CISC5950_Final Project Report

Haoran Xue, Sheng Yun

May 7th, 2023

Introduction

In this project, we would be leveraging the power of Apache Spark for data processing and machine learning tasks, focusing on four distinct parts, including a basic text cleaning process using the toxic comment text classification dataset, applying Logistic Regression to identify the most significant risk factors associated with heart disease and predict overall risk levels, analyzing the Census Dataset from the UCI Machine Learning Repository, as well as employing Random Forest Classifier and Decision Tree Classifier using Apache Spark ML/MLlib to train the dataset. The main objectives of the project were to provide an informative explanation of the data cleaning process, the pipeline, and the explanation and evaluation of model performances. The code utilized various libraries, such as PySpark, Pandas, and Sys, and functionalities to execute the data pre-processing, feature engineering, and model training tasks.

P1: Toxic Comment Classification

In this part, we were going to explore basic text processing using the toxic comment text classification dataset. Although the machine learning and text processing techniques employed were not sophisticated, our primary objective was to convert the comment text column into a sparse vector representation that could be utilized by a classification algorithm within the Spark ML library. According to the instructions, we only need to make the sample code work, so the main problem was just to convert the code to the spark code that looked like ours. The biggest challenge here was the resource setup because the sample code allocated the amount we could not give on our setup clusters(e2-standard), which only have 2GB and an 8-core CPU. This gave us a hard time testing out the settings(larger resources, the scheduler would not accept, but smaller ones would make the server idling). Then we gave up on setting up our own and chose to use the default settings(no configs at all!), the program run perfectly after making that changes.

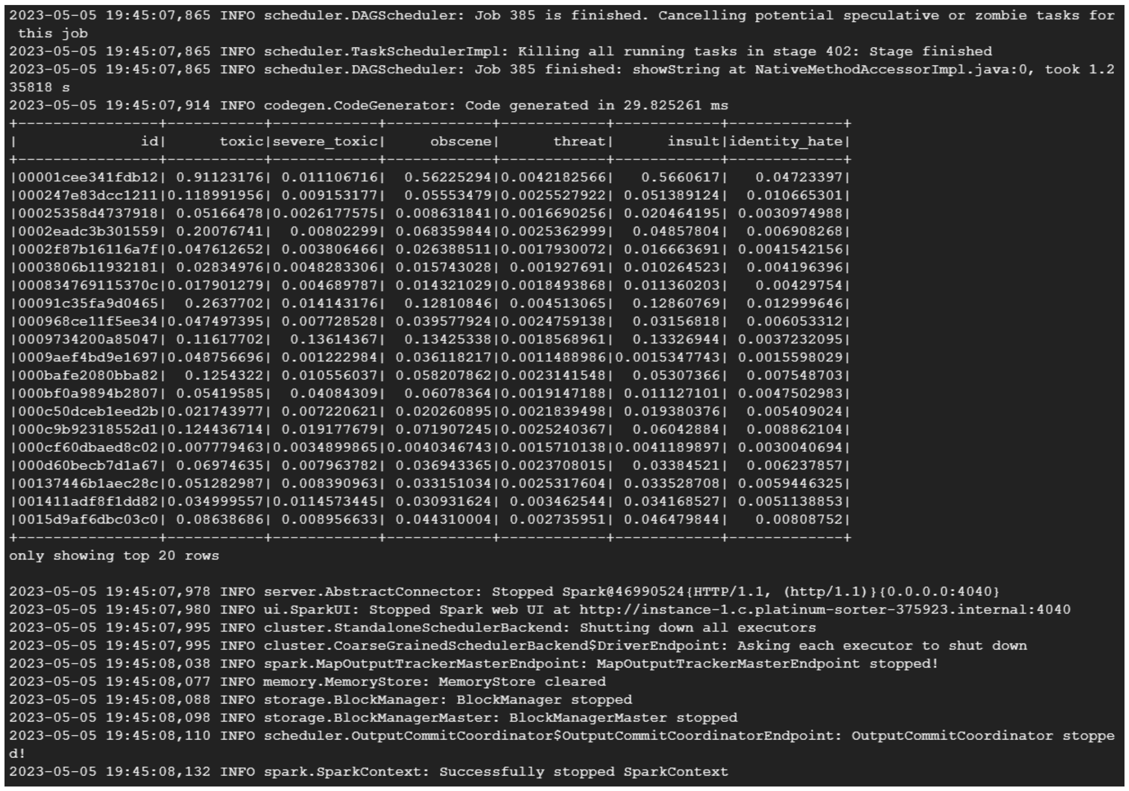
Data Description and Data Cleaning

The Toxic Comment Text Classification dataset is a large dataset originally released by Jigsaw and Google. The data is formatted as a 'csv' file, with each row representing a unique comment. The columns in the dataset include id, comment_text, and other binary labels (0 or 1) indicating whether the comment falls into the respective toxicity category.

The data cleaning process began with the conversion of input 'csv' files into Spark DataFrames using the 'to_spark_df()' function. This function read a 'csv' file using the Pandas library, replaced missing values with empty strings, and then converted the Pandas DataFrame into a Spark DataFrame. This process ensured that the data was cleaned and suitable for further analysis within the PySpark environment. The cleaned DataFrames were subsequently used as training and testing datasets.

Model Explanation and Evaluation

Their designed pipeline includes several steps of to do this task, including tokenization, feature extraction, and model training. First, the Tokenizer was applied to the 'comment_text' column to split the text into a list of words. Then the HashingTF (hashing term frequency) technique was employed to transform the list of words into numerical features. Following this, the IDF (inverse document frequency) was calculated to weigh the importance of words in the dataset. Finally, the product of TF and IDF was computed to generate the TF-IDF features, which represented the significance of words in each comment. The chosen model for this task was Logistic Regression, which was implemented using the PySpark 'LogisticRegression' class. The model was trained separately for each label column in the 'out_cols' list, excluding the 'id' and 'comment_text' columns. The regularization parameter (REG) was set to 0.1 to prevent overfitting. For each label, the model was fit on the training dataset with TF-IDF features and then used to make predictions on the test dataset. The probability of a comment being classified as toxic was extracted and added to the 'test_res' DataFrame. This process was repeated for all labels in the 'out_cols' list. By incorporating a well-structured data-cleaning process and a comprehensive pipeline, the program successfully prepared the data, extracted relevant features, and trained the model for accurate classification. The use of Logistic Regression ensured that



```
2023-05-05 19:45:07,865 INFO scheduler.DAGScheduler: Job 385 is finished. Cancelling potential speculative or zombie tasks for
 this job
2023-05-05 19:45:07,865 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 402: Stage finished
2023-05-05 19:45:07,865 INFO scheduler.DAGScheduler: Job 385 finished: showString at NativeMethodAccessorImpl.java:0, took 1.2
35818 s
2023-05-05 19:45:07,914 INFO codegen.CodeGenerator: Code generated in 29.825261 ms
+----------------+-----------+------------+------------+------------+------------+-------------+
|              id|      toxic|severe_toxic|     obscene|      threat|      insult|identity_hate|
+----------------+-----------+------------+------------+------------+------------+-------------+
|00001cee341fdb12| 0.91123176| 0.011106716|  0.56225294|0.0042182566|   0.5660617|   0.04723397|
|000247e83dcc1211|0.118991956| 0.009153177|  0.05553479|0.0025527922| 0.051389124|  0.010665301|
|00025358d4737918| 0.05166478|0.0026177575| 0.008631841|0.0016690256| 0.020464195| 0.0030974988|
|0002eadc3b301559| 0.20076741|  0.00802299| 0.068359844|0.0025362999|  0.04857804|  0.006908268|
|0002f87b16116a7f|0.047612652| 0.003806466| 0.026388511|0.0017930072| 0.016663691| 0.0041542156|
|0003806b11932181| 0.02834976|0.0048283306| 0.015743028| 0.001927691| 0.010264523|  0.004196396|
|000834769115370c|0.017901279| 0.004689787| 0.014321029|0.0018493868| 0.011360203|   0.00429754|
|00091c35fa9d0465|  0.2637702| 0.014143176|  0.12810846| 0.004513065|  0.12860769|  0.012999646|
|000968ce11f5ee34|0.047497395| 0.007728528| 0.039577924|0.0024759138|  0.03156818|  0.006053312|
|0009734200a85047| 0.11617702|  0.13614367|  0.13425338|0.0018568961|  0.13326944| 0.0037232095|
|0009aef4bd9e1697|0.048756696| 0.001222984| 0.036118217|0.0011488986|0.0015347743| 0.0015598029|
|000bafe2080bba82|  0.1254322| 0.010556037| 0.058207862|0.0023141548|  0.05307366|  0.007548703|
|000bf0a9894b2807| 0.05419585|  0.04084309|  0.06078364|0.0019147188| 0.011127101| 0.0047502983|
|000c50dceb1eed2b|0.021743977| 0.007220621| 0.020260895|0.0021839498| 0.019380376|  0.005409024|
|000c9b92318552d1|0.124436714| 0.019177679| 0.071907245|0.0025240367|  0.06042884|  0.008862104|
|000cf60dbaed8c02|0.007779463|0.0034899865|0.0040346743|0.0015710138|0.0041189897| 0.0030040694|
|000d60becb7d1a67| 0.06974635| 0.007963782| 0.036943365|0.0023708015|  0.03384521|  0.006237857|
|00137446b1aec28c|0.051282987| 0.008390963| 0.033151034|0.0025317604| 0.033528708| 0.0059446325|
|001411adf8f1dd82|0.034999557|0.0114573445| 0.030931624| 0.003462544| 0.034168527| 0.0051138853|
|0015d9af6dbc03c0| 0.08638686| 0.008956633| 0.044310004| 0.002735951| 0.046479844|   0.00808752|
+----------------+-----------+------------+------------+------------+------------+-------------+
only showing top 20 rows

2023-05-05 19:45:07,978 INFO server.AbstractConnector: Stopped Spark@46990524{HTTP/1.1, (http/1.1)}{0.0.0.0:4040}
2023-05-05 19:45:07,980 INFO ui.SparkUI: Stopped Spark web UI at http://instance-1.c.platinum-sorter-375923.internal:4040
2023-05-05 19:45:07,995 INFO cluster.StandaloneSchedulerBackend: Shutting down all executors
2023-05-05 19:45:07,995 INFO cluster.CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each executor to shut down
2023-05-05 19:45:08,038 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
2023-05-05 19:45:08,077 INFO memory.MemoryStore: MemoryStore cleared
2023-05-05 19:45:08,088 INFO storage.BlockManager: BlockManager stopped
2023-05-05 19:45:08,098 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
2023-05-05 19:45:08,110 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stoppe
d!
2023-05-05 19:45:08,132 INFO spark.SparkContext: Successfully stopped SparkContext
```

Figure 1: Toxic Comment with LR

the model was both interpretative and efficient in identifying toxic comments. We got the first 20 rows with the comment ids and the predicted probabilities for each class as output, shown in Figure 1.

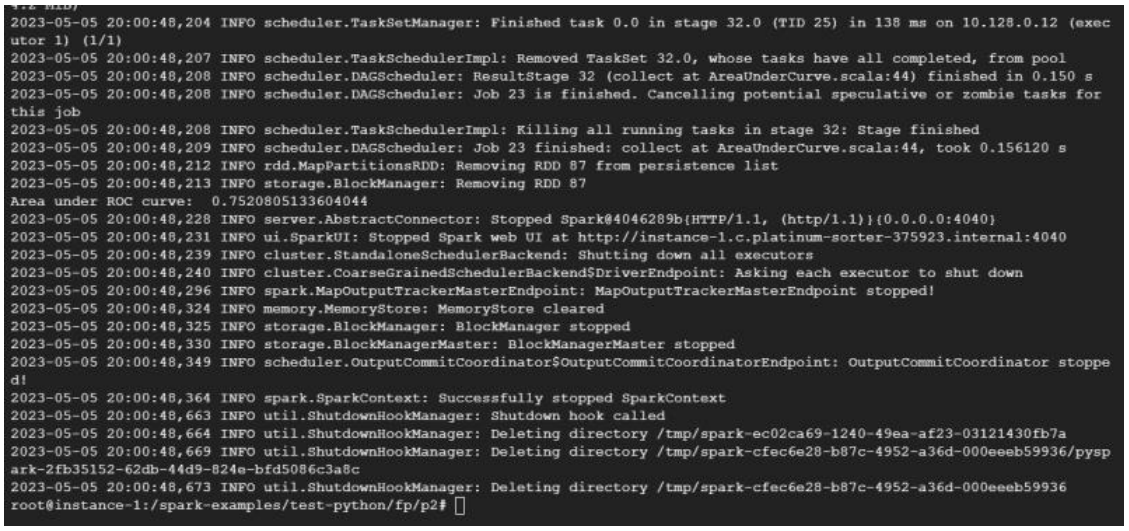
P2: Heart Disease Prediction using Logistic Regression

In this part, we converted a Python script to PySpark to predict the likelihood of heart disease within a 10-year period using logistic regression. The Python script used Logistic Regression to identify the most significant risk factors associated with heart disease and predict overall risk levels. In this question, we would be using the Framingham Heart dataset.

Data Description and Data Cleaning

The Framingham Heart Dataset is originated from the Framingham Heart Study, with an initial cohort of 5209 subjects. The Framingham Heart dataset comprises various demographic, clinical, and laboratory attributes collected from the study participants. The data typically includes various demographic information about patients, such as age, sex, BMI, the measurement information of patients such as heart rate, total cholesterol level, systolic blood pressure, diastolic blood pressure, and fasting blood pressure etc., and the outcome of the presence or absence of coronary heart disease. This dataset enables researchers to investigate the relationship between various risk factors.

The data cleaning process began with loading the dataset using the 'csv' format and inferring the schema as usual. The 'education' column was dropped because it is not relevant to the prediction task. Additionally, we renamed the 'male' column to 'Sex_male' for clarity. Missing values were removed using the 'dropna()' function, and the dataset was split into training and test sets with an 8-2 ratio. The 'cigsPerDay', 'totChol', and 'glucose' columns were then cast to the 'FloatType', and any remaining missing values were dropped.



```
2023-05-05 20:00:48,204 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 32.0 (TID 25) in 138 ms on 10.128.0.12 (exec
utor 1) (1/1)
2023-05-05 20:00:48,207 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 32.0, whose tasks have all completed, from pool
2023-05-05 20:00:48,208 INFO scheduler.DAGScheduler: ResultStage 32 (collect at AreaUnderCurve.scala:44) finished in 0.150 s
2023-05-05 20:00:48,208 INFO scheduler.DAGScheduler: Job 23 is finished. Cancelling potential speculative or zombie tasks for
this job
2023-05-05 20:00:48,208 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 32: Stage finished
2023-05-05 20:00:48,209 INFO scheduler.DAGScheduler: Job 23 finished: collect at AreaUnderCurve.scala:44, took 0.156120 s
2023-05-05 20:00:48,212 INFO rdd.MapPartitionsRDD: Removing RDD 87 from persistence list
2023-05-05 20:00:48,213 INFO storage.BlockManager: Removing RDD 87
Area under ROC curve:  0.7520805133604044
2023-05-05 20:00:48,228 INFO server.AbstractConnector: Stopped Spark@4046289b{HTTP/1.1, (http/1.1)}{0.0.0.0:4040}
2023-05-05 20:00:48,231 INFO ui.SparkUI: Stopped Spark web UI at http://instance-1.c.platinum-sorter-375923.internal:4040
2023-05-05 20:00:48,239 INFO cluster.StandaloneSchedulerBackend: Shutting down all executors
2023-05-05 20:00:48,240 INFO cluster.CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each executor to shut down
2023-05-05 20:00:48,296 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
2023-05-05 20:00:48,324 INFO memory.MemoryStore: MemoryStore cleared
2023-05-05 20:00:48,325 INFO storage.BlockManager: BlockManager stopped
2023-05-05 20:00:48,330 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
2023-05-05 20:00:48,349 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stoppe
d!
2023-05-05 20:00:48,364 INFO spark.SparkContext: Successfully stopped SparkContext
2023-05-05 20:00:48,663 INFO util.ShutdownHookManager: Shutdown hook called
2023-05-05 20:00:48,664 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-ec02ca69-1240-49ea-af23-03121430fb7a
2023-05-05 20:00:48,669 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-cfec6e28-b87c-4952-a36d-000eeeb59936/pysp
ark-2fb35152-62db-44d9-824e-bfd5086c3a8c
2023-05-05 20:00:48,673 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-cfec6e28-b87c-4952-a36d-000eeeb59936
root@instance-1:/spark-examples/test-python/fp/p2#
```

Figure 2: Hear Dsease with LR

Model Explanation and Evaluation

We used several steps of pipeline, including feature engineering, model training, and evaluation. First, the 'VectorAssembler' was used to combine the selected feature columns into a single 'features' column. The features were then standardized using the 'StandardScaler' to ensure that they had equal importance during model training. Next, a logistic regression model was defined with 'scaled_features' as input and 'TenYearCHD' as the label column. The pipeline was then built using the 'Pipeline' class, with the assembler, scaler, and logistic regression model as stages.

The chosen model for this task was logistic regression, so we implemented using the PySpark 'LogisticRegression' class. Logistic regression is a popular choice for binary classification tasks as it provides an interpretable and efficient method for predicting probabilities. The model is trained on the preprocessed and standardized features in the training dataset, and predictions are made on the test dataset using the 'transform()' function. The performance of the logistic regression model was evaluated using the area under the receiver operating characteristic (ROC) curve, which is a widely accepted metric for binary classification tasks. The 'BinaryClassificationEvaluator' class was used to calculate the area under the ROC curve, which provided a measure of the model's ability to discriminate between positive and negative cases. The resulting value was then printed to provide insight into the model's performance. As the result shown in Figure 2, we got the AUC score with 0.752, which meant that the model was moderate successful in correctly predicting instances of both classes.

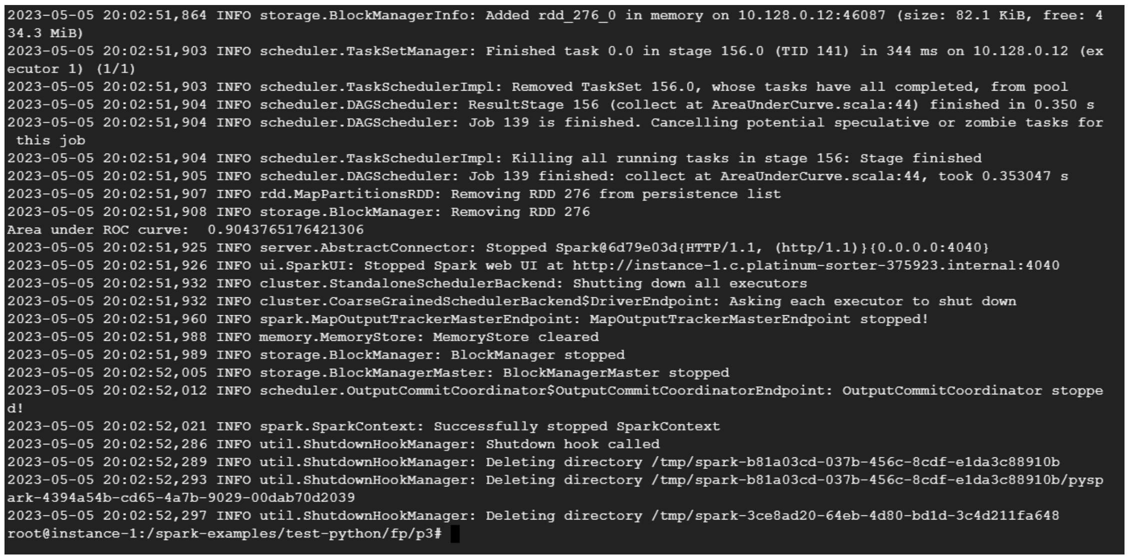
P3: Logistic Regression classifier on Census Income Data

In this part, we aimed at training a binary classifier to predict an individual's income level which had two possible values, ¿50K and ¡50K, using the Census dataset from the UCI Machine Learning Repository, based on various demographic and economic features.

Dataset Description and Data Cleaning

The dataset on Census Dataset consists of anonymous information such as age, occupation, education, and working class, with 48,842 instances and 14 attributes, the dataset offers a diverse mix of categorical, numerical, and missing values. We would utilize Spark ML/MLlib's logistic regression capabilities to train our model and evaluate its performance, aiming to achieve a high accuracy rate.

Our data cleaning process started with loading the train and test datasets using the 'csv' format and inferring the schema. Columns are renamed to more descriptive names, and the 'income' column is processed such that values were converted to binary indicators (1 for "¿50K" and 0 otherwise). We have found there was a leading space for "¿50" in both train and test set, and a "." in the end of the "¿50K" for test set. The train and test datasets were then combined for pre-processing purposes.



```
2023-05-05 20:02:51,864 INFO storage.BlockManagerInfo: Added rdd_276_0 in memory on 10.128.0.12:46087 (size: 82.1 KiB, free: 4
34.3 MiB)
2023-05-05 20:02:51,903 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 156.0 (TID 141) in 344 ms on 10.128.0.12 (ex
ecutor 1) (1/1)
2023-05-05 20:02:51,903 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 156.0, whose tasks have all completed, from pool
2023-05-05 20:02:51,904 INFO scheduler.DAGScheduler: ResultStage 156 (collect at AreaUnderCurve.scala:44) finished in 0.350 s
2023-05-05 20:02:51,904 INFO scheduler.DAGScheduler: Job 139 is finished. Cancelling potential speculative or zombie tasks for
 this job
2023-05-05 20:02:51,904 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 156: Stage finished
2023-05-05 20:02:51,905 INFO scheduler.DAGScheduler: Job 139 finished: collect at AreaUnderCurve.scala:44, took 0.353047 s
2023-05-05 20:02:51,907 INFO rdd.MapPartitionsRDD: Removing RDD 276 from persistence list
2023-05-05 20:02:51,908 INFO storage.BlockManager: Removing RDD 276
Area under ROC curve:  0.9043765176421306
2023-05-05 20:02:51,925 INFO server.AbstractConnector: Stopped Spark@6d79e03d{HTTP/1.1, (http/1.1)}{0.0.0.0:4040}
2023-05-05 20:02:51,926 INFO ui.SparkUI: Stopped Spark web UI at http://instance-1.c.platinum-sorter-375923.internal:4040
2023-05-05 20:02:51,932 INFO cluster.StandaloneSchedulerBackend: Shutting down all executors
2023-05-05 20:02:51,932 INFO cluster.CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each executor to shut down
2023-05-05 20:02:51,960 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
2023-05-05 20:02:51,988 INFO memory.MemoryStore: MemoryStore cleared
2023-05-05 20:02:51,989 INFO storage.BlockManager: BlockManager stopped
2023-05-05 20:02:52,005 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
2023-05-05 20:02:52,012 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stoppe
d!
2023-05-05 20:02:52,021 INFO spark.SparkContext: Successfully stopped SparkContext
2023-05-05 20:02:52,286 INFO util.ShutdownHookManager: Shutdown hook called
2023-05-05 20:02:52,289 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-b81a03cd-037b-456c-8cdf-e1da3c88910b
2023-05-05 20:02:52,293 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-b81a03cd-037b-456c-8cdf-e1da3c88910b/pysp
ark-4394a54b-cd65-4a7b-9029-00dab70d2039
2023-05-05 20:02:52,297 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-3ce8ad20-64eb-4d80-bd1d-3c4d211fa648
root@instance-1:/spark-examples/test-python/fp/p3#
```

Figure 3: Census with LR

Categorical columns were identified and indexed using the 'StringIndexer' class, with the '_index' suffix added to the output column names. OneHotEncoder was used to encode the indexed categorical columns, appending an '_encoded' suffix to the output column names.

Model Explanation and Evaluation

In order to achieve our prediction goal, we implemented several processing steps in a pipeline that as the same as we did in part2. First, we employed the 'VectorAssembler' to merge the encoded categorical columns with the numerical columns, creating a unified 'features' column. Next, we utilized the 'StandardScaler' to standardize the features, ensuring that each feature carried equal significance during the model training phase. We then established a logistic regression model with 'scaled_features' as input and 'income' as the label column. Finally, we constructed the pipeline using the 'Pipeline' class, incorporating the indexers, encoders, assembler, scaler, and logistic regression model as stages in the process.

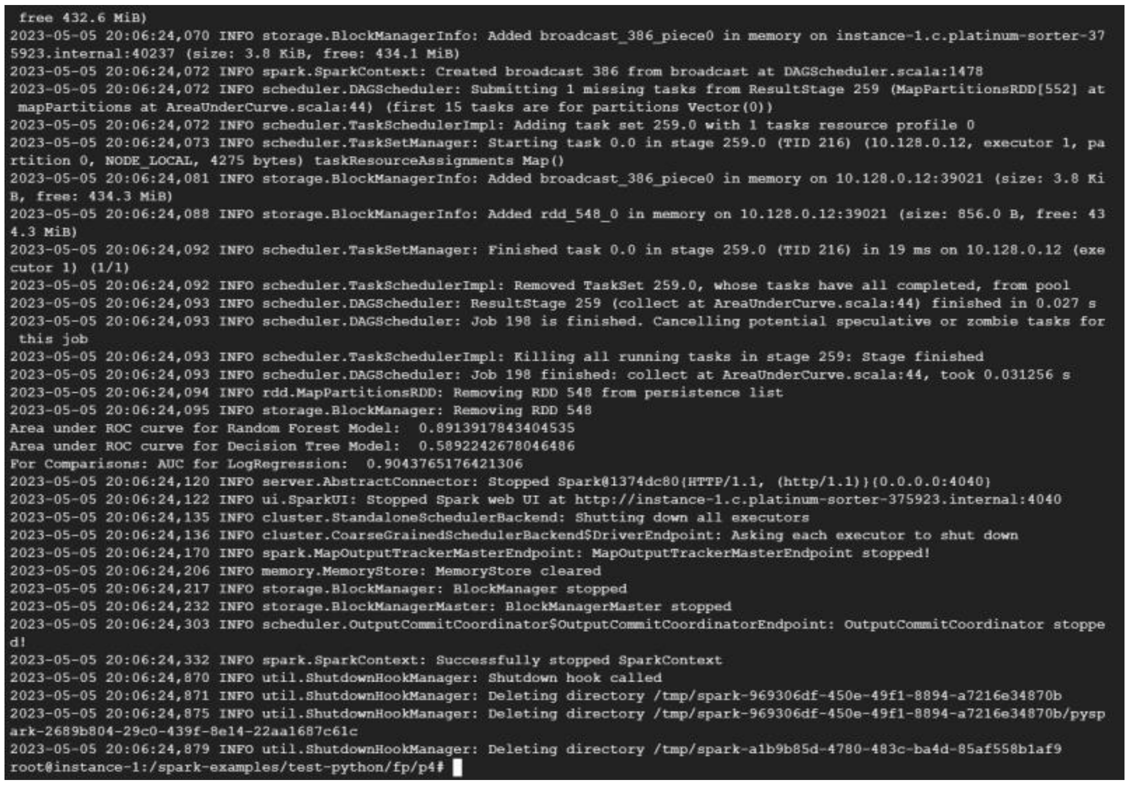
What we need to do for this task was logistic regression, which was also implemented using the PySpark 'LogisticRegression' class. The output of the logistic regression model was measured using the area under the receiver operating characteristic curve (ROC AUC) that could measure the ability of the model to distinguish between individuals with high and low income based on their characteristics. In this case, the code had produced an AUC score of 0.904, which indicated that the Logistic Regression model had good performance in predicting the income of the individuals.

P4: Random Forest and Decision Tree on Census Income Data

In this part, we would revisit the third part's problem but instead employ Random Forest Classifier and Decision Tree Classifier using Apache Spark ML/MLlib. This approach would enable us to compare the performance of different machine learning algorithms on the same dataset, further enhancing our understanding of their effectiveness and suitability for specific tasks.

Mmodel Explanation and Evaluation

The script utilized another two different classification models: Random Forest, and Decision Tree classifiers. Random forest is an ensemble method that combines multiple decision trees and averages their predictions, making it more robust and less prone to overfitting. Decision tree classifier is simple yet powerful model that recursively split the feature space based on feature importance. Both of the two models were trained on the pre-processed and standardized features in the training dataset,



```
 free 432.6 MiB)
2023-05-05 20:06:24,070 INFO storage.BlockManagerInfo: Added broadcast_386_piece0 in memory on instance-1.c.platinum-sorter-37
5923.internal:40237 (size: 3.8 KiB, free: 434.1 MiB)
2023-05-05 20:06:24,072 INFO spark.SparkContext: Created broadcast 386 from broadcast at DAGScheduler.scala:1478
2023-05-05 20:06:24,072 INFO scheduler.DAGScheduler: Submitting 1 missing tasks from ResultStage 259 (MapPartitionsRDD[552] at
 mapPartitions at AreaUnderCurve.scala:44) (first 15 tasks are for partitions Vector(0))
2023-05-05 20:06:24,072 INFO scheduler.TaskSchedulerImpl: Adding task set 259.0 with 1 tasks resource profile 0
2023-05-05 20:06:24,073 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 259.0 (TID 216) (10.128.0.12, executor 1, pa
rtition 0, NODE_LOCAL, 4275 bytes) taskResourceAssignments Map()
2023-05-05 20:06:24,081 INFO storage.BlockManagerInfo: Added broadcast_386_piece0 in memory on 10.128.0.12:39021 (size: 3.8 Ki
B, free: 434.3 MiB)
2023-05-05 20:06:24,088 INFO storage.BlockManagerInfo: Added rdd_548_0 in memory on 10.128.0.12:39021 (size: 856.0 B, free: 43
4.3 MiB)
2023-05-05 20:06:24,092 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 259.0 (TID 216) in 19 ms on 10.128.0.12 (exe
cutor 1) (1/1)
2023-05-05 20:06:24,092 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 259.0, whose tasks have all completed, from pool
2023-05-05 20:06:24,093 INFO scheduler.DAGScheduler: ResultStage 259 (collect at AreaUnderCurve.scala:44) finished in 0.027 s
2023-05-05 20:06:24,093 INFO scheduler.DAGScheduler: Job 198 is finished. Cancelling potential speculative or zombie tasks for
 this job
2023-05-05 20:06:24,093 INFO scheduler.TaskSchedulerImpl: Killing all running tasks in stage 259: Stage finished
2023-05-05 20:06:24,093 INFO scheduler.DAGScheduler: Job 198 finished: collect at AreaUnderCurve.scala:44, took 0.031256 s
2023-05-05 20:06:24,094 INFO rdd.MapPartitionsRDD: Removing RDD 548 from persistence list
2023-05-05 20:06:24,095 INFO storage.BlockManager: Removing RDD 548
Area under ROC curve for Random Forest Model:  0.8913917843404535
Area under ROC curve for Decision Tree Model:  0.5892242678046486
For Comparisons: AUC for LogRegression:  0.9043765176421306
2023-05-05 20:06:24,120 INFO server.AbstractConnector: Stopped Spark@1374dc80{HTTP/1.1, (http/1.1)}{0.0.0.0:4040}
2023-05-05 20:06:24,122 INFO ui.SparkUI: Stopped Spark web UI at http://instance-1.c.platinum-sorter-375923.internal:4040
2023-05-05 20:06:24,135 INFO cluster.StandaloneSchedulerBackend: Shutting down all executors
2023-05-05 20:06:24,136 INFO cluster.CoarseGrainedSchedulerBackend$DriverEndpoint: Asking each executor to shut down
2023-05-05 20:06:24,170 INFO spark.MapOutputTrackerMasterEndpoint: MapOutputTrackerMasterEndpoint stopped!
2023-05-05 20:06:24,206 INFO memory.MemoryStore: MemoryStore cleared
2023-05-05 20:06:24,217 INFO storage.BlockManager: BlockManager stopped
2023-05-05 20:06:24,232 INFO storage.BlockManagerMaster: BlockManagerMaster stopped
2023-05-05 20:06:24,303 INFO scheduler.OutputCommitCoordinator$OutputCommitCoordinatorEndpoint: OutputCommitCoordinator stoppe
d!
2023-05-05 20:06:24,332 INFO spark.SparkContext: Successfully stopped SparkContext
2023-05-05 20:06:24,870 INFO util.ShutdownHookManager: Shutdown hook called
2023-05-05 20:06:24,871 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-969306df-450e-49f1-8894-a7216e34870b
2023-05-05 20:06:24,875 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-969306df-450e-49f1-8894-a7216e34870b/pysp
ark-2689b804-29c0-439f-8e14-22aa1687c61c
2023-05-05 20:06:24,879 INFO util.ShutdownHookManager: Deleting directory /tmp/spark-a1b9b85d-4780-483c-ba4d-85af558b1af9
root@instance-1:/spark-examples/test-python/fp/p4# 
```

Figure 4: Census with RF&DT

and predictions were made on the test dataset using the 'transform()' function. The performance of the random forest, and decision tree models was evaluated using the area under the receiver operating characteristic (ROC) curve, we got the AUC of Random Forest was 0.89, and the AUC of Decision Tree was 0.59. Compared with the AUC score of Logistic Regression we got from P3(0.904), we could say that Logistic Regression performed best among these three models in predicting the personal income, and the Decision Tree made the worth performance. This could be because Decision Trees were prone to overfitting, and could be affected by the randomness introduced during the tree construction process. However, for the results we got for now, we could not conclude that Logistic Regression was the most suitable model for the dataset, we still need to consider other metrics such as precision, recall and F1 scores.

Conclusion

In this project, we were expected to utilize Apache Spark for data processing jobs and applied various machine learning algorithms to different datasets, and then showed their performance with Area Under ROC curve score. The project was divided into four distinct parts, each focused on a specific aspect of data processing and machine learning method. The project showed us the power and versatility of Apache Spark for handling data processing jobs.