Activity_Build a decision tree

December 29, 2023

1 Activity: Build a decision tree

1.1 Introduction

A decision tree model can makes predictions for a target based on multiple features. Because decision trees are used across a wide array of industries, becoming proficient in the process of building one will help you expand your skill set in a widely-applicable way.

For this activity, you work as a consultant for an airline. The airline is interested in predicting whether a future customer would be satisfied with their services given customer feedback given previous customer feedback about their flight experience. The airline would like you to construct and evaluate a model that can accomplish this goal. Specifically, they are interested in knowing which features are most important to customer satisfaction.

The data for this activity includes survey responses from 129,880 customers. It includes data points such as class, flight distance, and in-flight entertainment, among others. In a previous activity, you utilized a binomial logistic regression model to help the airline better understand this data. In this activity, your goal will be to utilize a decision tree model to predict whether or not a customer will be satisfied with their flight experience.

Because this activity uses a dataset from the industry, you will need to conduct basic EDA, data cleaning, and other manipulations to prepare the data for modeling.

In this activity, you'll practice the following skills:

- Importing packages and loading data
- Exploring the data and completing the cleaning process
- Building a decision tree model
- Tuning hyperparameters using `GridSearchCV`
- Evaluating a decision tree model using a confusion matrix and various other plots

1.2 Step 1: Imports

Import relevant Python packages. Use `DecisionTreeClassifier`,`plot_tree`, and various imports from `sklearn.metrics` to build, visualize, and evaluate the model.

1.2.1 Import packages

[23]:
```
# Standard operational package imports
import numpy as np
import pandas as pd

# Important imports for modeling and evaluation
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix
from sklearn.metrics import recall_score, precision_score, f1_score,␣
↪accuracy_score
from sklearn.model_selection import GridSearchCV
import sklearn.metrics as metrics
from sklearn.tree import plot_tree

# Visualization package imports
import matplotlib.pyplot as plt
import seaborn as sns
```

1.2.2 Load the dataset

`Pandas` is used to load the **Invistico_Airline.csv** dataset. The resulting pandas DataFrame is saved in a variable named `df_original`. As shown in this cell, the dataset has been automatically loaded in for you. You do not need to download the .csv file, or provide more code, in order to access the dataset and proceed with this lab. Please continue with this activity by completing the following instructions.

[24]:
```
# RUN THIS CELL TO IMPORT YOUR DATA.

### YOUR CODE HERE ###

df_original = pd.read_csv("Invistico_Airline.csv")
```

Hint 1

Use a function from the pandas library to read in the csv file.

Hint 2

Use the `read_csv` function and pass in the file name as a string.

Hint 3

Use `pd.read_csv("insertfilenamehere")`.

1.2.3 Output the first 10 rows of data

[25]:
```
df_original.head(10)
```

[25]:
```
  satisfaction   Customer Type  Age   Type of Travel     Class  \
0    satisfied  Loyal Customer   65  Personal Travel       Eco
1    satisfied  Loyal Customer   47  Personal Travel  Business
2    satisfied  Loyal Customer   15  Personal Travel       Eco
3    satisfied  Loyal Customer   60  Personal Travel       Eco
4    satisfied  Loyal Customer   70  Personal Travel       Eco
5    satisfied  Loyal Customer   30  Personal Travel       Eco
6    satisfied  Loyal Customer   66  Personal Travel       Eco
7    satisfied  Loyal Customer   10  Personal Travel       Eco
8    satisfied  Loyal Customer   56  Personal Travel  Business
9    satisfied  Loyal Customer   22  Personal Travel       Eco

   Flight Distance  Seat comfort  Departure/Arrival time convenient  \
0              265             0                                  0
1             2464             0                                  0
2             2138             0                                  0
3              623             0                                  0
4              354             0                                  0
5             1894             0                                  0
6              227             0                                  0
7             1812             0                                  0
8               73             0                                  0
9             1556             0                                  0

   Food and drink  Gate location  …  Online support  Ease of Online booking  \
0               0              2  …               2                       3
1               0              3  …               2                       3
2               0              3  …               2                       2
3               0              3  …               3                       1
4               0              3  …               4                       2
5               0              3  …               2                       2
6               0              3  …               5                       5
7               0              3  …               2                       2
8               0              3  …               5                       4
9               0              3  …               2                       2

   On-board service  Leg room service  Baggage handling  Checkin service  \
0                 3                 0                 3                5
1                 4                 4                 4                2
2                 3                 3                 4                4
3                 1                 0                 1                4
4                 2                 0                 2                4
5                 5                 4                 5                5
```

```
6                  5                  0                  5                  5
7                  3                  3                  4                  5
8                  4                  0                  1                  5
9                  2                  4                  5                  3

   Cleanliness  Online boarding  Departure Delay in Minutes  \
0            3                2                           0
1            3                2                         310
2            4                2                           0
3            1                3                           0
4            2                5                           0
5            4                2                           0
6            5                3                          17
7            4                2                           0
8            4                4                           0
9            4                2                          30

   Arrival Delay in Minutes
0                       0.0
1                     305.0
2                       0.0
3                       0.0
4                       0.0
5                       0.0
6                      15.0
7                       0.0
8                       0.0
9                      26.0

[10 rows x 22 columns]
```

Hint 1

Use the `head()` function.

Hint 2

If only five rows are output, it is because the function by default returns five rows. To change this, specify how many rows `(n = )` you want to output.

1.3 Step 2: Data exploration, data cleaning, and model preparation

1.3.1 Prepare the data

After loading the dataset, prepare the data to be suitable for decision tree classifiers. This includes:

- Exploring the data
- Checking for missing values
- Encoding the data

- Renaming a column
- Creating the training and testing data

1.3.2 Explore the data

Check the data type of each column. Note that decision trees expect numeric data.

[26]:
```
df_original.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 129880 entries, 0 to 129879
Data columns (total 22 columns):
 #   Column                             Non-Null Count   Dtype
---  ------                             --------------   -----
 0   satisfaction                       129880 non-null  object
 1   Customer Type                      129880 non-null  object
 2   Age                                129880 non-null  int64
 3   Type of Travel                     129880 non-null  object
 4   Class                              129880 non-null  object
 5   Flight Distance                    129880 non-null  int64
 6   Seat comfort                       129880 non-null  int64
 7   Departure/Arrival time convenient  129880 non-null  int64
 8   Food and drink                     129880 non-null  int64
 9   Gate location                      129880 non-null  int64
 10  Inflight wifi service              129880 non-null  int64
 11  Inflight entertainment             129880 non-null  int64
 12  Online support                     129880 non-null  int64
 13  Ease of Online booking             129880 non-null  int64
 14  On-board service                   129880 non-null  int64
 15  Leg room service                   129880 non-null  int64
 16  Baggage handling                   129880 non-null  int64
 17  Checkin service                    129880 non-null  int64
 18  Cleanliness                        129880 non-null  int64
 19  Online boarding                    129880 non-null  int64
 20  Departure Delay in Minutes         129880 non-null  int64
 21  Arrival Delay in Minutes           129487 non-null  float64
dtypes: float64(1), int64(17), object(4)
memory usage: 21.8+ MB
```

Hint 1

Use the `dtypes` attribute on the DataFrame.

1.3.3 Output unique values

The `Class` column is ordinal (meaning there is an inherent order that is significant). For example, airlines typically charge more for ‘Business’ than ‘Eco Plus’ and ‘Eco’. Output the unique values in the `Class` column.

[27]:
```
df_original['Class'].unique()
```

[27]:
```
array(['Eco', 'Business', 'Eco Plus'], dtype=object)
```

Hint 1

Use the `unique()` function on the column `'Class'`.

1.3.4 Check the counts of the predicted labels

In order to predict customer satisfaction, verify if the dataset is imbalanced. To do this, check the counts of each of the predicted labels.

[28]:
```
df_original['satisfaction'].value_counts()
```

[28]:
```
satisfied       71087
dissatisfied    58793
Name: satisfaction, dtype: int64
```

Hint 1

Use a function from the pandas library that returns a pandas series containing counts of unique values.

Hint 2

Use the `value_counts()` function. Set the `dropna` parameter passed in to this function to `False` if you want to examine how many NaN values there are.

Question: How many satisfied and dissatisfied customers were there?

There are 71087 satisfied customers and 58793 dissatisfied customers.

Question: What percentage of customers were satisfied?

82% of customers are satisfied

1.3.5 Check for missing values

The sklearn decision tree implementation does not support missing values. Check for missing values in the rows of the data.

[29]:
```
df_original.isna().sum()
```

[29]:
```
satisfaction                       0
Customer Type                      0
Age                                0
Type of Travel                     0
Class                              0
Flight Distance                    0
Seat comfort                       0
```

```
Departure/Arrival time convenient      0
Food and drink                         0
Gate location                          0
Inflight wifi service                  0
Inflight entertainment                 0
Online support                         0
Ease of Online booking                 0
On-board service                       0
Leg room service                       0
Baggage handling                       0
Checkin service                        0
Cleanliness                            0
Online boarding                        0
Departure Delay in Minutes             0
Arrival Delay in Minutes             393
dtype: int64
```

Hint 1

Use the `isnull` function and the `sum` function.

Hint 2

To get the number of rows in the data with missing values, use the `isnull` function followed by the `sum` function.

Question: Why is it important to check how many rows and columns there are in the dataset?

To know how large the set is and it's also important to keep in mind when implementing training modeling.

1.3.6 Check the number of rows and columns in the dataset

[30]:
```
df_original.shape
```

[30]:
```
(129880, 22)
```

Hint 1

Use the `shape` attribute on the DataFrame.

1.3.7 Drop the rows with missing values

Drop the rows with missing values and save the resulting pandas DataFrame in a variable named `df_subset`.

[31]:
```
df_subset = df_original.dropna(axis=0)

df_subset.isna().any(axis=0).sum()
```

[31]: 0

Hint 1

Use the `dropna` function.

Hint 2

Set the axis parameter passed into the `dropna` function to `0` if you want to drop rows containing missing values, or `1` if you want to drop columns containing missing values. Optionally, use reset_index to avoid a SettingWithCopy warning later in the notebook.

1.3.8 Check for missing values

Check that `df_subset` does not contain any missing values.

[32]:
```
df_subset.isna().sum()
```

[32]:
```
satisfaction                         0
Customer Type                        0
Age                                  0
Type of Travel                       0
Class                                0
Flight Distance                      0
Seat comfort                         0
Departure/Arrival time convenient    0
Food and drink                       0
Gate location                        0
Inflight wifi service                0
Inflight entertainment               0
Online support                       0
Ease of Online booking               0
On-board service                     0
Leg room service                     0
Baggage handling                     0
Checkin service                      0
Cleanliness                          0
Online boarding                      0
Departure Delay in Minutes           0
Arrival Delay in Minutes             0
dtype: int64
```

Hint 1

Use the `isna()`function and the `sum()` function.

Hint 2

To get the number of rows in the data with missing values, use the `isna()` function followed by the `sum()` function.

1.3.9 Check the number of rows and columns in the dataset again

Check how many rows and columns are remaining in the dataset. You should now have 393 fewer rows of data.

[33]:
```
df_subset.shape
```

[33]:
```
(129487, 22)
```

1.3.10 Encode the data

Four columns (`satisfaction`, `Customer Type`, `Type of Travel`, `Class`) are the pandas dtype object. Decision trees need numeric columns. Start by converting the ordinal `Class` column into numeric.

[34]:
```
df_subset['Class'] = df_subset['Class'].map({"Business": 3, "Eco Plus": 2,␣
↪"Eco": 1})
```

Hint 1

Use the `map()` or `replace()` function.

Hint 2

For both functions, you will need to pass in a dictionary of class mappings `{"Business": 3, "Eco Plus": 2, "Eco": 1})`.

1.3.11 Represent the data in the target variable numerically

To represent the data in the target variable numerically, assign `"satisfied"` to the label `1` and `"dissatisfied"` to the label `0` in the `satisfaction` column.

[35]:
```
df_subset['satisfaction'] = df_subset['satisfaction'].map({"satisfied": 1,␣
↪"dissatisfied": 0})
```

Hint 1

Use the `map()` function to assign existing values in a column to new values.

Hint 2

Call `map()` on the `satisfaction` column and pass in a dictionary specifying that `"satisfied"` should be assigned to `1` and `"dissatisfied"` should be assigned to `0`.

Hint 3

Update the `satisfaction` column in `df_subset` with the newly assigned values.

1.3.12 Convert categorical columns into numeric

There are other columns in the dataset that are still categorical. Be sure to convert categorical columns in the dataset into numeric.

[36]:
```
df_subset = pd.get_dummies(df_subset, drop_first = True)
```

Hint 1

Use the `get_dummies()` function.

Hint 2

Set the `drop_first` parameter to `True`. This removes redundant data.

1.3.13 Check column data types

Now that you have converted categorical columns into numeric, check your column data types.

[37]:
```
df_subset.dtypes
```

[37]:
```
satisfaction                           int64
Age                                    int64
Class                                  int64
Flight Distance                        int64
Seat comfort                           int64
Departure/Arrival time convenient      int64
Food and drink                         int64
Gate location                          int64
Inflight wifi service                  int64
Inflight entertainment                 int64
Online support                         int64
Ease of Online booking                 int64
On-board service                       int64
Leg room service                       int64
Baggage handling                       int64
Checkin service                        int64
Cleanliness                            int64
Online boarding                        int64
Departure Delay in Minutes             int64
Arrival Delay in Minutes             float64
Customer Type_disloyal Customer        uint8
Type of Travel_Personal Travel         uint8
dtype: object
```

Hint 1

Use the `dtypes` attribute on the DataFrame.

1.3.14 Create the training and testing data

Put 75% of the data into a training set and the remaining 25% into a testing set.

[38]:
```
y = df_subset["satisfaction"]

X = df_subset.copy()
X = X.drop("satisfaction", axis = 1)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
↪random_state=0)
```

Hint 1

Use `train_test_split`.

Hint 2

Pass in `0` to `random_state`.

Hint 3

If you named your features matrix X and your target y, then it would be `train_test_split(X, y, test_size=0.25, random_state=0)`.

1.4 Step 3: Model building

1.4.1 Fit a decision tree classifier model to the data

Make a decision tree instance called `decision_tree` and pass in `0` to the `random_state` parameter. This is only so that if other data professionals run this code, they get the same results. Fit the model on the training set, use the `predict()` function on the testing set, and assign those predictions to the variable `dt_pred`.

[39]:
```
decision_tree = DecisionTreeClassifier(random_state=0)
decision_tree.fit(X_train, y_train)
dt_pred = decision_tree.predict(X_test)
```

Hint 1

Use `DecisionTreeClassifier`, the `fit()` function, and the `predict()` function.

Question: What are some advantages of using decision trees versus other models you have learned about?

With decision trees it's easier to understand even when the tree begins to become very complex. It's also easier to see the replication of work if and when someone may need to repeat the process.

1.5 Step 4: Results and evaluation

Print out the decision tree model's accuracy, precision, recall, and F1 score.

[40]:
```python
print("Decision Tree")
print("Accuracy:", "%.6f" % metrics.accuracy_score(y_test, dt_pred))
print("Precision:", "%.6f" % metrics.precision_score(y_test, dt_pred))
print("Recall:", "%.6f" % metrics.recall_score(y_test, dt_pred))
print("F1 Score:", "%.6f" % metrics.f1_score(y_test, dt_pred))
```

```
Decision Tree
Accuracy: 0.935438
Precision: 0.942859
Recall: 0.939030
F1 Score: 0.940940
```

Hint 1

Use four different functions from `metrics` to get the accuracy, precision, recall, and F1 score.

Hint 2

Input `y_test` and `y_pred` into the `metrics.accuracy_score`, `metrics.precision_score`, `metrics.recall_score` and `metrics.f1_score` functions.

Question: Are there any additional steps you could take to improve the performance or function of your decision tree?

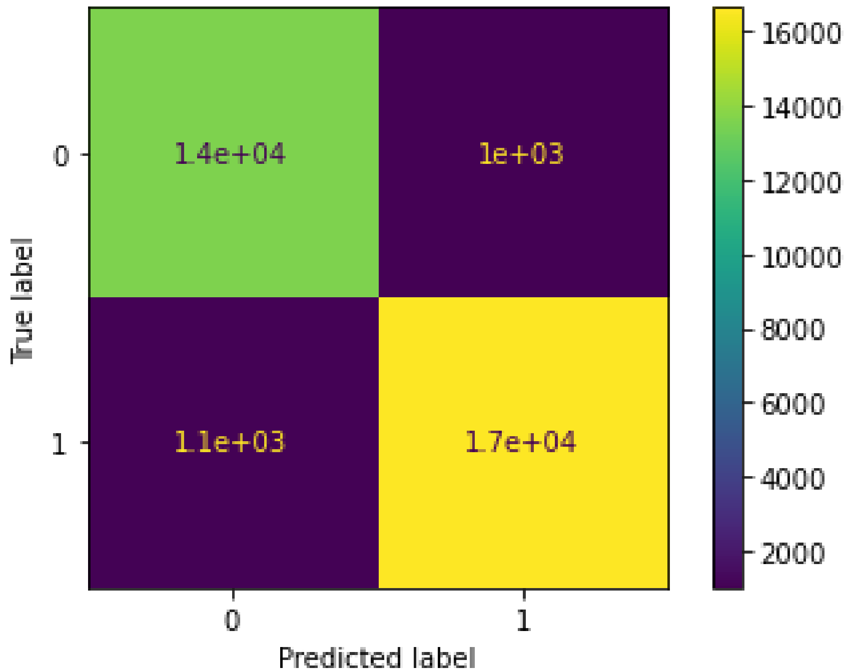
In this case I would say no as I do not want to overfit the model and it seems to not be underfit either. It's in the area of being just right.

1.5.1 Produce a confusion matrix

Data professionals often like to know the types of errors made by an algorithm. To obtain this information, produce a confusion matrix.

[41]:
```python
cm = metrics.confusion_matrix(y_test, dt_pred, labels= decision_tree.classes_)
disp = metrics.ConfusionMatrixDisplay(confusion_matrix = cm,display_labels =
 ↪decision_tree.classes_)
disp.plot()
```

[41]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f80662ac250>
```



Hint 1

Refer to the content about plotting a confusion matrix.

Hint 2

Use `metrics.confusion_matrix`, `metrics.ConfusionMatrixDisplay`, and the `plot()` function.

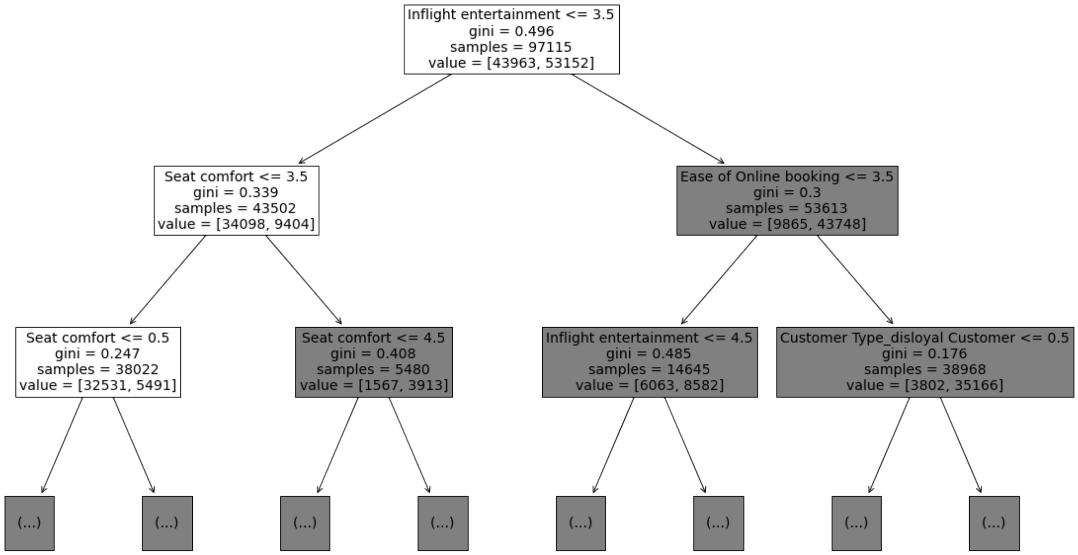
Question: What patterns can you identify between true positives and true negatives, as well as false positives and false negatives?

From the results of the matrix it is showing that it's accuracy is high on predicting customers satisfaction. It also appears that the inaccuracies is low with this model.

1.5.2 Plot the decision tree

Examine the decision tree. Use `plot_tree` function to produce a visual representation of the tree to pinpoint where the splits in the data are occurring.

[42]:
```
plt.figure(figsize=(20,12))
plot_tree(decision_tree, max_depth=2, fontsize=14, feature_names=X.columns);
```



Hint 1

If your tree is hard to read, pass `2` or `3` in the parameter `max_depth`.

1.5.3 Hyperparameter tuning

Knowing how and when to adjust or tune a model can help a data professional significantly increase performance. In this section, you will find the best values for the hyperparameters `max_depth` and `min_samples_leaf` using grid search and cross validation. Below are some values for the hyperparameters `max_depth` and `min_samples_leaf`.

[43]:
```python
tree_para = {'max_depth':
↪[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,30,40,50],
             'min_samples_leaf': [2,3,4,5,6,7,8,9, 10, 15, 20, 50]}

scoring = {'accuracy', 'precision', 'recall', 'f1'}
```

1.5.4 Check combinations of values

Check every combination of values to examine which pair has the best evaluation metrics. Make a decision tree instance called `tuned_decision_tree` with `random_state=0`, make a `GridSearchCV` instance called `clf`, make sure to refit the estimator using `"f1"`, and fit the model on the training set.

Note: This cell may take up to 15 minutes to run.

```
[44]: tuned_decision_tree = DecisionTreeClassifier(random_state=0)

clf = GridSearchCV(tuned_decision_tree,
                   tree_para,
                   scoring = scoring,
                   cv=5,
                   refit="f1")

clf.fit(X_train, y_train)
```

```
[44]: GridSearchCV(cv=5, error_score=nan,
             estimator=DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features=None,
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              presort='deprecated',
                                              random_state=0, splitter='best'),
             iid='deprecated', n_jobs=None,
             param_grid={'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12,
                                       13, 14, 15, 16, 17, 18, 19, 20, 30, 40,
                                       50],
                         'min_samples_leaf': [2, 3, 4, 5, 6, 7, 8, 9, 10, 15,
                                              20, 50]},
             pre_dispatch='2*n_jobs', refit='f1', return_train_score=False,
             scoring={'precision', 'f1', 'recall', 'accuracy'}, verbose=0)
```

Hint 1

Refer to the content about decision trees and grid search.

Hint 2

Use `DecisionTreeClassifier()`, `GridSearchCV()`, and the `clf.fit()` function.

Question: How can you determine the best combination of values for the hyperparameters?

With the best estimator package

1.5.5 Compute the best combination of values for the hyperparameters

```
[45]: clf.best_estimator_
```

```
[45]: DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=18, max_features=None, max_leaf_nodes=None,
```

```
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=2, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, presort='deprecated',
                                random_state=0, splitter='best')
```

Hint 1

Use the `best_estimator_` attribute.

Question: What is the best combination of values for the hyperparameters?

The results of the Decision Tree Classifier shows that max depth is 18 and samples is 2, so this would be the best combo.

Question: What was the best average validation score?

[46]:
```
print("Best average Validation Score: " , "%.4f" % clf.best_score_)
```

```
Best average Validation Score:  0.9454
```

The bes validation score is 0.9454

Hint 1

Use the `.best_score_` attribute.

1.5.6 Determine the "best" decision tree model's accuracy, precision, recall, and F1 score

Print out the decision tree model's accuracy, precision, recall, and F1 score. This task can be done in a number of ways.

[52]:
```
results = pd.DataFrame(columns=['Model', 'F1', 'Recall', 'Precision',␣
 ↪'Accuracy'])

def make_results(model_name, model_object):


    # Get all the results from the CV and put them in a df.
    cv_results = pd.DataFrame(model_object.cv_results_)

    # Isolate the row of the df with the max(mean f1 score).
    best_estimator_results = cv_results.iloc[cv_results['mean_test_f1'].
 ↪idxmax(), :]

    # Extract accuracy, precision, recall, and f1 score from that row.
    f1 = best_estimator_results.mean_test_f1
    recall = best_estimator_results.mean_test_recall
    precision = best_estimator_results.mean_test_precision
    accuracy = best_estimator_results.mean_test_accuracy
```

```
    # Create a table of results.
    table = pd.DataFrame()
    table = table.append({'Model': model_name,
                          'F1': f1,
                          'Recall': recall,
                          'Precision': precision,
                          'Accuracy': accuracy},
                         ignore_index=True)

    return table

result_table = make_results("Tuned Decision Tree", clf)

result_table
```

[52]:

	Model	F1	Recall	Precision	Accuracy
0	Tuned Decision Tree	0.945422	0.935863	0.955197	0.940864

Hint 1

Get all the results (`.cv_results_`) from the GridSearchCV instance (`clf`).

Hint 2

Output `mean_test_f1`, `mean_test_recall`, `mean_test_precision`, and `mean_test_accuracy` from `clf.cv_results_`.

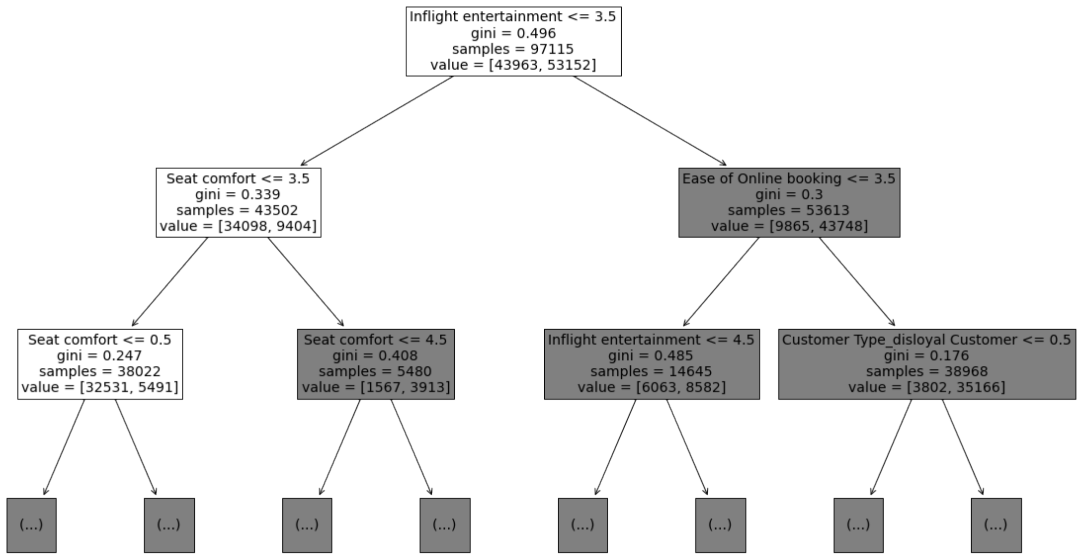
Question: Was the additional performance improvement from hyperparameter tuning worth the computational cost? Why or why not?

The tuning was very marginal with this model, in this case it was not worth it.

1.5.7 Plot the "best" decision tree

Use the `plot_tree` function to produce a representation of the tree to pinpoint where the splits in the data are occurring. This will allow you to review the "best" decision tree.

[54]:
```
plt.figure(figsize=(20,12))
plot_tree(clf.best_estimator_, max_depth=2, fontsize=14, feature_names=X.
 ↪columns);
```



Which features did the model use first to sort the samples?

Inflight entertainment, Seat comfort, and ease of online booking

1.6 Conclusion

What are some key takeaways that you learned from this lab? It is important to ensure the model is properly fit making sure it is not under nor over fit. Also there is such thing as over computing when things are already as accurate as possible it is easy to make changes that are not needed and can even hinder your work at times.

What findings would you share with others? This specific model I have made has an accuracy of 94%.

What would you recommend to stakeholders? Customers seem to hold inflight entertainment, seat comfort, and ease of online booking higher than other ammenities, focusing more on these 3 areas should increase customer satisfaction more.

Congratulations! You've completed this lab. However, you may not notice a green check mark next to this item on Coursera's platform. Please continue your progress regardless of the check mark. Just click on the "save" icon at the top of this notebook to ensure your work has been logged