Case Study #2: Classification Tree for Flight Status

The *FlightDelays.csv* file contains a data set with the information on all commercial flights departing the Washington, DC area and arriving at one of the New York area airports. For each flight, there is information on the departure and arrival airports, the distance of the route, the scheduled time and day of the flight, and so on. The outcome that you need to predict is the flight arriving status, i.e., whether a flight is delayed or on time (*FL_STATUS*). A delay is defined as an arrival that is at least 15 minutes later than scheduled. The table below descries each of the columns in the data set.

Variables	***Description of Variables***
SCH_TIME	Scheduled departure time.
CARRIER	Carrier abbreviation: CO=Continental, DH=Discovery Airways, DL= Delta Airlines, MQ=American Eagle, OH=PSA Airlines, RU=AirBridge Cargo, UA=United Airlines, US=US Airways.
DEP_TIME	Actual flight departure time.
DEST	Abbreviation of the destination airport in New York area: EWR=Newark, JFK=John F. Kennedy, LGA= LaGuardia.
DISTANCE	Distance of the route in miles.
FL_NUM	Flight number.
ORIGIN	Abbreviation of the DC area airport: BWI=Baltimore /Washington, DCA=Ronald Reagan Washington, and IAD=Dulles International.
WEATHER	Weather condition for a flight: good flying condition = 0, poor flying condition = 1.
WK_ DAY	Day of the week, from Monday = 1 through Sunday = 7.
MTH_DAY	Day of the month, from 1 through 31.
FL_STATUS	Flight arriving status, two classes: 'delayed' and 'ontime'.

Questions

1. Upload, explore, clean, and preprocess data for classification tree.
    a. Create a *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.
    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.
    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. For that, change the 'CARRIER' data type from 'object' to 'category', and then convert this categorical variable into dummy variables. Display in Python the modified column data types and provide them in your report.
    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.

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 (70% for training and 30% 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 split levels) equals 5; (b) minimum impurity decrease per split of 0.001; and (c) minimum number of node records (samples) to split equals to 10. Use *plotDecisionTree()* with the *feature_names* and *class_names* parameters to display the classification tree in Python and present it in your report.
    b. Using the classification tree, explain the classification outcome ('FL_STATUS') of a flight if the weather ('WEATHER') is in good flying condition, departure time ('DEP_TIME') is 1605 (4:05 pm), scheduled time ('SCH_TIME') is 1510 (3:10 pm), and the flight happens on the 28th day of the month.
    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.
    d. Using the trained classification tree, make classification of flight status ('delayed' or 'ontime') for the following two new flight records:

	SCH_TIME	DEP_TIME	DISTANCE	FL_NUM	WEATHER	WK_DAY	MTH_DAY	CARRIER_DH	\
0	1230	1240	214	808	1	4	20	0	
1	2050	2105	199	4976	0	5	30	0	

	CARRIER_DL	CARRIER_MQ	CARRIER_OH	CARRIER_RU	CARRIER_UA	CARRIER_US	\
0	0	0	0	0	1	0	
1	1	0	0	0	0	0	

    Present and briefly explain the classification results in your report.

3. Apply grid search and ensemble trees 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 split levels) in the range from 2 to 25; (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 20. 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.
    b. Present and compare in your report the validation confusion matrices for the classification results in questions 2b and 3a. Using the accuracy value (misclassification rate), which classification tree model would you recommend using for making predictions of the flight status ('delayed' or 'ontime')? Briefly explain your answer.