Airline Company Satisfaction Random Forest

August 24, 2023

1 Airline Company Satisfaction Random Forest

1.1 Step 1: Imports

[34]:
```
import pandas as pd
import numpy as np
import pickle
import sklearn
import matplotlib.pyplot as plt
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, PredefinedSplit,␣
↪GridSearchCV
from sklearn.metrics import f1_score, precision_score, recall_score,␣
↪accuracy_score
import sklearn.metrics as metrics
```

[2]:
```
air_data = pd.read_csv("Invistico_Airline.csv")
```

1.2 Step 2: Data cleaning

To get a sense of the data, display the first 10 rows.

[3]:
```
# Display first 10 rows.
air_data.head(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]
```

Now, display the variable names and their data types.

[4]:
```
# Display variable names and types.
air_data.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
```

We observe that satisfaction, Customer Type, Type of Travel and Class columns are categorical while other variables are numeric.

Next, to understand the size of the dataset, identify the number of rows and the number of columns.

[5]:
```
# Identify the number of rows and the number of columns.
air_data.shape
```

[5]:
```
(129880, 22)
```

Now, check for missing values in the rows of the data.

[6]:
```
# Get the number of rows that contain missing values.
air_data.isna().sum()
```

[6]:
```
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
```

There are 393 missing values in the column "Arrival Delay in Minutes".

Drop the rows with missing values. Then, save the resulting pandas DataFrame in a variable named `air_data_subset`.

[7]:
```
# Drop missing values.
# Save the DataFrame in variable `air_data_subset`.
air_data_subset=air_data.dropna(axis=0)
```

```
air_data_subset.isna().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              0
dtype: int64
```

Next, display the first 10 rows to examine the data subset.

[8]:
```
# Display the first 10 rows.
air_data_subset.head(10)
```

[8]:

	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]
```

Confirm that it does not contain any missing values.

[9]:
```
# Count of missing values.
air_data_subset.isna().sum()
```

[9]:
```
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
```

Next, convert the categorical features to indicator (one-hot encoded) features. In order to use random forest model in machine learning, it is important to transform all of the variables (both predictor and target values) into numeric.

The target variable, `satisfaction`, does not need to be encoded and will be extracted in a later step.

[10]:
```
# Convert categorical features to one-hot encoded features.
air_data_subset_dummies=pd.get_dummies(air_data_subset, columns=["Customer␣
 ↪Type", "Type of Travel", "Class"])
air_data_subset_dummies
```

[10]:

```
        satisfaction  Age  Flight Distance  Seat comfort  \
0          satisfied   65              265             0
1          satisfied   47             2464             0
2          satisfied   15             2138             0
3          satisfied   60              623             0
4          satisfied   70              354             0
…                ...  ...              ...           ...
129875     satisfied   29             1731             5
129876  dissatisfied   63             2087             2
129877  dissatisfied   69             2320             3
129878  dissatisfied   66             2450             3
129879  dissatisfied   38             4307             3

        Departure/Arrival time convenient  Food and drink  Gate location  \
0                                       0               0              2
1                                       0               0              3
2                                       0               0              3
3                                       0               0              3
4                                       0               0              3
…                                     ...             ...            ...
129875                                  5               5              3
129876                                  3               2              4
129877                                  0               3              3
129878                                  2               3              2
129879                                  4               3              3

        Inflight wifi service  Inflight entertainment  Online support  …  \
0                           2                       4               2  …
1                           0                       2               2  …
2                           2                       0               2  …
3                           3                       4               3  …
4                           4                       3               4  …
…                         ...                     ...             ...  …
129875                      2                       5               2  …
129876                      2                       1               1  …
129877                      3                       2               2  …
129878                      3                       2               2  …
129879                      3                       3               3  …

        Online boarding  Departure Delay in Minutes  Arrival Delay in Minutes  \
0                     2                          0                       0.0
1                     2                        310                     305.0
```

```
2                    2                          0                      0.0
3                    3                          0                      0.0
4                    5                          0                      0.0
…                  …                          …                      …
129875               2                          0                      0.0
129876               1                        174                    172.0
129877               2                        155                    163.0
129878               2                        193                    205.0
129879               3                        185                    186.0

        Customer Type_Loyal Customer  Customer Type_disloyal Customer  \
0                                  1                                 0
1                                  1                                 0
2                                  1                                 0
3                                  1                                 0
4                                  1                                 0
…                                …                                 …
129875                             0                                 1
129876                             0                                 1
129877                             0                                 1
129878                             0                                 1
129879                             0                                 1

        Type of Travel_Business travel  Type of Travel_Personal Travel  \
0                                    0                                1
1                                    0                                1
2                                    0                                1
3                                    0                                1
4                                    0                                1
…                                  …                                …
129875                               0                                1
129876                               0                                1
129877                               0                                1
129878                               0                                1
129879                               0                                1

        Class_Business  Class_Eco  Class_Eco Plus
0                    0          1               0
1                    1          0               0
2                    0          1               0
3                    0          1               0
4                    0          1               0
…                  …          …               …
129875               0          1               0
129876               1          0               0
129877               0          1               0
129878               0          1               0
```

```
129879               0          1                0

[129487 rows x 26 columns]
```

Turning categorical variable into numeric varible is necessary because the sklearn implementation of RandomForestClassifier() requires that categorical features be encoded to numeric, which can be done using dummy variables or one-hot encoding.

Next, display the first 10 rows to review the `air_data_subset_dummies`.

[11]:
```
# Display the first 10 rows.
air_data_subset_dummies.head(10)
```

[11]:

	satisfaction	Age	Flight Distance	Seat comfort	\
0	satisfied	65	265	0	
1	satisfied	47	2464	0	
2	satisfied	15	2138	0	
3	satisfied	60	623	0	
4	satisfied	70	354	0	
5	satisfied	30	1894	0	
6	satisfied	66	227	0	
7	satisfied	10	1812	0	
8	satisfied	56	73	0	
9	satisfied	22	1556	0	

	Departure/Arrival time convenient	Food and drink	Gate location	\
0	0	0	2	
1	0	0	3	
2	0	0	3	
3	0	0	3	
4	0	0	3	
5	0	0	3	
6	0	0	3	
7	0	0	3	
8	0	0	3	
9	0	0	3	

	Inflight wifi service	Inflight entertainment	Online support	…	\
0	2	4	2	…	
1	0	2	2	…	
2	2	0	2	…	
3	3	4	3	…	
4	4	3	4	…	
5	2	0	2	…	
6	2	5	5	…	
7	2	0	2	…	
8	5	3	5	…	
9	2	0	2	…	

	Online boarding	Departure Delay in Minutes	Arrival Delay in Minutes	\
0	2	0	0.0	
1	2	310	305.0	
2	2	0	0.0	
3	3	0	0.0	
4	5	0	0.0	
5	2	0	0.0	
6	3	17	15.0	
7	2	0	0.0	
8	4	0	0.0	
9	2	30	26.0	

	Customer Type_Loyal Customer	Customer Type_disloyal Customer	\
0	1	0	
1	1	0	
2	1	0	
3	1	0	
4	1	0	
5	1	0	
6	1	0	
7	1	0	
8	1	0	
9	1	0	

	Type of Travel_Business travel	Type of Travel_Personal Travel	\
0	0	1	
1	0	1	
2	0	1	
3	0	1	
4	0	1	
5	0	1	
6	0	1	
7	0	1	
8	0	1	
9	0	1	

	Class_Business	Class_Eco	Class_Eco Plus
0	0	1	0
1	1	0	0
2	0	1	0
3	0	1	0
4	0	1	0
5	0	1	0
6	0	1	0
7	0	1	0
8	1	0	0

```
9                0          1                0

[10 rows x 26 columns]
```

Then, check the variables of air_data_subset_dummies.

[12]:
```
# Display variables.
air_data_subset_dummies.dtypes
```

[12]:
```
satisfaction                          object
Age                                    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_Loyal Customer           uint8
Customer Type_disloyal Customer        uint8
Type of Travel_Business travel         uint8
Type of Travel_Personal Travel         uint8
Class_Business                         uint8
Class_Eco                              uint8
Class_Eco Plus                         uint8
dtype: object
```

All of the variables with the relevant column name, are converted into uint8 data type.

1.3 Step 3: Model building

The first step to building the model is separating the labels (y) from the features (X).

[13]:
```
# Separate the dataset into labels (y) and features (X).
X= air_data_subset_dummies.drop("satisfaction", axis=1)
y=pd.DataFrame(air_data_subset["satisfaction"].replace({"dissatisfied":1,␣
 ↪"satisfied":0}))
```

Since target variable should be one column data frame, it would be better to use replace with a map of numeric values, instead of using get_dummies, since the latter turns the categoric variables into more than one columns.

Once separated, split the data into train, validate, and test sets.

[14]:
```
# Separate into train, validate, test sets.
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪stratify=y, random_state=42)
X_tr, X_val, y_tr, y_val = train_test_split(X_train, y_train, test_size=0.20,␣
 ↪stratify=y_train, random_state=42)
```

1.3.1 Tune the model

Now, we shall fit and tune a random forest model with separate validation set. We shall begin by determining a set of hyperparameters for tuning the model using GridSearchCV.

[15]:
```
# Determine set of hyperparameters.
cv_params = {'max_depth': [2,3,4,5, None],
             'min_samples_leaf': [1,2,3],
             'max_features': [2,3,4],
             'n_estimators': [75, 100, 125, 150]
             }
```

Next, create a list of split indices.

[16]:
```
# Create list of split indices.
split_index = [0 if x in X_val.index else -1 for x in X_train.index]
custom_split=PredefinedSplit(split_index)
```

Now, instantiate the model.

[17]:
```
# Instantiate model.
rf=RandomForestClassifier(random_state=0)
scoring = {'accuracy', 'precision', 'recall', 'f1'}
```

Next, use GridSearchCV to search over the specified parameters.

[18]:
```
# Search over specified parameters.
rf_val = GridSearchCV(rf, cv_params, scoring=scoring, cv=custom_split,␣
 ↪refit='f1', verbose=1)
```

Now, fit the model.

[19]:
```
# Fit the model.
rf_val.fit(X_train, y_train)
```

```
Fitting 1 folds for each of 180 candidates, totalling 180 fits
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done 180 out of 180 | elapsed: 11.0min finished
```

[19]:
```
GridSearchCV(cv=PredefinedSplit(test_fold=array([-1, -1, …, -1, -1])),
             error_score=nan,
             estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                              class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              max_samples=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weig…
                                              n_estimators=100, n_jobs=None,
                                              oob_score=False, random_state=0,
                                              verbose=0, warm_start=False),
             iid='deprecated', n_jobs=None,
             param_grid={'max_depth': [2, 3, 4, 5, None],
                         'max_features': [2, 3, 4],
                         'min_samples_leaf': [1, 2, 3],
                         'n_estimators': [75, 100, 125, 150]},
             pre_dispatch='2*n_jobs', refit='f1', return_train_score=False,
             scoring={'precision', 'accuracy', 'f1', 'recall'}, verbose=1)
```

Finally, obtaining the optimal parameters.

[20]:
```
# Obtain optimal parameters.
rf_val.best_params_
```

[20]:
```
{'max_depth': None,
 'max_features': 4,
 'min_samples_leaf': 1,
 'n_estimators': 150}
```

1.4 Step 4: Results and evaluation

Use the selected model to predict on the test data. Use the optimal parameters found via GridSearchCV.

[21]:
```
# Use optimal parameters on GridSearchCV.
rf_opt=RandomForestClassifier(max_depth= None, max_features= 4,␣
 ↪min_samples_leaf= 1, n_estimators= 125, random_state=0)
```

Once again, fit the optimal model.

[22]:
```
# Fit the optimal model.
rf_opt.fit(X_train, y_train)
```

[22]:
```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features=4,
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=125,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

And predict on the test set using the optimal model.

[23]:
```
# Predict on test set.
y_pred=rf_opt.predict(X_test)
```

1.4.1 Obtain performance scores

Calculate the scores: precision score, recall score, accuracy score, F1 score.

[24]:
```
# Precision score on test data set.
pc_test=precision_score(y_test, y_pred)
print("Precision score is " + str(round(pc_test,2)))
```

```
Precision score is 0.94
```

[25]:
```
# Recall score on test data set.
rc_test=recall_score(y_test, y_pred)
print("Recall score is " + str(round(rc_test,2)))
```

```
Recall score is 0.96
```

[26]:
```
# Accuracy score on test data set.
ac_test=accuracy_score(y_test, y_pred)
print("Accuracy score is " + str(round(ac_test, 2)))
```

```
Accuracy score is 0.95
```

[27]:
```
# F1 score on test data set.
f1_test=f1_score(y_test, y_pred)
print("F1 score is " + str(round(f1_test, 2)))
```

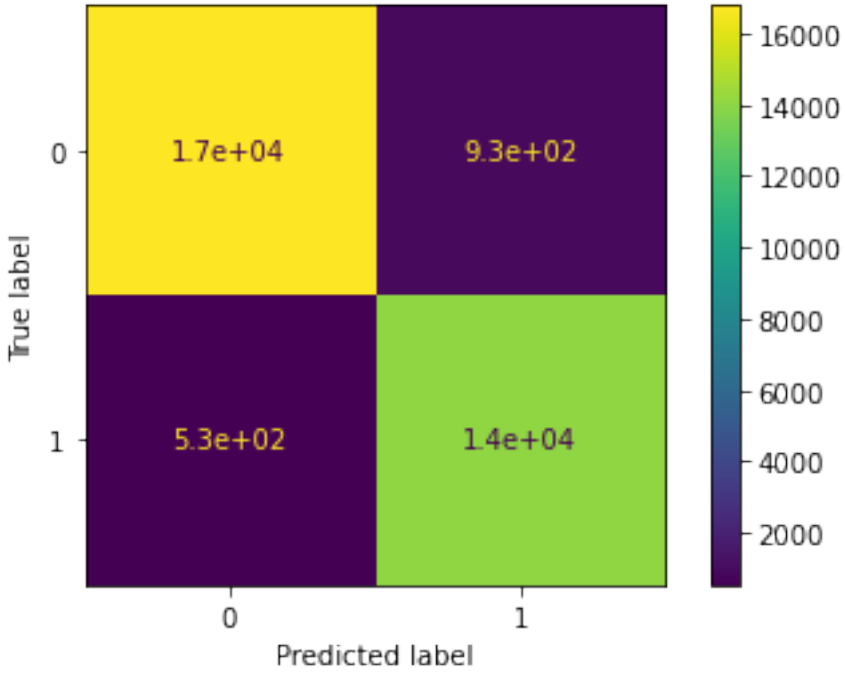
```
F1 score is 0.95
```

The model performs well according to all 4 performance metrics. The model's recall score is slightly better than the 3 other metrics.

1.4.2 Produce a confusion matrix

[31]:
```
cm=metrics.confusion_matrix(y_test, y_pred, labels=rf_opt.classes_)
disp=metrics.ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=rf_opt.
↪classes_)
disp.plot()
```

[31]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7fcdfd7eea50>
```



1.4.3 Evaluate the model

Finally, create a table of results to evaluate the performace of the model.

[32]:
```
# Create table of results.
table=pd.DataFrame()

table = table.append({'Model': "Tuned Random Forest",
                      'F1':  f1_test,
                      'Recall': rc_test,
                      'Precision': pc_test,
                      'Accuracy': ac_test
                     },
                      ignore_index=True
```

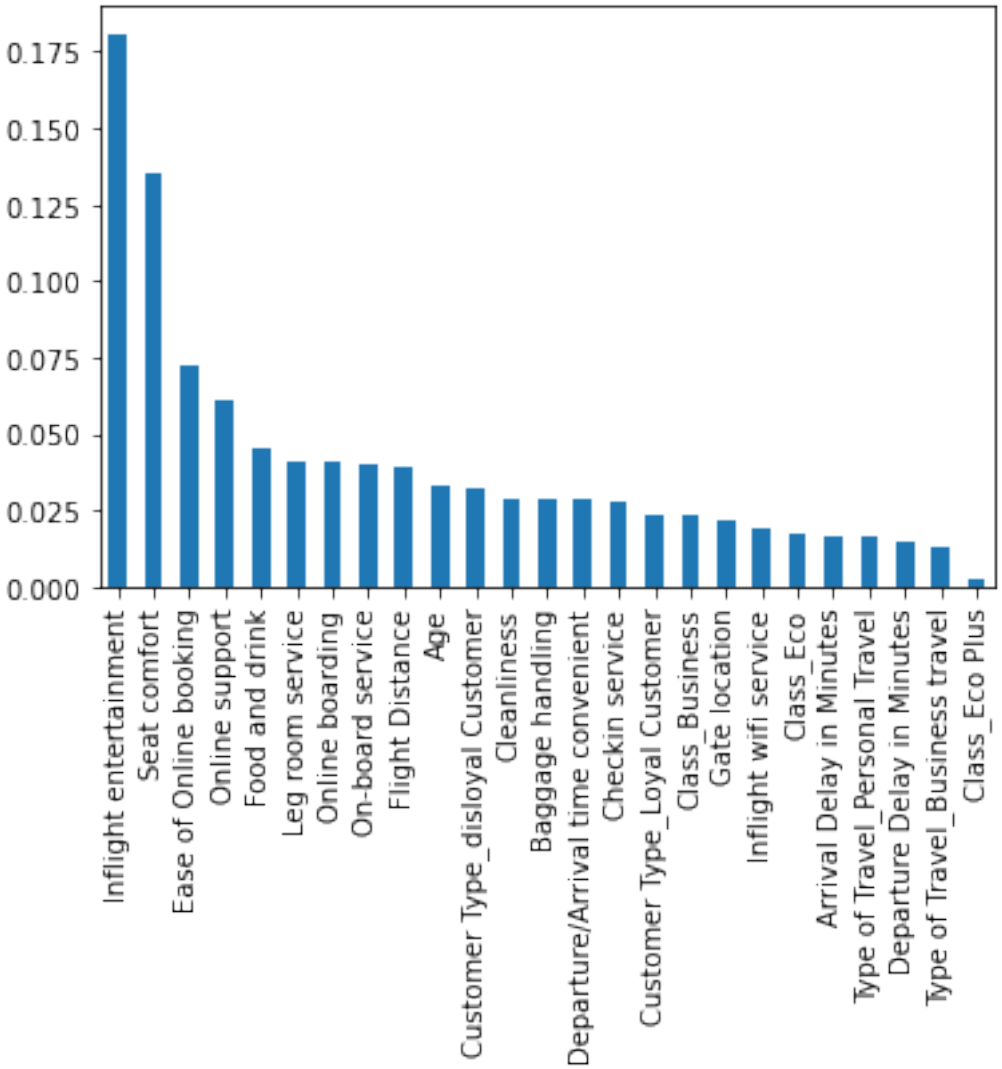
```
                    )
table
```

[32]:

	Model	F1	Recall	Precision	Accuracy
0	Tuned Random Forest	0.950906	0.963757	0.938393	0.954961

1.4.4 Feature Importance

[35]:
```
importances = rf_opt.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 Conclusions

- The random forest model predicted satisfaction with more than 95.4% accuracy. The precision is over 93.8% and the recall is approximately 96.4%.
- Customer satisfaction is highly tied to 'Inflight entertainment', 'Seat comfort', and 'Ease of Online booking'. Improving these experiences should lead to better customer satisfaction.