Airline Company Satisfaction Decision Tree

August 17, 2023

1 Airline Company Satisfaction Decision Tree

1.1 Step 1: Imports

1.1.1 Import packages

[4]:
```
# Standard operational package imports
import pandas as pd
import numpy as np

# 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
```

1.1.2 Load the dataset

[5]:
```
df_original = pd.read_csv("Invistico_Airline.csv")
```

1.1.3 Output the first 10 rows of data

[6]:
```
df_original.head(10)
```

[6]:
```
  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]
```

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

1.2.1 Explore the data

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

[7]:
```
df_original.dtypes
```

[7]:
```
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
```

1.2.2 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.

[48]:
```python
df_original["satisfaction"].value_counts()
```

[48]:
```
satisfied       71087
dissatisfied    58793
Name: satisfaction, dtype: int64
```

1.2.3 Check for missing values

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

[10]:
```python
df_original.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              393
dtype: int64
```

1.2.4 Check the number of rows and columns in the dataset

[13]:
```
df_original.shape
```

[13]:
```
(129880, 22)
```

1.2.5 Drop the rows with missing values

[14]:
```
df_subset=df_original.dropna(axis=0).reset_index(drop=True)
```

1.2.6 Check for missing values

[15]:
```
df_subset.isna().sum()
```

[15]:
```
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
```

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

[49]:
```
df_subset.shape
```

[49]:
```
(129487, 22)
```

1.2.8 Encode the data

Four columns (`satisfaction`, `Customer Type`, `Type of Travel`, `Class`) are the pandas dtype object. Decision trees need numeric columns.

[19]:
```
df_subset["Class"]=df_subset["Class"].map({"Business":3, "Eco Plus":2, "Eco":1})

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

1.2.9 Convert other categorical columns into numeric

[21]:
```
df_subset=pd.get_dummies(df_subset, drop_first=True)
```

1.2.10 Check column data types

[23]:
```
df_subset.dtypes
```

[23]:
```
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
```

1.2.11 Create the training and testing data

[28]:
```
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)
```

1.3 Step 3: Model building

1.3.1 Fit a decision tree classifier model to the data

[26]:
```
decision_tree= DecisionTreeClassifier(random_state=0)
decision_tree.fit(X_train, y_train)
y_pred_dt=decision_tree.predict(X_test)
```

1.4 Step 4: Results and evaluation

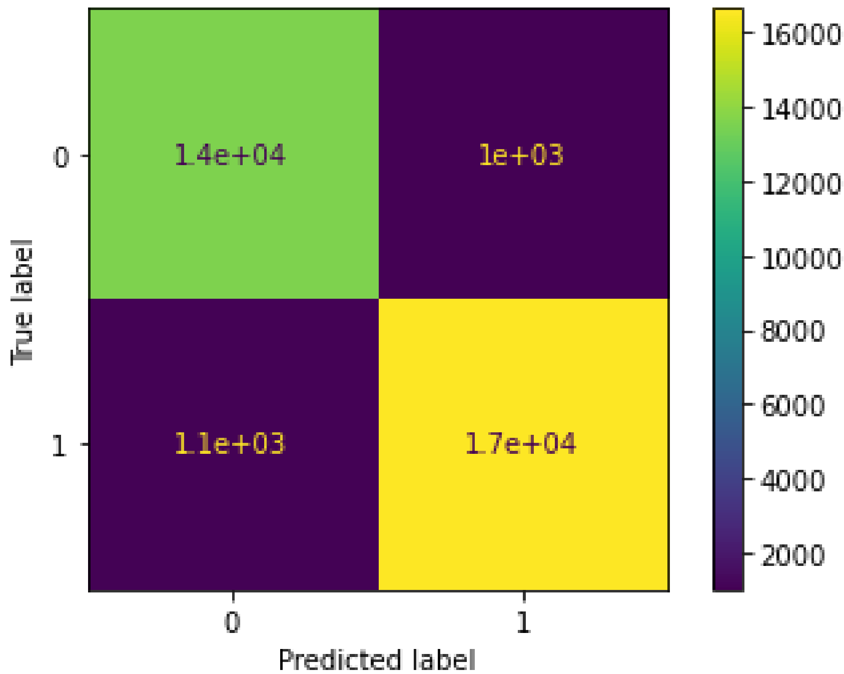
[32]:
```
print("Accuracy score:" + str(round(metrics.accuracy_score(y_test,␣
 ↪y_pred_dt),3)))
print("Precision score:" + str(round(metrics.precision_score(y_test,␣
 ↪y_pred_dt),3)))
print("Recall score:" + str(round(metrics.recall_score(y_test, y_pred_dt),3)))
print("F1 score:" + str(round(metrics.f1_score(y_test, y_pred_dt),3)))
```

```
Accuracy score:0.935
Precision score:0.943
Recall score:0.939
F1 score:0.941
```

1.4.1 Produce a confusion matrix

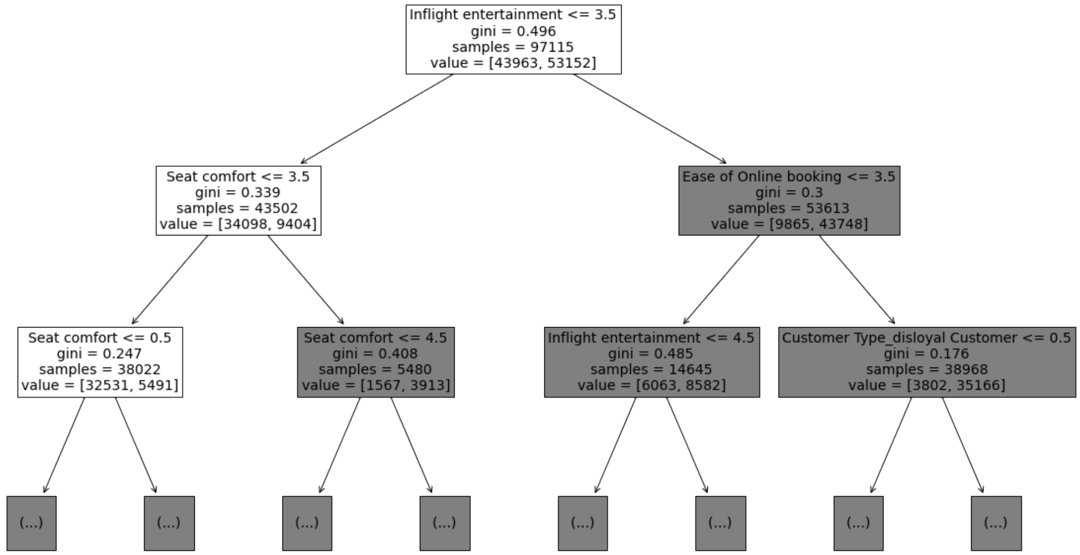
[33]:
```
cm=metrics.confusion_matrix(y_test, y_pred_dt, labels=decision_tree.classes_)
disp=metrics.ConfusionMatrixDisplay(confusion_matrix=cm,␣
 ↪display_labels=decision_tree.classes_)
disp.plot()
```

[33]: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7f51fe51fcd0>



1.4.2 Plot the decision tree

[34]:
```
plt.figure(figsize=(20,12))
plot_tree(decision_tree, max_depth=2, fontsize=14, feature_names=X.columns)
plt.show()
```



1.4.3 Hyperparameter tuning

Knowing how and when to adjust or tune a model can help a data professional significantly increase performance. In this section, we 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`.

[36]:
```
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'}
```

1.4.4 Check combinations of values

[38]:
```
tuned_decision_tree=DecisionTreeClassifier()

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

clf.fit(X_train, y_train)
```

[38]:
```
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=None,
                                              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={'accuracy', 'precision', 'f1', 'recall'}, verbose=0)
```

1.4.5 Compute the best combination of values for the hyperparameters

[39]:
```
clf.best_estimator_
```

[39]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=17, 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=None, splitter='best')
```

[41]:
```
print("Best average validation score:")

print(round(clf.best_score_, 2))
```

```
Best average validation score:
0.95
```

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

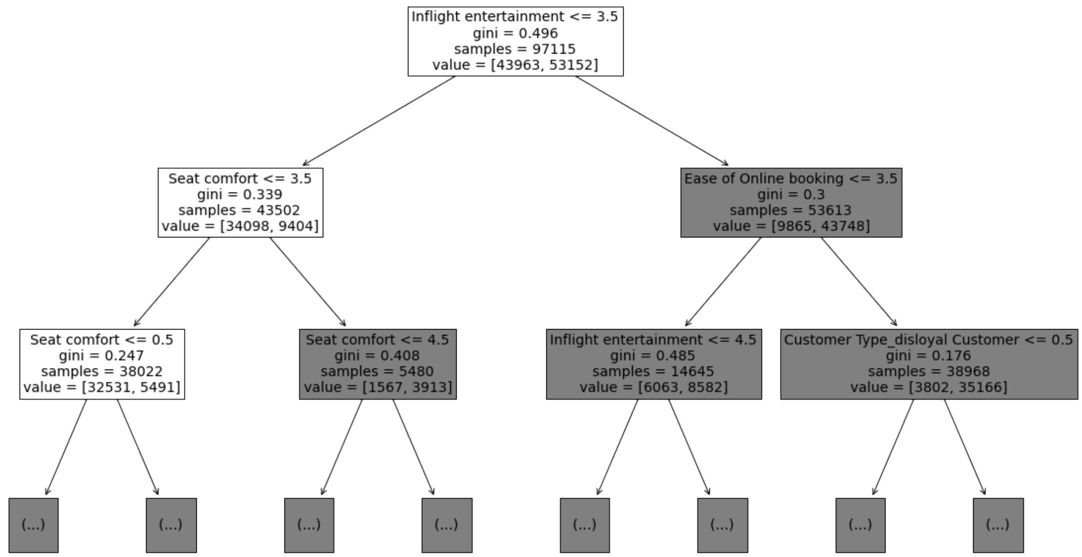
[45]:
```
results = pd.DataFrame(columns=["Model", "F1", "Recall", "Precision",␣
 ↪"Accuracy"])

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 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
```

[45]:

	Model	F1	Recall	Precision	Accuracy
0	Tuned Decision Tree	0.945168	0.936315	0.954215	0.940545

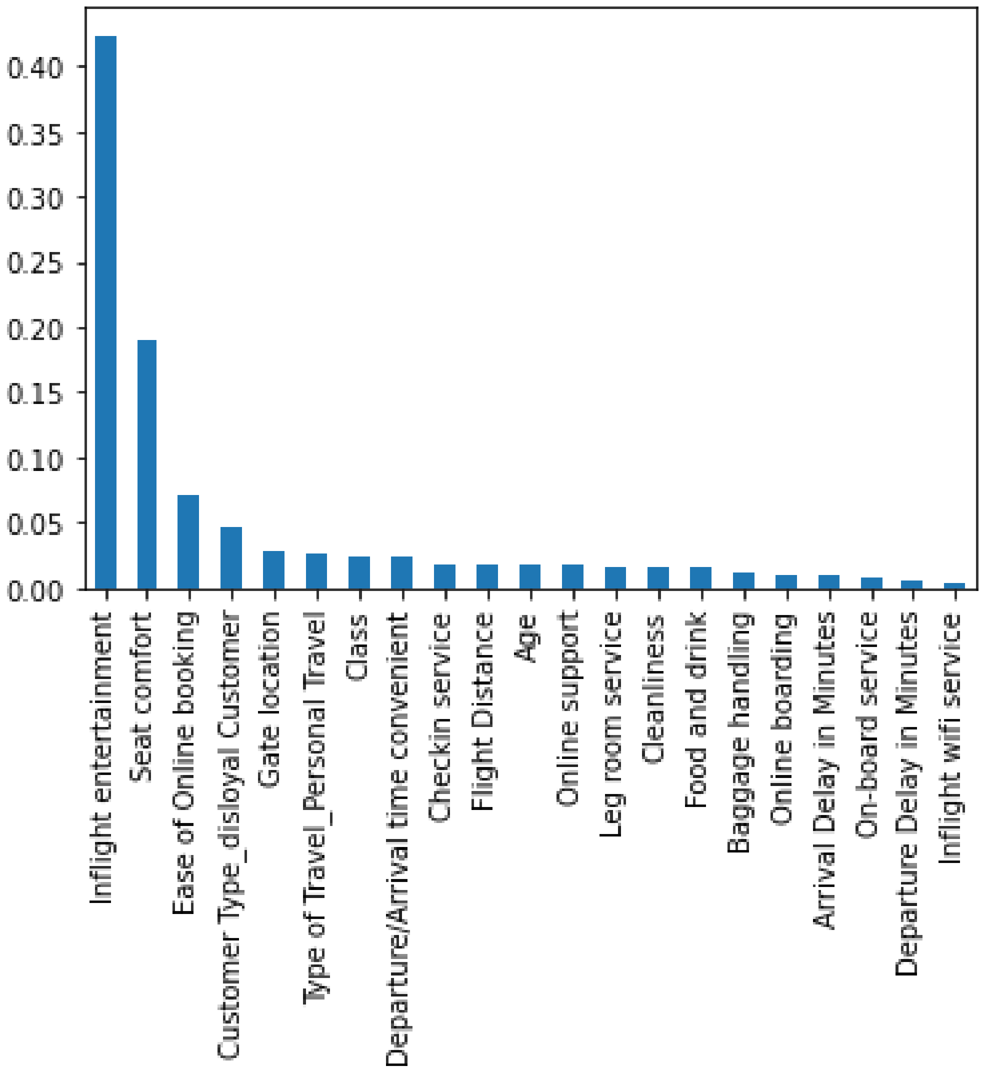
1.4.7 Plot the "best" decision tree

[46]:
```
plt.figure(figsize=(20,12))
plot_tree(clf.best_estimator_, max_depth=2, fontsize=14, feature_names=X.
 ↪columns)
plt.show()
```



1.4.8 Build feature importance graph

[47]:
```
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);
```



1.5 Conclusion

- 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.