**Classification Tree for Flight Status**

**1. Upload, explore, clean, and preprocess data for classification tree.**

**a. Create the flight\_df data frame by uploading the original data set into Python. Determine and present in this report the data frame dimensions, i.e., number of rows and columns.**

To upload the FlightDelays(1).csv file, you can into Python, we can use the pandas library as follows:

Text

Description automatically generated

The data frame dimensions for the flight delay data set is shown below:



**b. Remove ‘DEST’ and ‘ORIGIN’ variables from the flight\_df data frame. Then, display the column data types in flight\_df, provide and briefly explain them in your report.**

To remove the 'DEST' and 'ORIGIN' variables from the flight\_df data frame, we can use the ‘*drop’* method as follows:

**Table

Description automatically generated**

The output shows 9 columns as compared to 11 columns in the original data frame.

To display the column data types in *flight\_df*, you can use the *dtypes* attribute as follows:

Graphical user interface

Description automatically generated with medium confidence

Based on the output above, it is visible that the 'DEST' and 'ORIGIN' variables have been removed from the flight\_df data frame. We can also see that 'Carrier' and 'Flight\_Arriving\_Status' variables are of type object, which typically indicates that they are categorical variables. We may need to convert them to dummy variables to move forward.

**c. You leave the outcome variable ‘FL\_STATUS’ unchanged in flight\_df. However, for the ‘CARRIER’ predictor variable, you need to convert it into binary variables. Display in Python the modified column data types, provide and briefly explain them in your report.**

To convert the ‘CARRIER’ predictor variable into binary variables:

Text

Description automatically generated with medium confidence

This will create a new binary column for each unique value in the 'CARRIER' column.

To display the modifies column data types:

Graphical user interface, application

Description automatically generated

The 'CARRIER' column has been replaced with 8 binary columns (one for each carrier), with data type uint8. uint8 is a data type that represents an unsigned 8-bit integer. The values in these columns represent whether or not the flight was operated by that carrier, with 1 indicating that the flight was operated by that carrier and 0 otherwise.

**d. Display in Python and provide in your report the first 10 records of the modified flight\_df data frame. Briefly explain the outcome and predictors in this case.**

To display the first 10 records of the modified *flight\_df* data frame in Python:



The outcome:

Table

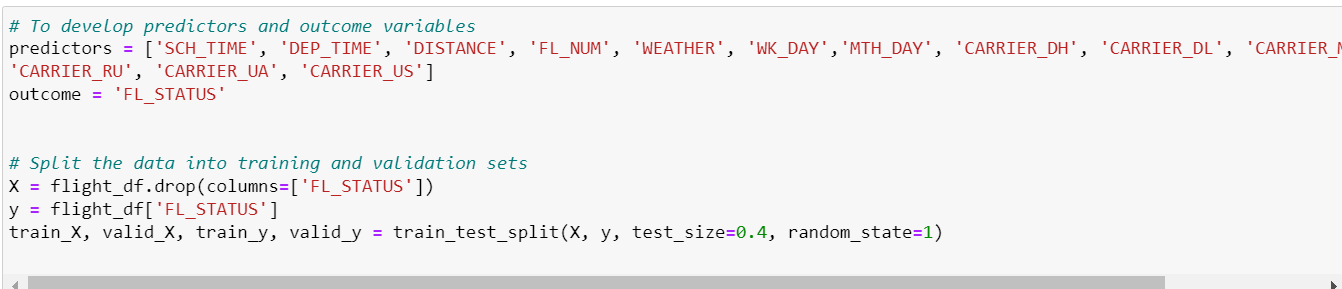
Description automatically generated

First 10 rows of the Data Frame can be useful for quickly checking the structure and contents of the Data Frame.

**2. Develop a classification tree for the Flight Delays case.**

**a. Develop in Python the predictor variables (14 variables) and outcome variable (‘FL\_STATUS’), partition the data set (80% for training and 20% for validation partitions). Train a classification tree model using DecisionTreeClassifier() with the training data set and the following tree control parameters: (a) maximum depth (number of splits) equals 4; (b) minimum impurity decrease per split of 0.001; and (c) minimum number of node records (samples) to split equals to 30. Use plotDecisionTree() with the feature\_names and class\_names parameters to display the classification tree in Python and present it in your report.**

We will use the modified *flight\_df* data frame with binary variables for the CARRIER column and will split the data into training and validation partitions using the *train\_test\_split* function from the *sklearn* module:

****

Next, we can train a classification tree model using *DecisionTreeClassifier()* from the *sklearn* module with the specified tree control parameters and plot the classification tree by using *plotDecisionTree()*:

Text

Description automatically generated

Outcome:

A picture containing diagram

Description automatically generated

**b. Using the classification tree, explain the outcome (‘FL\_STATUS’) of a flight if the weather (‘WEATHER’) is in good flying condition, departure time (‘DEP\_TIME’) is 1450 (2:50 pm), and scheduled time (‘SCH\_TIME’) is 1435 (2:35 pm).**

The outcome (‘FL\_STATUS’) of a flight if the weather (‘WEATHER’) is in good flying condition, departure time (‘DEP\_TIME’) is 1450 (2:50 pm), and scheduled time (‘SCH\_TIME’) is 1435 (2:35 pm) is “Ontime”.

**c. Identify and display in Python confusion matrices for training and validation partitions. Present them in your report and comment on accuracy (misclassification) rate for both partitions and explain if there is a possibility of overfitting.**

The code to generate the confusion matrices for training and validation partitions:

**Graphical user interface, text

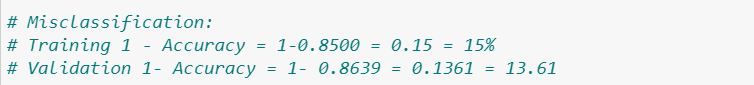
Description automatically generated**

Outcome:

Text

Description automatically generated with low confidence

High accuracy on the validation partition can indicate a good fit for the model, but it's still important to consider other factors such as the representativeness of the data and the specific evaluation metrics used. The misclassification is close to the training set which means that model is not overfit.



**d. Using the trained classification tree, make classification of flight status (‘delayed’ or ‘ontime’) for the following two new flight records: Present and briefly explain the classification results in your report.**

**A picture containing text

Description automatically generated**

**Graphical user interface

Description automatically generated with low confidence**

Outcome**:**

**Table

Description automatically generated**

**3. Apply grid search to improve classification results.**

**a. Use the GridSearchCV() algorithm in Python to improve (optimize) the classification tree control parameters. Consider the following control parameters: (a) maximum depth (number of splits) in the range from 2 to 30; (b) minimum impurity decrease per split of 0, 0.0005, and 0.001; and (c) minimum number of node records (samples) to split in the range from 5 to 30. Do not use the initial guess grid search, and directly apply the improved grid search. Provide in your report the improved parameters and display in Python the associated classification tree. Display the confusion matrices for training and validation partitions for the improved classification tree.**

To improve (optimize) the classification tree control parameters using GridSearchCV() algorithm:

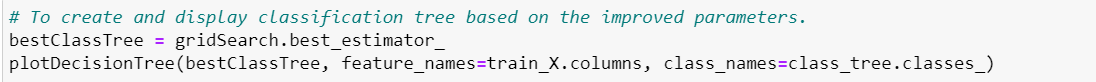
**Graphical user interface, text, application

Description automatically generated**

Outcome**:**

****

To display in Python the associated classification tree:

****

Outcome:

**A picture containing diagram

Description automatically generated**

The confusion matrices are as follows:

Graphical user interface, text, application, email

Description automatically generated

**b. Present and compare in your report the validation confusion matrices for the classification results in questions 2c and 3a. Using the accuracy value (misclassification rate), which classification tree model would you recommend using for making predictions in this case of flight status (‘delayed’ or ‘ontime’)? Briefly explain your answer.**

**In 2c:**

Text

Description automatically generated with low confidence

**In 3a:**

Graphical user interface, text, application, email

Description automatically generated

When comparing the accuracy of the two validation partitions obtained from the best classification tree with grid search and the one obtained from the smaller tree, the former showed better results with an 84.90% accuracy rate, while the latter achieved 84.55%. This suggests that the more complex model was able to capture the underlying patterns in the data more effectively, resulting in better predictions. However, it's worth noting that the difference in accuracy between the two models was relatively small, and there's a possibility that the larger model may have overfit the data to some extent.