Exemplar_Build a decision tree

January 7, 2024

1 Exemplar: Build a decision tree

1.1 Introduction

In this activity, you'll build a decision tree model that 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

[1]:
```
### YOUR CODE HERE ###

# 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.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
import sklearn.metrics as metrics

# 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.

[2]:
```
# 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

[3]:
```
### YOUR CODE HERE ###

df_original.head(n = 10)
```

[3]:

	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.

[4]:
```
### YOUR CODE HERE ###

df_original.dtypes
```

[4]:
```
satisfaction                          object
Customer Type                         object
Age                                    int64
Type of Travel                        object
Class                                 object
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
dtype: object
```

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.

[5]:
```
### YOUR CODE HERE ###

df_original["Class"].unique()
```

[5]:
```
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.

[6]:
```
### YOUR CODE HERE ###

df_original['satisfaction'].value_counts(dropna = False)
```

[6]:
```
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?

54.7 percent (71087/129880) of customers were satisfied. This value can be compared to a decision tree's model accuracy.

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.

[7]:
```
### YOUR CODE HERE ###

df_original.isnull().sum()
```

[7]:
```
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?

This is important to check because if there are only a small number of missing values in the dataset, they can more safely be removed.

1.3.6 Check the number of rows and columns in the dataset

[8]:
```
### YOUR CODE HERE ###

df_original.shape
```

[8]:
```
(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`.

[9]:
```
### YOUR CODE HERE ###

df_subset = df_original.dropna(axis=0).reset_index(drop = True)
```

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.

[10]:
```
### YOUR CODE HERE ###

df_subset.isna().sum()
```

[10]:
```
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.

[11]:
```
### YOUR CODE HERE ###

df_subset.shape
```

[11]:
```
(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.

[12]:
```
### YOUR CODE HERE ###

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.

[13]:
```
### YOUR CODE HERE ###
```

```
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.

[14]:
```
### YOUR CODE HERE ###

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.

[15]:
```
### YOUR CODE HERE ###

df_subset.dtypes
```

[15]:
```
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.

[16]:
```
### YOUR CODE HERE ###

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`.

[17]:
```
### YOUR CODE HERE ###

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?

Decision trees require no assumptions regarding the distribution of underlying data and don't require scaling of features. This lab uses decision trees because there is no need for additional data processing, unlike some other models.

1.5 Step 4: Results and evaluation

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

[18]:
```
### YOUR CODE HERE ###

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?

Decision trees can be particularly susceptible to overfitting. Combining hyperparameter tuning and grid search can help ensure this doesn't happen. For instance, setting an appropriate value for max depth could potentially help reduce a decision tree's overfitting problem by limiting how deep a tree can grow.

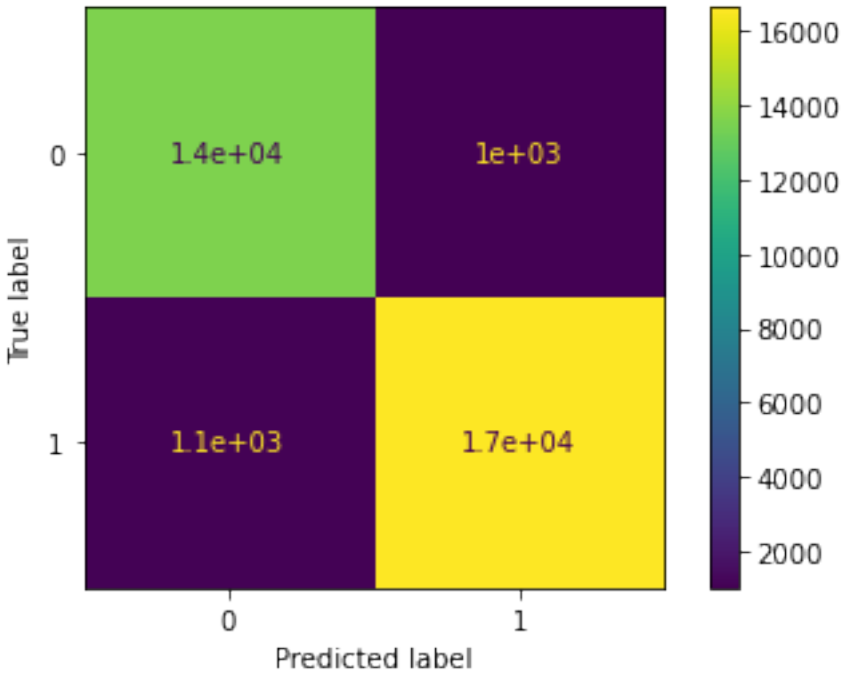
1.5.1 Produce a confusion matrix

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

[19]:
```
### YOUR CODE HERE ###

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()
```

[19]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f98bdca6090>
```



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?

In the confusion matrix, there are a high proportion of true positives and true negatives (where the matrix accurately predicted that the customer would be satisfied or dissatified, respectively).

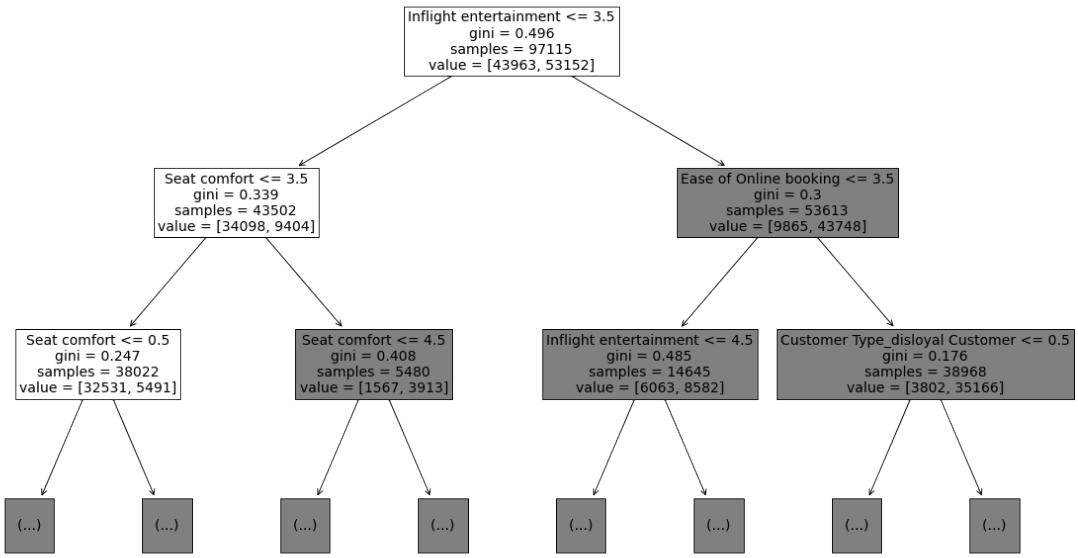
The matrix also had a relatively low number of false positives and false negatives (where the matrix innacurately predicted that the customer would be satisfied or dissatified, respectively.)

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.

[20]:
```
### YOUR CODE HERE ###

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 BONUS STEP: Calculate feature importance

The `DecisionTreeClassifier` class comes with a `feature_importances_` attribute that provides access to the features' relative importance in the model. The feature importances are computed when the `fit()` method is called on the `DecisionTreeClassifier` instance. In other words, it's information that is generated during model training. Here's how it works.

For each feature used in the tree, the algorithm finds all of the decision nodes that use that particular feature as the split criterion. Then for each of those decision nodes it computes the decrease in Gini impurity (or entropy, or log loss, or whatever metric you select when you fit the model—default is Gini impurity) that results from that split (so, the decrease from parent to children). Then the algorithm sums up the decreases across all the decisions made using that feature and expresses it as a percentage of the total decrease that resulted from *all* features.

Here is a simple example of how you can calculate feature importance from a tree diagram. This tree splits 10 samples using two features, X1 and X2:

Notice that the reduction in Gini impurity is weighted based on how many samples are in each node. Ultimately, the reduction that results from each feature is normalized (i.e., expressed as a percentage), and this percentage represents the importance of a given feature.

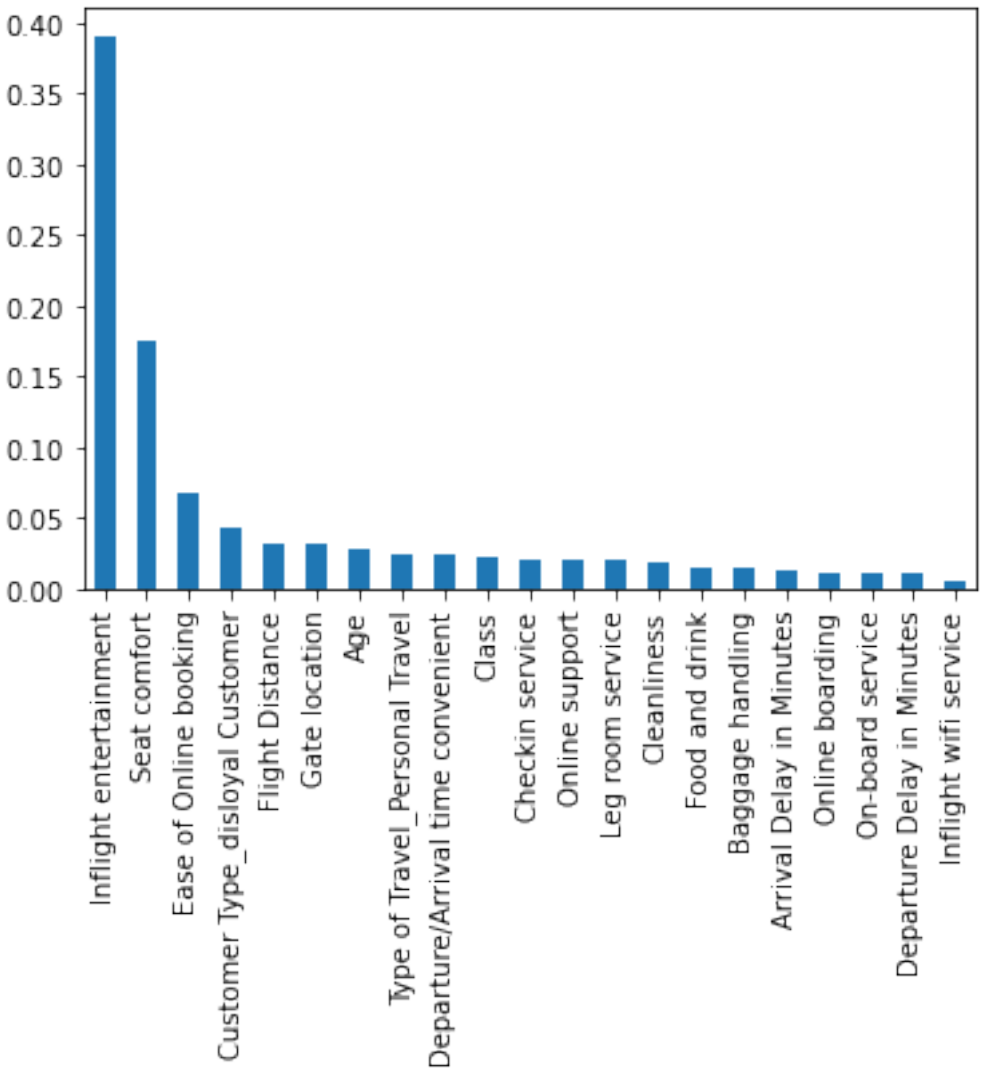
This can be interpreted as: X1 is responsible for 64.3% of the overall reduction of Gini impurity in the model, and X2 is responsible for 35.7% of the overall reduction of Gini impurity in the model.

Using the `feature_importances_` attribute to fetch the relative importances of each feature, you can then plot the results.

[21]:
```
importances = decision_tree.feature_importances_

forest_importances = pd.Series(importances, index=X.columns).
 ↪sort_values(ascending=False)

fig, ax = plt.subplots()
forest_importances.plot.bar(ax=ax);
```



The feature importance graph seems to confirm that `'Inflight entertainment'`, `'Seat comfort'`, and `'Ease of Online booking'` are the most important features for this model.

1.5.4 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`.

[22]:
```
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'}
```

1.5.5 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.

[23]:
```
### YOUR CODE HERE ###

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)
```

[23]:
```
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={'f1', 'recall', 'precision', '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?

Use the best estimator tool to help uncover the best pair combination.

1.5.6 Compute the best combination of values for the hyperparameters

[24]:
```
### YOUR CODE HERE ###

clf.best_estimator_
```

[24]:
```
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?

After running the DecisionTreeClassifier, the maximum depth is 18 and the minimum number of samples is two, meaning this is the best combination of values.

1.5.7 Determine the best average validation score

[25]:
```
### YOUR CODE HERE ###

print("Best Avg. Validation Score: ", "%.4f" % clf.best_score_)
```

```
Best Avg. Validation Score:  0.9454
```

Hint 1

Use the `.best_score_` attribute.

Question: What is the best average validation score?

The best validation score is 0.9454.

1.5.8 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.

[26]:
```
### YOUR CODE HERE ###

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

def make_results(model_name, model_object):
    """
    Accepts as arguments a model name (your choice - string) and
    a fit GridSearchCV model object.

    Returns a pandas df with the F1, recall, precision, and accuracy scores
    for the model with the best mean F1 score across all validation folds.
    """

    # 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 table of results
    table = pd.DataFrame({'Model': [model_name],
                          'F1': [f1],
                          'Recall': [recall],
                          'Precision': [precision],
                          'Accuracy': [accuracy]
                         }
                        )

    return table

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

```
result_table
```

[26]:

	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?

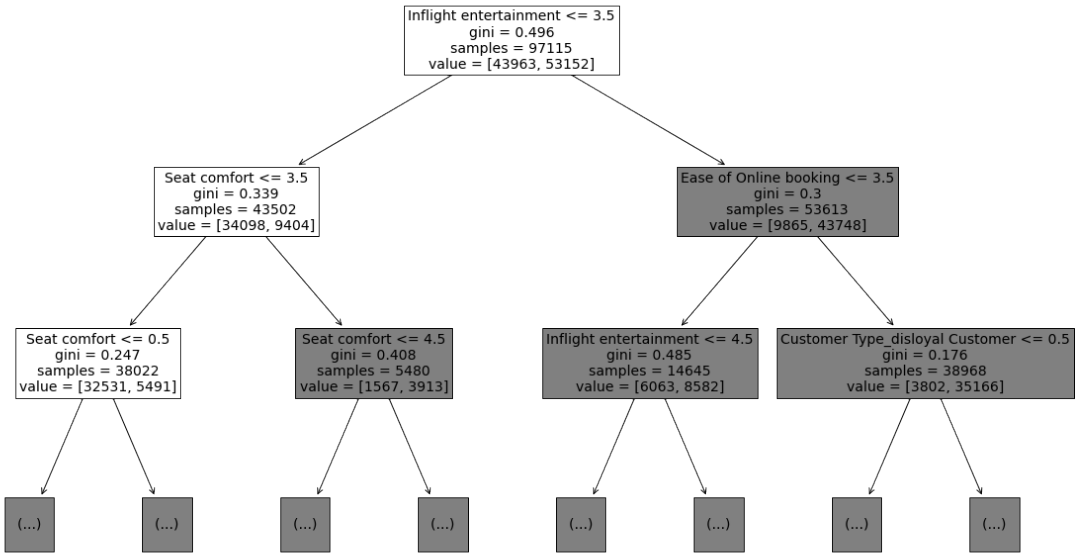
The F1 score for the decision tree that was not hyperparameter tuned is 0.940940 and the F1 score for the hyperparameter-tuned decision tree is 0.945422. While ensuring that overfitting doesn't occur is necessary for some models, it didn't make a meaningful difference in improving this model.

1.5.9 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.

[27]:
```
### YOUR CODE HERE ###

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



The plot makes it seem like `'Inflight entertainment'`, `'Seat comfort'`, and `'Ease of Online booking'` are among the most important features. The code below outputs a "most important" features graph from the model.

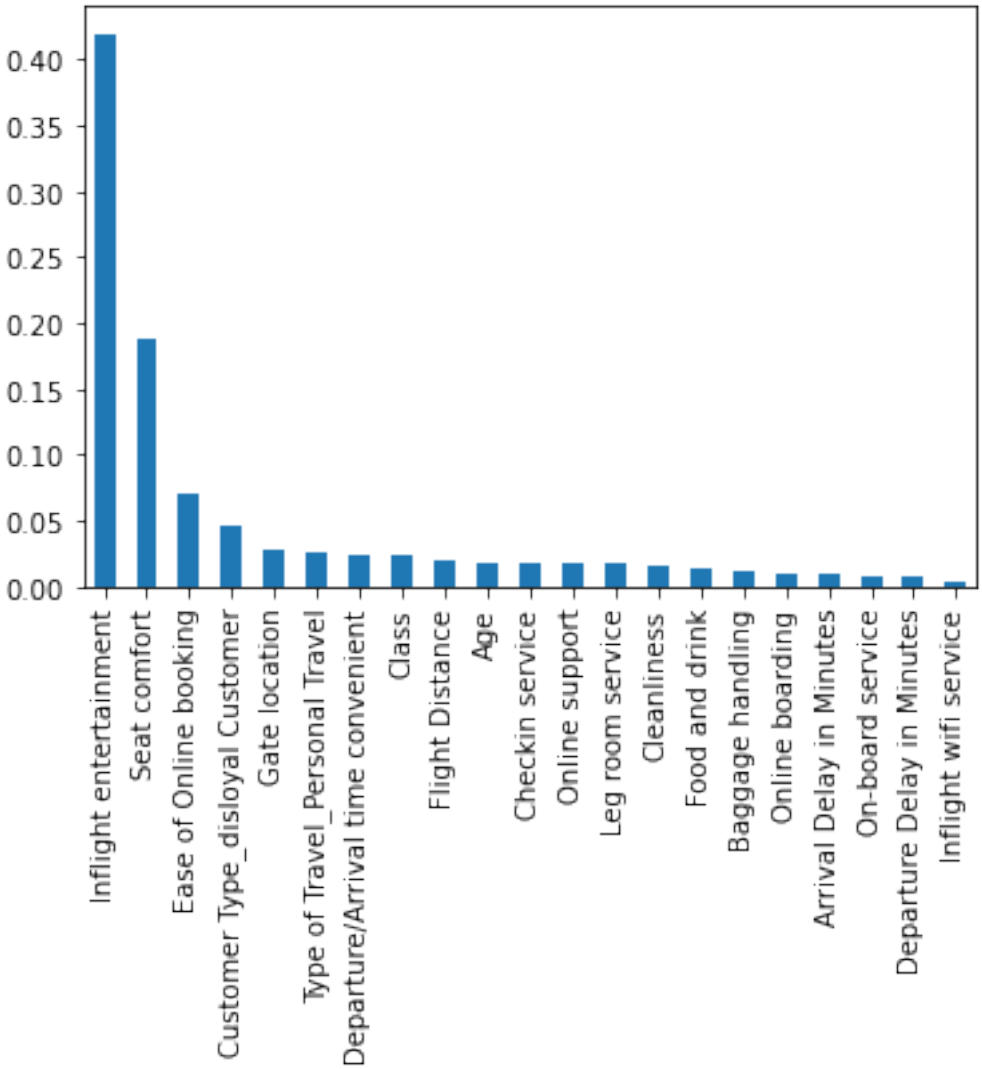
1.5.10 BONUS STEP: Build another feature importance graph

You can build another feature importance graph to validate which features are most important.

[29]:
```
importances = clf.best_estimator_.feature_importances_

forest_importances = pd.Series(importances, index=X.columns).
 ↪sort_values(ascending=False)

fig, ax = plt.subplots()
forest_importances.plot.bar(ax=ax);
```



The feature importance graph seems to confirm that `'Inflight entertainment'`, `'Seat comfort'`, and `'Ease of Online booking'` are the "most important" features for this model.

Question: What do you think is the most important metric in this business case?

Any of the metrics in this business case could be considered important, depending on what the stakeholder or domain expert cares about. The following are reasons why each metric is important:

- Accuracy tends to be the metric that the stakeholders can best understand.
- Precision measures what proportion of predicted positives is truly positive. For example, if you wanted to not falsely claiming a customer is satisfied, precision would be a good metric. Assuming a customer is happy when they are really not might lead to customer churn.
- Recall measures the percentage of actual positives a model correctly identified (true positive). For this dataset, the airline might want to limit false negatives (actually satisfied people who are predicted to be unsatisfied). Assuming a customer is unhappy when the customer is

happy can lead to the airline wasting resources trying to improve the customer experience of an already happy customer.

- F1 balances precision and recall. It is the harmonic mean of precision and recall, or their product divided by their sum.

1.6 Considerations

What are some key takeaways that you learned from this lab? * Machine learning workflows may be used to clean and encode data for machine learning. * While hyperparameter tuning can lead to an increase in performance, it doesn't always. * The visualization of the decision tree as well as the feature graph can be used to determine which features are most important for a decision tree.

What findings would you share with others? * Decision trees accurately predicted satisfaction over 94 percent of the time.
* The confusion matrix is useful as it shows a similar number of true positives and true negatives. * The visualization of the decision tree and the feature importance graph both suggest that `'Inflight entertainment'`, `'Seat comfort'`, and `'Ease of Online booking'` are the most important features in the model.

What would you recommend to stakeholders? * Customer satisfaction is highly tied to `'Inflight entertainment'`, `'Seat comfort'`, and `'Ease of Online booking'`. Improving these experiences should lead to better customer satisfaction. * The success of the model suggests that the airline should invest more effort into model building and model understanding since this model semed to be very good at predicting customer satisfaction.

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