CA03 Decision Trees Write-Up

Q.1.1 Why does it make sense to discretize columns for this problem?
Discretizing the data columns makes sense because it helps to group fluctuations in the data that have the same impact on the model. For example, in the age_bin category, someone at age 40 will have the same impact on the model as someone at age 55; therefore it makes the data more readable and easier for the model to understand because it groups the small fluctuations that have the same impact, and allows the model to identify where the true impacts lie.

Q.1.2 What might be the issues (if any) if we DID NOT discretize the column
Discretization of data did lead to some extra steps in this process because we needed to use the LabelEncoder() function to turn the categorical variables into numerical ones for the model to read. However, if we had not discretized the data, then the data would have had a lot of fluctuations between values in a column and the model would need to do more work in identifying what fluctuations have a real impact on the decision and which do not.

7.1 Decision Tree Hyper-parameter variation vs. performance

Decision Tree Hyperparameter Variations Vs. Tree Performance							
========== Complete the following table ==========							
Hyperparameter Variations				Model Perfromance			
Split Criteria (Entropy or Gini)	Minimum Sample Split	Minimum Sample Leaf	Maximum Depth	Accuracy	Recall	Precision	F1 Score
Entropy	*Split Value 1*	*Leaf Value 1*	*Depth 1*	0.76	0	0	0
	Split Value 1	*Leaf Value 1*	*Depth 2*	0.76	0	0	0
	Split Value 1	*Leaf Value 2*	*Depth 1*	0.76	0	0	0
	Split Value 2	*Leaf Value 1*	*Depth 1*	0.76	0	0	0
Gini Impurity	*Split Value 1*	*Leaf Value 1*	*Depth 1*	0.76	0	0	0
	Split Value 1	*Leaf Value 1*	*Depth 2*	0.76	0	0	0
	Split Value 1	*Leaf Value 2*	*Depth 1*	0.76	0	0	0
	Split Value 2	*Leaf Value 1*	*Depth 1*	0.76	0	0	0

Q.8.1 How long was the time to train the model?

```
#DecisionTreeClassifier Model
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier(max_depth=3, random_state=101,
                               max_features = None, min_samples_leaf=5)
dtree.fit(X_train, Y_train)
y_pred=dtree.predict(X_test)
%time
```

```
CPU times: user 2 µs, sys: 1 µs, total: 3 µs
Wall time: 5.01 µs
```

CA03 Decision Trees Write-Up

Q.8.2 Did you find the best tree?
When I changed the hyperparameters, I assigned each tree to a different variable (a variation of the y_pred variable) name so that the performance scores would correspond to the proper tree. However, all of the scores were the same across all eight trees. I am not sure why this occurred. For my best tree, I would consider the original model (featured in Q.7.1) to be the best tree because it has high accuracy and precision scores at 83% accuracy and 68% precision.

Q.8.3 Draw the graph of the best tree using GraphViz
Because my original model was the best tree, the graph can be seen in my notebook at Part 5.

Q.8.4 What makes it the best tree?
This tree is the best due to its strong performance scores, especially accuracy and precision. Additionally, it's max depth of 3 aids in not overfitting the model and the initial max leaf splits of 5 guarantees each leaf has a minimum size which avoids low-variance.

Q.10.1
For part 10, I was unsure how to test these values in the model. My initial idea was to put these values with their corresponding column headings into a csv file and read them in as a dataframe. Then apply that data as test data for the model. But these values did not include a Y value so I was not sure how to split the data into X_test and Y_test variables.