunction.scala:61	2023-01-26 06:09 (UTC+2)	2 s	2 / 2	10 / 10
103	Succeeded	treeAggregate at RDDLossFunction.scala:61	2023-01-26 06:09 (UTC+2)	2 s	2 / 2	10 / 10
102	Succeeded	treeAggregate at RDDLossFunction.scala:61	2023-01-26 06:09 (UTC+2)	2 s	2 / 2	10 / 10
101	Succeeded	treeAggregate at RDDLossFunction.scala:61	2023-01-26 06:09 (UTC+2)	2 s	2 / 2	10 / 10
100	Succeeded	treeAggregate at RDDLossFunction.scala:61	2023-01-26 06:09 (UTC+2)	2 s	2 / 2	10 / 10
99	Succeeded	treeAggregate at RDDLossFunction.scala:61	2023-01-26 06:09 (UTC+2)	2 s	2 / 2	10 / 10
98	Succeeded	treeAggregate at RDDLossFunction.scala:61	2023-01-26 06:09 (UTC+2)	2 s	2 / 2	10 / 10
97	Succeeded	treeAggregate at RDDLossFunction.scala:61	2023-01-26 06:09 (UTC+2)	3 s	2 / 2	10 / 10

Future Enhancements

There are several potential future enhancements that can be made to this project:

1. **Data augmentation:** One way to improve the performance of the model is to use data augmentation techniques, such as rotating, flipping, or cropping images, to increase the diversity of the training data. This can help to reduce overfitting and improve the robustness of the model.

2. **Model Ensemble:** Another way to improve the performance of the model is to use an ensemble of different models, such as decision trees, random forests, or gradient boosting. This can help to reduce the variance and bias of the model and improve the overall accuracy.

3. **Feature Engineering:** Another way to improve the performance of the model is to use feature engineering techniques, such as dimensionality reduction, to extract more meaningful features from the data. This can help to improve the interpretability of the model and reduce its complexity of the model.

4. **Upgrading to newer versions of PySpark:** As We are using Spark 2.4.0, We can try to upgrade your PySpark version to the latest one which is more robust and has more features than the older version. This can help to improve the performance of the model and make model more robust. But this will need a method to generate features from images as the **pic2vec** library won't work.