



Table of Content

Topic	page
Preparing the Data	3-5
Random Forest Classifier	6
GBT Classifier	7
RandomForestClassifier GBTClassifier Comparison	8-9
Draw the ROC curves of testing results	10

Remarks:

- At the first part of preparing the data the handling of the training data and the test data occurs simultaneously.
- We have decided to remove the "?" as atrium asked.
- The classifiers parameters might change a little bit from one run to another.

Sources of information:

GBT articles:

<https://vagifaliyev.medium.com/a-hands-on-explanation-of-gradient-boosting-regression-4cfe7cfd9e>

random forest article:

<https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/>

Preparing the Data

Machine learning classification models that are based on "test data" or "labeled data" have a significant first stage before even computing the data.

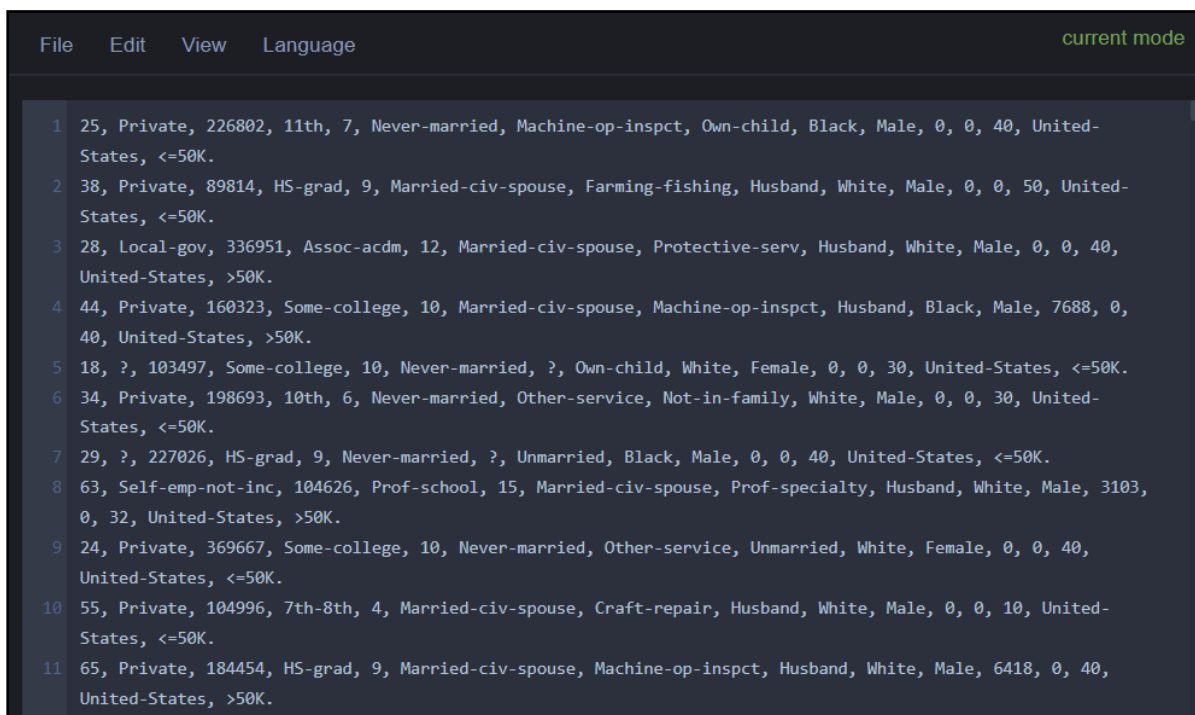
In that first stage we have a raw data, in this case a csv files, that have different attributes, some of them will be our features and one of them is the label, according to those the ML model can "learn" if his predictions are correct or if it have to change the weight of each attribute so that in his future predictions it will be able to predict correctly.

the obvious conclusion is that the more data our model is trained by, the better predictions it will be able to make.

The data that we receive is different from what our model can be trained by.

we must clean the data and convert it so the algorithm could work with it.

At first, we received the data as a simple CSV file that represent a relational data:



1	25, Private, 226802, 11th, 7, Never-married, Machine-op-inspct, Own-child, Black, Male, 0, 0, 40, United-States, <=50K.
2	38, Private, 89814, HS-grad, 9, Married-civ-spouse, Farming-fishing, Husband, White, Male, 0, 0, 50, United-States, <=50K.
3	28, Local-gov, 336951, Assoc-acdm, 12, Married-civ-spouse, Protective-serv, Husband, White, Male, 0, 0, 40, United-States, >50K.
4	44, Private, 160323, Some-college, 10, Married-civ-spouse, Machine-op-inspct, Husband, Black, Male, 7688, 0, 40, United-States, >50K.
5	18, ?, 103497, Some-college, 10, Never-married, ?, Own-child, White, Female, 0, 0, 30, United-States, <=50K.
6	34, Private, 198693, 10th, 6, Never-married, Other-service, Not-in-family, White, Male, 0, 0, 30, United-States, <=50K.
7	29, ?, 227026, HS-grad, 9, Never-married, ?, Unmarried, Black, Male, 0, 0, 40, United-States, <=50K.
8	63, Self-emp-not-inc, 104626, Prof-school, 15, Married-civ-spouse, Prof-specialty, Husband, White, Male, 3103, 0, 32, United-States, >50K.
9	24, Private, 369667, Some-college, 10, Never-married, Other-service, Unmarried, White, Female, 0, 0, 40, United-States, <=50K.
10	55, Private, 104996, 7th-8th, 4, Married-civ-spouse, Craft-repair, Husband, White, Male, 0, 0, 10, United-States, <=50K.
11	65, Private, 184454, HS-grad, 9, Married-civ-spouse, Machine-op-inspct, Husband, White, Male, 6418, 0, 40, United-States, >50K.

adult.Test file: the test data with 16,281 samples.

We can see that each row represents a person, with 14 different attributes, some of the data is missing and filled with "?" (Question mark).

Preparing the Data

In order to be able to compute the data with the ML algorithm first we have to load it as RDD object so we could manipulate it with RDD actions.

```
#create the spark sc object that will allow us to use the spark operations
spark = SparkSession.builder \
    .appName("Adult Income Classification") \
    .config("spark.driver.memory", "8g") \
    .config("spark.executor.memory", "16g") \
    .getOrCreate()

#debugged a problem with this line i found on the internet...
spark.conf.set("spark.worker.timeout", "600s")

# Load the training data and test data to a csv spark RDD variable
training_data = spark.read.csv("adult.data", header=True, inferSchema=True)
test_data = spark.read.csv("adult.test", header=True, inferSchema=True)
```

now that we have a Data Frame that contain our Data we can clean and prepare it and make it numerical so that :

```
the schema of the data loaded to the training Data Spark RDD variable:
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|age|    workclass|  fnlwgt|  education|education-num|  marital-status|  occupation|  relationship|  race|  sex|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 38|    Private| 89814.0|    HS-grad|         9.0| Married-civ-spouse|  Farming-fishing|    Husband|  White|  M
ale|    0.0|    0.0|    50.0| United-States| <=50K.|
| 28|    Local-gov|336951.0|  Assoc-acdm|        12.0| Married-civ-spouse|  Protective-serv|    Husband|  White|  M
ale|    0.0|    0.0|    40.0| United-States| >50K.|
| 44|    Private|160323.0|  Some-college|        10.0| Married-civ-spouse|  Machine-op-inspct|    Husband|  Black|  M
ale| 7688.0|    0.0|    40.0| United-States| >50K.|
| 18|    ?|103497.0|  Some-college|        10.0|  Never-married|    ?|  Own-child|  White|  Fem
```

filtering the "?" data from all the cells that hold that value, so that it will not impact our model.

```
filtered_test_data = training_data.filter((col("workclass") != " ?") & (col("education") != " ?") &
                                           (col("marital-status") != " ?") & (col("occupation") != " ?") &
                                           (col("relationship") != " ?") & (col("race") != " ?") &
                                           (col("sex") != " ?") & (col("native-country") != " ?")
)
```

Preparing the Data

Defining the **feature** columns and the creating the **label** column according to the income attribute in the training data set, alternating it to numerical so it can be waited in the end.

```
#indexing the catagorial features creating them numerical
catagorial_features_indexer = [StringIndexer(inputCol=col, outputCol=col+"_indexed") for col in catagorial_features]
label_indexer = StringIndexer(inputCol="income", outputCol="label")

#pipeline- shared memmory fot the indexers for the nodes in the cluster
catagorial_features_pipeline = Pipeline(stages=catagorial_features_indexer)

#defining the lable "income" according to that we will build our ML modle and train it
training_data_Numerical = label_indexer.fit(training_data).transform(training_data)
test_data_Numerical = label_indexer.fit(test_data).transform(test_data)
```

at this point we have our Data RDD ready to be computed by the Random Forest Classifier, so when we will handle the real data we could classify it.

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										
age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week	label	workclass_indexed	education_indexed	marital-status_indexed	occupation_indexed
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+										
50	83311.0	13.0	0.0	0.0	13.0	0.0	1.0	2.0	0.0	
2.0		0.0	0.0	0.0	0.0					
38	215646.0	9.0	0.0	0.0	40.0	0.0	0.0	0.0	2.0	
9.0		1.0	0.0	0.0	0.0					
53	234721.0	7.0	0.0	0.0	40.0	0.0	0.0	5.0	0.0	
9.0		0.0	1.0	0.0	0.0					
28	338409.0	13.0	0.0	0.0	40.0	0.0	0.0	2.0	0.0	
0.0		4.0	1.0	1.0	9.0					
37	284582.0	14.0	0.0	0.0	40.0	0.0	0.0	3.0	0.0	
2.0		4.0	0.0	1.0	0.0					
49	160187.0	5.0	0.0	0.0	16.0	0.0	0.0	10.0	5.0	
5.0		1.0	1.0	1.0	11.0					
52	209642.0	9.0	0.0	0.0	45.0	1.0	1.0	0.0	0.0	

In this section we have used the Spark RDD object to make our program to be capable to scale up and work on a Cluster of nodes, we can see that handling the data using the RDD interface was simple.

by making some adjustments to the data it can now be processed by the classifiers, in larger scales the data probably will be transmitted by DataStream, but in our case static RDD actions made it possible to be processed .

Random Forest Classifier

Random Forest is one of the most popular and commonly used algorithms by Data Scientists. Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables, as in the case of regression, and categorical variables, as in the case of classification. It performs better for classification and regression tasks. In this tutorial, we will understand the working of random forest and implement random forest on a classification task.

In our code we have used the RDD training data and the Random Forest object we had to pass it those parameters of the max num of trees and max depth so it will work properly.

Then we used our 'label' as our evaluator in the training part.

We ran the cross validation to check if we really had the best model.

In the end we have a cross validation Random Forest classifier.

```
# Define the RandomForestClassifier model
randomForest_classifierOBJ = RandomForestClassifier(numTrees=100, maxDepth=10, maxBins=64)

# Define the grid of hyperparameters to search over
param_grid = ParamGridBuilder() \
    .addGrid(randomForest_classifierOBJ.numTrees, [10, 50, 100]) \
    .addGrid(randomForest_classifierOBJ.subsamplingRate, [0.5, 0.8]) \
    .addGrid(randomForest_classifierOBJ.featureSubsetStrategy, ['sqrt', 'log2']) \
    .build()

#creating the assemblers with the training_data_Numerical2 and test_data_Numerical2
assembler = VectorAssembler(inputCols=featurelist, outputCol="features")
assembled_trainig_data = assembler.transform(training_data_Numerical2)
assembled_test_data = assembler.transform(test_data_Numerical2)

# Define the evaluation metric
evaluator = BinaryClassificationEvaluator(labelCol = 'label')

# Define the evaluation metric
cv = CrossValidator(estimator=randomForest_classifierOBJ, estimatorParamMaps=param_grid, evaluator=evaluator)

# Fit the cross-validation model on the training data
cv_model = cv.fit(assembled_trainig_data)

# Use the best model to make predictions on the validation set
predictions = cv_model.transform(assembled_test_data)

assembled_test_data.show()
assembled_trainig_data.show()
```

Random Forest Classifier

```
#showing the accuracy and the confusion matrix of my model
evaluator = MulticlassClassificationEvaluator(predictionCol="prediction", labelCol="label", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("the confusion matrix and the accuracy of the model: \n\n")
print("Accuracy = %g" % (accuracy))

confusion_matrix = predictions.groupBy("label").pivot("prediction").count().na.fill(0)
confusion_matrix.show()

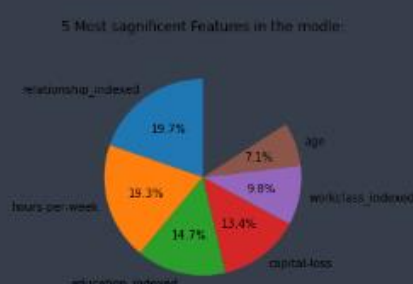
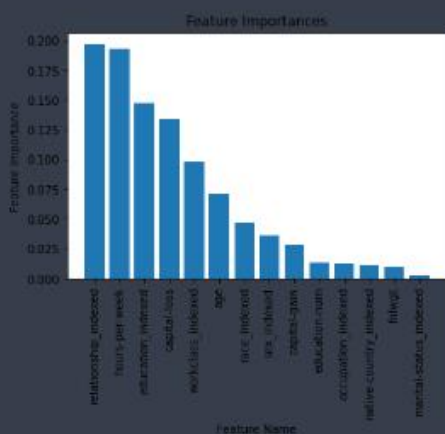
print("best HyperParameter's for the model: \n")
# Extract the values of the best hyperparameters
best_rf_model = cv_model.bestModel
best_num_trees = best_rf_model.getNumTrees
best_subsampling_rate = best_rf_model.getSubsamplingRate()
best_feature_subset_strategy = best_rf_model.getFeatureSubsetStrategy()

# Print out the values of the hyperparameters and the AUC score
print("best number Trees = ", best_num_trees)
print("best sub sampling Rate = ", best_subsampling_rate)
print("best feature Subset Strategy = ", best_feature_subset_strategy)
```

We received our Random Forest Model.

From the model we can extract data such as the most significant attributes for determining if a person income is higher or lower than 50,000 (which is our label).

Confusion matrix and the model best parameters. (Best num of trees, best sub sampling rate and best strategy). Graphically presenting the most significant features.



```
+-----+-----+-----+
|label|  0.0|  1.0|
+-----+-----+
|  0.0|11616| 818|
|  1.0| 1717|2129|
+-----+-----+-----+
```

Random forest classifier after Cross validation: accuracy F1 and Auc score:

Random forest classifier Accuracy after Cross validation:

Accuracy = 0.844287

Random forest classifier F1 Score after Cross validation:

F1 score: 0.8367

Random forest classifier Area Under the curve after Cross validation:

AUC: 0.9009

best HyperParameter's for the model:

best number Trees = 100

best sub sampling Rate = 0.8

best feature Subset Strategy = sqrt

GBT Classifier

In Gradient Boosting Trees model, each predictor tries to improve on its predecessor by reducing the errors. But the fascinating idea behind Gradient Boosting is that instead of fitting a predictor on the data at each iteration, it fits a new predictor to the residual errors made by the previous predictor.

```
#Defining the Gradient Boosted Trees GBT MODEL
gbt = GBTClassifier(labelCol="label", featuresCol="features", maxBins=50)

# Train the model on the training data
model = gbt.fit(assembled_trainig_data)

# Generate predictions for the test data
predictions = model.transform(assembled_test_data)

# Evaluate accuracy
evaluator_acc = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName='accuracy')
accuracy = evaluator_acc.evaluate(predictions)

# Evaluate F1 score
evaluator_f1 = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName='f1_score')
f1_score = evaluator_f1.evaluate(predictions)

auc = evaluator.evaluate(predictions, {evaluator_acc.metricName: 'areaUnderROC'})

print("the GBT classifier parameter's are: \n")
print('Accuracy:', accuracy)
print('F1 Score:', f1_score)
print('Area Under Curve:', auc)
```

the GBT classifier parameter's are:

GBT model Accuracy: 0.8427518427518428

F1 Score: 0.8356821786611337

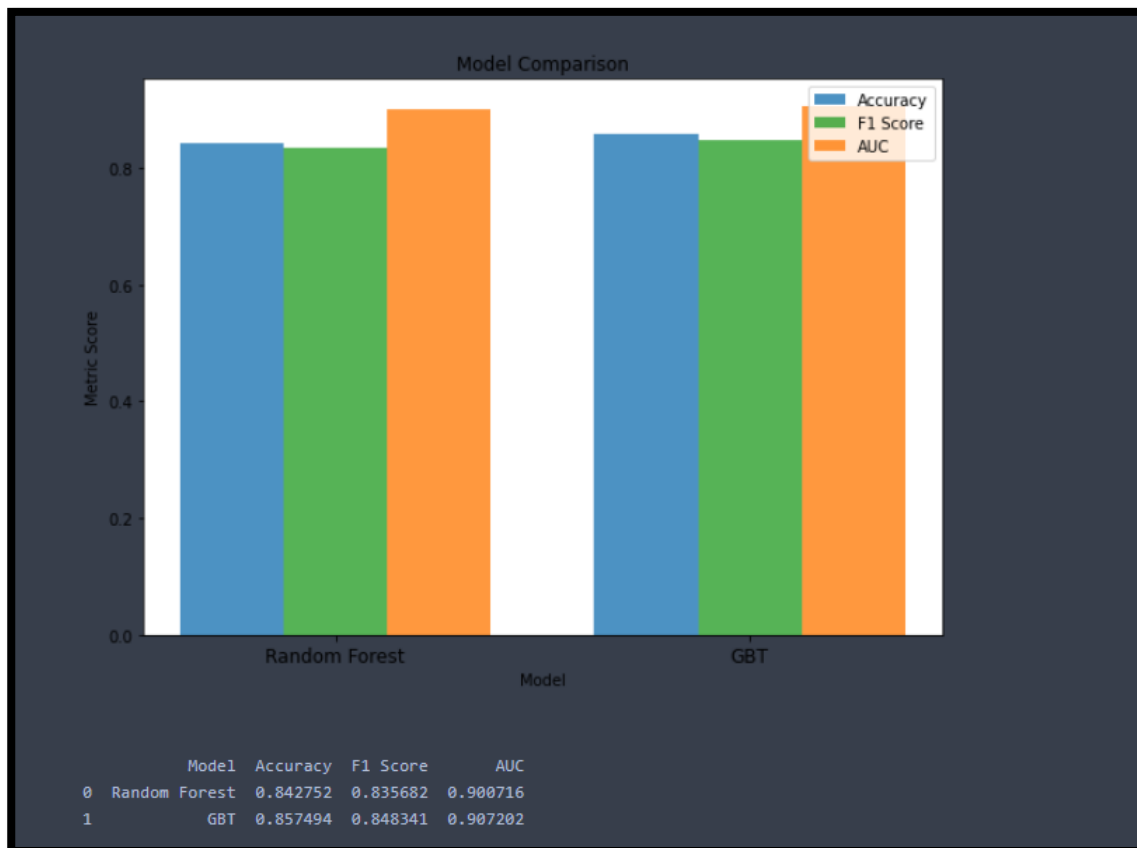
Area Under Curve: 0.9007158211372686

RandomForestClassifier GBTClassifier Comparison

in this section we will present the results of both Random Forest and GBT classifiers.

Note that in each round the results might differ a little from past runs.

But the overall trend of both classifiers is almost the same, both are able to predict if a person will earn more or less than 50K is at ~85% accuracy, as that the F1 and the AUC are very similar as well.



Model	Accuracy	F1_Score	AUC
GBT	0.8427	0.8356	0.900716
CV Random Forest	0.8574	0.8483	0.907202

In this particular run of both classifiers, we can clearly see that the Random Forest classifier outperformed the GBT classifier in all three metrics: accuracy, F1 score, and AUC.

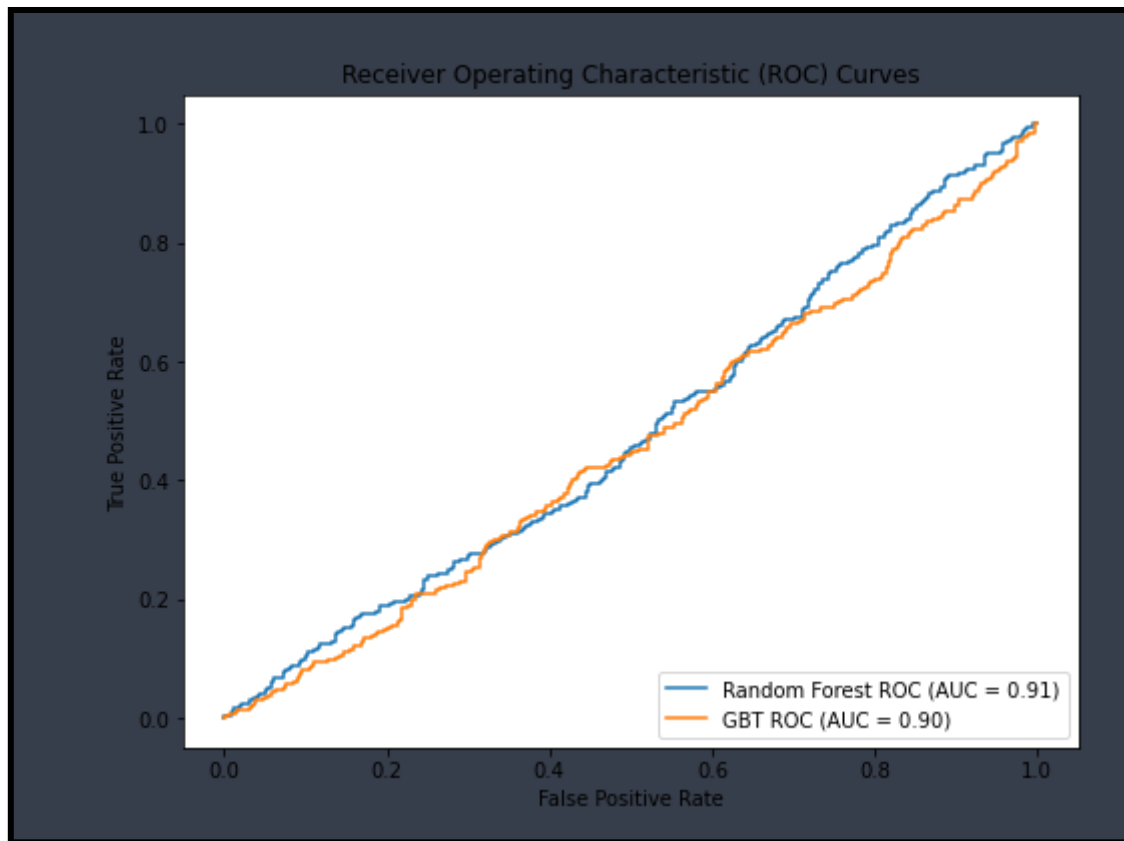
BUT still at the bottom line all three parameters are almost the same with little differences between them, and I can .

Draw the ROC curves of testing results:

An ROC curve (receiver operating characteristic curve) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters:

- True Positive Rate
- False Positive Rate

$$TPR = \frac{TP}{TP + FN} \quad FPR = \frac{FP}{FP + TN}$$



From what we can see our 2 classifiers have almost the same curve, when the random forest classifier is a little bit ahead by classifying more TP against its FP.