wz3wstcwq

November 4, 2024

Applied Machine Learning Assignment 2 Question 2

The assignment involves an exploration of various classification algorithms on a flight satisfaction dataset, focusing on binary classification. The dataset comprises 24 features, such as Gender, Customer Type, Age, Type of Travel, Class, Flight Distance, Online boarding, Seat comfort, and satisfaction levels, among others. Each feature provides essential context about passenger demographics, travel conditions, and experiences, allowing for an in-depth exploration of which classification methods yield the most accurate predictions for satisfaction levels. The objective is to compare classifier performance and determine the most effective model through metrics like accuracy, precision, recall, and F1 score.

Dataset Features

This dataset contains an airline passenger satisfaction survey. The following factors are examined to determine their correlation with passenger satisfaction:

- **Gender**: Gender of the passengers (Female, Male)
- **Customer Type**: The customer type (Loyal customer, Disloyal customer)
- **Age**: The actual age of the passengers
- **Type of Travel**: Purpose of the flight of the passengers (Personal Travel, Business Travel)
- **Class**: Travel class in the plane of the passengers (Business, Eco, Eco Plus)
- **Flight Distance**: The flight distance of this journey
- **Inflight Wifi Service**: Satisfaction level of the inflight wifi service (0: Not Applicable; 1-5)
- **Departure/Arrival Time Convenient**: Satisfaction level of Departure/Arrival time convenience
- **Ease of Online Booking**: Satisfaction level of online booking
- **Gate Location**: Satisfaction level of gate location
- **Food and Drink**: Satisfaction level of food and drink
- **Online Boarding**: Satisfaction level of online boarding
- **Seat Comfort**: Satisfaction level of seat comfort
- **Inflight Entertainment**: Satisfaction level of inflight entertainment
- **On-board Service**: Satisfaction level of on-board service
- **Leg Room Service**: Satisfaction level of leg room service
- **Baggage Handling**: Satisfaction level of baggage handling
- **Check-in Service**: Satisfaction level of check-in service
- **Inflight Service**: Satisfaction level of inflight service
- **Cleanliness**: Satisfaction level of cleanliness
- **Departure Delay in Minutes**: Minutes delayed during departure
- **Arrival Delay in Minutes**: Minutes delayed during arrival
- **Satisfaction**: Airline satisfaction level (Satisfaction or Dissatisfaction)

Q1. Display the statistical values for each of the attributes, along with visualizations (e.g., histogram) of the distributions for each attribute. Are there any attributes that might require special treatment? If so, what special treatment might they require?

[142]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')

df = pd.read_csv("FlightSatisfaction.csv").iloc[:480,1:]
```

[143]:
```
df.head()
```

[143]:

	id	Gender	Customer Type	Age	Type of Travel	Class	\
0	19556	Female	Loyal Customer	52	Business travel	Eco	
1	90035	Female	Loyal Customer	36	Business travel	Business	
2	12360	Male	disloyal Customer	20	Business travel	Eco	
3	77959	Male	Loyal Customer	44	Business travel	Business	
4	36875	Female	Loyal Customer	49	Business travel	Eco	

	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	\
0	160	5	4	
1	2863	1	1	
2	192	2	0	
3	3377	0	0	
4	1182	2	3	

	Ease of Online booking	…	Inflight entertainment	On-board service	\
0	3	…	5	5	
1	3	…	4	4	
2	2	…	2	4	
3	0	…	1	1	
4	4	…	2	2	

	Leg room service	Baggage handling	Checkin service	Inflight service	\
0	5	5	2	5	
1	4	4	3	4	
2	1	3	2	2	
3	1	1	3	1	
4	2	2	4	2	

	Cleanliness	Departure Delay in Minutes	Arrival Delay in Minutes	\
0	5	50	44.0	
1	5	0	0.0	

```
2           2                          0                       0.0
3           4                          0                       6.0
4           4                          0                      20.0

              satisfaction
0                satisfied
1                satisfied
2  neutral or dissatisfied
3                satisfied
4                satisfied

[5 rows x 24 columns]
```

[144]:
```
df.shape
```

[144]:
```
(480, 24)
```

[145]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 24 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   id                                 480 non-null    int64
 1   Gender                             480 non-null    object
 2   Customer Type                      480 non-null    object
 3   Age                                480 non-null    int64
 4   Type of Travel                     480 non-null    object
 5   Class                              480 non-null    object
 6   Flight Distance                    480 non-null    int64
 7   Inflight wifi service              480 non-null    int64
 8   Departure/Arrival time convenient  480 non-null    int64
 9   Ease of Online booking             480 non-null    int64
 10  Gate location                      480 non-null    int64
 11  Food and drink                     480 non-null    int64
 12  Online boarding                    480 non-null    int64
 13  Seat comfort                       480 non-null    int64
 14  Inflight entertainment             480 non-null    int64
 15  On-board service                   480 non-null    int64
 16  Leg room service                   480 non-null    int64
 17  Baggage handling                   480 non-null    int64
 18  Checkin service                    480 non-null    int64
 19  Inflight service                   480 non-null    int64
 20  Cleanliness                        480 non-null    int64
 21  Departure Delay in Minutes         480 non-null    int64
 22  Arrival Delay in Minutes           480 non-null    float64
```

```
 23  satisfaction                      480 non-null    object
dtypes: float64(1), int64(18), object(5)
memory usage: 90.1+ KB
```

[148]:
```
df.describe()
```

[148]:

	id	Age	Flight Distance	Inflight wifi service
count	480.00000	480.000000	480.000000	480.000000
mean	64794.62500	40.145833	1239.781250	2.739583
std	37397.13963	15.481135	1033.088286	1.303959
min	738.00000	7.000000	77.000000	0.000000
25%	31613.25000	27.750000	414.000000	2.000000
50%	65058.00000	41.000000	903.500000	3.000000
75%	96744.75000	52.000000	1862.750000	4.000000
max	129240.00000	80.000000	4817.000000	5.000000

	Departure/Arrival time convenient	Ease of Online booking
count	480.000000	480.000000
mean	3.039583	2.670833
std	1.508672	1.343034
min	0.000000	0.000000
25%	2.000000	2.000000
50%	3.000000	3.000000
75%	4.000000	4.000000
max	5.000000	5.000000

	Gate location	Food and drink	Online boarding	Seat comfort
count	480.000000	480.000000	480.000000	480.000000
mean	2.891667	3.239583	3.264583	3.462500
std	1.271468	1.309551	1.373320	1.339402
min	1.000000	1.000000	0.000000	1.000000
25%	2.000000	2.000000	2.000000	2.000000
50%	3.000000	3.000000	4.000000	4.000000
75%	4.000000	4.000000	4.000000	5.000000
max	5.000000	5.000000	5.000000	5.000000

	Inflight entertainment	On-board service	Leg room service
count	480.000000	480.000000	480.000000
mean	3.404167	3.402083	3.358333
std	1.334926	1.314285	1.289403
min	1.000000	1.000000	0.000000
25%	2.000000	2.000000	2.000000
50%	4.000000	4.000000	4.000000
75%	5.000000	4.000000	4.000000
max	5.000000	5.000000	5.000000

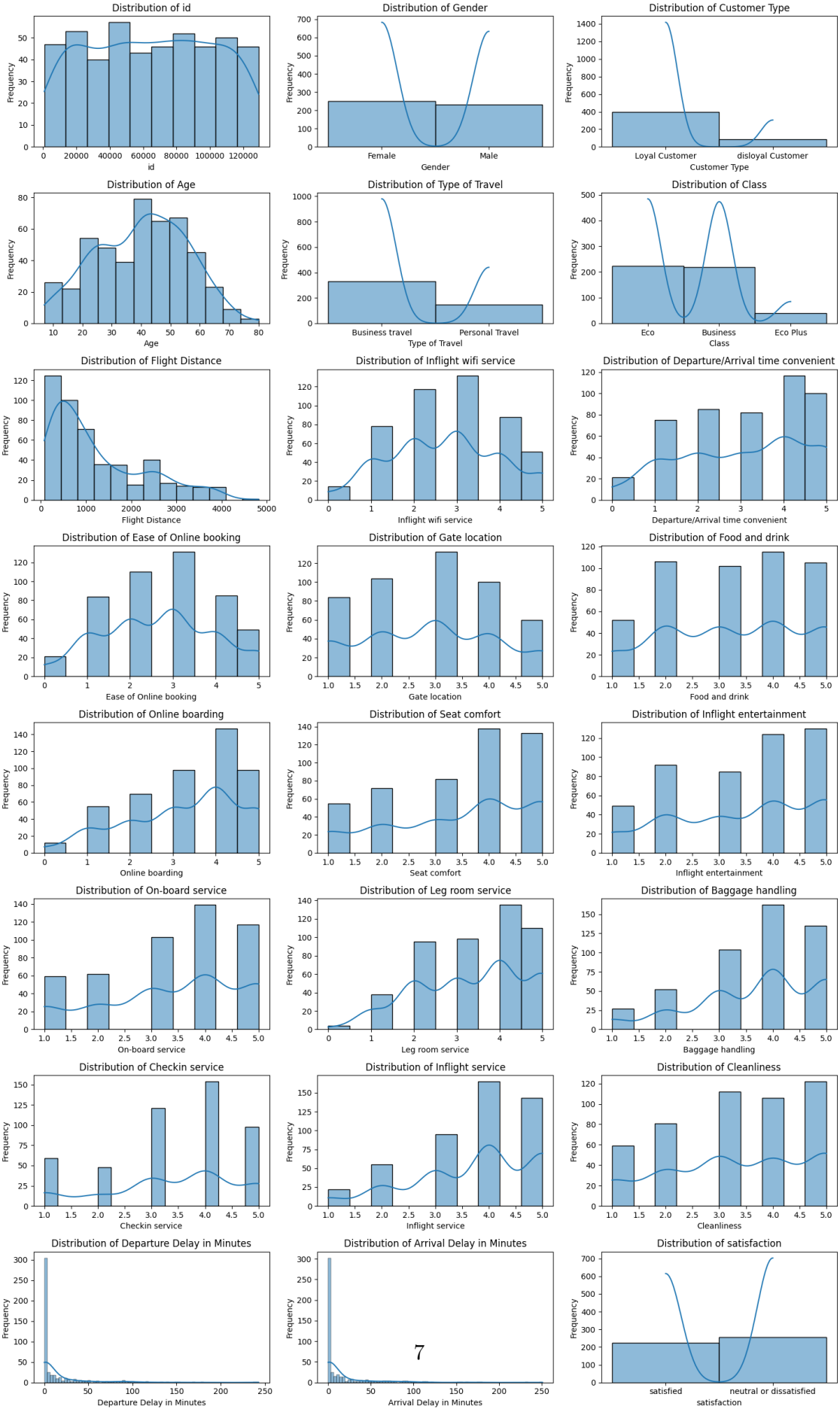
	Baggage handling	Checkin service	Inflight service	Cleanliness

```
count        480.000000        480.000000        480.000000   480.000000
mean           3.679167          3.383333          3.733333     3.314583
std            1.156439          1.259176          1.139412     1.342650
min            1.000000          1.000000          1.000000     1.000000
25%            3.000000          3.000000          3.000000     2.000000
50%            4.000000          4.000000          4.000000     3.000000
75%            5.000000          4.000000          5.000000     5.000000
max            5.000000          5.000000          5.000000     5.000000

       Departure Delay in Minutes  Arrival Delay in Minutes
count                  480.000000                480.000000
mean                    14.691667                 15.231250
std                     32.132631                 32.708344
min                      0.000000                  0.000000
25%                      0.000000                  0.000000
50%                      0.000000                  0.000000
75%                     13.000000                 13.250000
max                    243.000000                251.000000
```

[149]:
```
df.isnull().sum()
```

[149]:
```
id                                  0
Gender                              0
Customer Type                       0
Age                                 0
Type of Travel                      0
Class                               0
Flight Distance                     0
Inflight wifi service               0
Departure/Arrival time convenient   0
Ease of Online booking              0
Gate location                       0
Food and drink                      0
Online boarding                     0
Seat comfort                        0
Inflight entertainment              0
On-board service                    0
Leg room service                    0
Baggage handling                    0
Checkin service                     0
Inflight service                    0
Cleanliness                         0
Departure Delay in Minutes          0
Arrival Delay in Minutes            0
satisfaction                        0
dtype: int64
```

[150]:
```
plt.figure(figsize=(15, 25))
for i, col in enumerate(df.columns):
    plt.subplot(8, 3, i+1)
    sns.histplot(df[col].dropna(), kde=True)
    plt.title(f"Distribution of {col}")
    plt.xlabel(col)
    plt.ylabel('Frequency')
plt.tight_layout()
plt.show()
```


Distribution of id
Frequency
id
Distribution of Gender
Female
Male
Gender
Distribution of Customer Type
Loyal Customer
disloyal Customer
Customer Type
Distribution of Age
Age
Distribution of Type of Travel
Business travel
Personal Travel
Type of Travel
Distribution of Class
Eco
Business
Eco Plus
Class
Distribution of Flight Distance
Flight Distance
Distribution of Inflight wifi service
Inflight wifi service
Distribution of Departure/Arrival time convenient
Departure/Arrival time convenient
Distribution of Ease of Online booking
Ease of Online booking
Distribution of Gate location
Gate location
Distribution of Food and drink
Food and drink
Distribution of Online boarding
Online boarding
Distribution of Seat comfort
Seat comfort
Distribution of Inflight entertainment
Inflight entertainment
Distribution of On-board service
On-board service
Distribution of Leg room service
Leg room service
Distribution of Baggage handling
Baggage handling
Distribution of Checkin service
Checkin service
Distribution of Inflight service
Inflight service
Distribution of Cleanliness
Cleanliness
Distribution of Departure Delay in Minutes
Departure Delay in Minutes
Distribution of Arrival Delay in Minutes
Arrival Delay in Minutes
Distribution of satisfaction
satisfied
neutral or dissatisfied
satisfaction

[151]:
```
from sklearn.preprocessing import LabelEncoder

categorical_columns = df.select_dtypes(include=['object']).columns
label_encoder = LabelEncoder()

for col in categorical_columns:
    df[col] = label_encoder.fit_transform(df[col])

df.head()
```

[151]:

	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	\
0	19556	0	0	52	0	1	160	
1	90035	0	0	36	0	0	2863	
2	12360	1	1	20	0	1	192	
3	77959	1	0	44	0	0	3377	
4	36875	0	0	49	0	1	1182	

	Inflight wifi service	Departure/Arrival time convenient	\
0	5	4	
1	1	1	
2	2	0	
3	0	0	
4	2	3	

	Ease of Online booking	…	Inflight entertainment	On-board service	\
0	3	…	5	5	
1	3	…	4	4	
2	2	…	2	4	
3	0	…	1	1	
4	4	…	2	2	

	Leg room service	Baggage handling	Checkin service	Inflight service	\
0	5	5	2	5	
1	4	4	3	4	
2	1	3	2	2	
3	1	1	3	1	
4	2	2	4	2	

	Cleanliness	Departure Delay in Minutes	Arrival Delay in Minutes	\
0	5	50	44.0	
1	5	0	0.0	
2	2	0	0.0	
3	4	0	6.0	
4	4	0	20.0	

```
   satisfaction
0             1
1             1
2             0
3             1
4             1

[5 rows x 24 columns]
```

[152]:
```
for column in df.columns:
    skewness = df[column].skew()
    kurt = df[column].kurtosis()
    print(f"\n{column} - Skewness: {skewness}, Kurtosis: {kurt}")
```

```
id - Skewness: -0.011902900161568121, Kurtosis: -1.2067539775469216

Gender - Skewness: 0.075288270434495, Kurtosis: -2.0026937035271946

Customer Type - Skewness: 1.6971262932269415, Kurtosis: 0.8839032197745755

Age - Skewness: -0.10487607565488108, Kurtosis: -0.6633845759482861

Type of Travel - Skewness: 0.8221001655747, Kurtosis: -1.3297092185412962

Class - Skewness: 0.4919059721241659, Kurtosis: -0.6524905475880307

Flight Distance - Skewness: 1.0468835364163775, Kurtosis: 0.1126561912316566

Inflight wifi service - Skewness: 0.020579537658566273, Kurtosis:
-0.7578975038310425

Departure/Arrival time convenient - Skewness: -0.2835801589510711, Kurtosis:
-1.0739424222790412

Ease of Online booking - Skewness: -0.0014615039303219953, Kurtosis:
-0.7742953739673752

Gate location - Skewness: 0.05739987201268121, Kurtosis: -1.0146521356167675

Food and drink - Skewness: -0.15310844476352445, Kurtosis: -1.1458764815535598

Online boarding - Skewness: -0.5090815116677345, Kurtosis: -0.6886578892136472

Seat comfort - Skewness: -0.47967839339855195, Kurtosis: -0.9759208034938105

Inflight entertainment - Skewness: -0.34075814707603247, Kurtosis:
```

```
-1.117856483609738

On-board service - Skewness: -0.4508715363320918, Kurtosis: -0.9024680658283302

Leg room service - Skewness: -0.3620126832102195, Kurtosis: -0.8536156997120594

Baggage handling - Skewness: -0.6527944041143089, Kurtosis: -0.3721077094657863

Checkin service - Skewness: -0.5065791687315574, Kurtosis: -0.6863552684565484

Inflight service - Skewness: -0.6794182930536473, Kurtosis: -0.3587886510034157

Cleanliness - Skewness: -0.2605957647521839, Kurtosis: -1.1036742802306947

Departure Delay in Minutes - Skewness: 3.3986894539642982, Kurtosis:
14.47239178834231

Arrival Delay in Minutes - Skewness: 3.3135104007245983, Kurtosis:
13.913852814754186

satisfaction - Skewness: 0.1340498910304727, Kurtosis: -1.9903411846304795
```

0.0.1 Statistical Values

The dataset consists of 480 rows and 24 columns, with no missing values, confirming data completeness.

0.0.2 Visualization

The histograms provide a visual representation of the distribution of various features in the dataset. Example: - Inflight wifi service: The distribution is relatively even, suggesting a variety of opinions on inflight wifi service. - Ease of Online Booking: The distribution is skewed to the right, indicating that most passengers found the booking process easy.

0.0.3 Attributes and Special Treatment

The analysis of skewness values indicates that certain attributes in the dataset may require special treatment for effective analysis. Here is a breakdown based on the skewness and kurtosis values:

1. **Customer Type**
   - **Skewness**: 1.697
   - **Kurtosis**: 0.884
   - The positive skewness suggests a concentration of values on the left. This attribute might benefit from transformation (e.g., log transformation) or binning to better balance the distribution.
2. **Flight Distance**
   - **Skewness**: 1.047
   - **Kurtosis**: 0.113

- Similar to Customer Type, the positive skewness indicates potential outliers or a long tail. A log transformation could help normalize this attribute.

3. **Departure Delay in Minutes**
   - **Skewness**: 3.399
   - **Kurtosis**: 14.472
   - This high positive skewness signals extreme outliers and a highly non-normal distribution. A log transformation or capping of extreme values is can be used to reduce skewness and improve analysis.

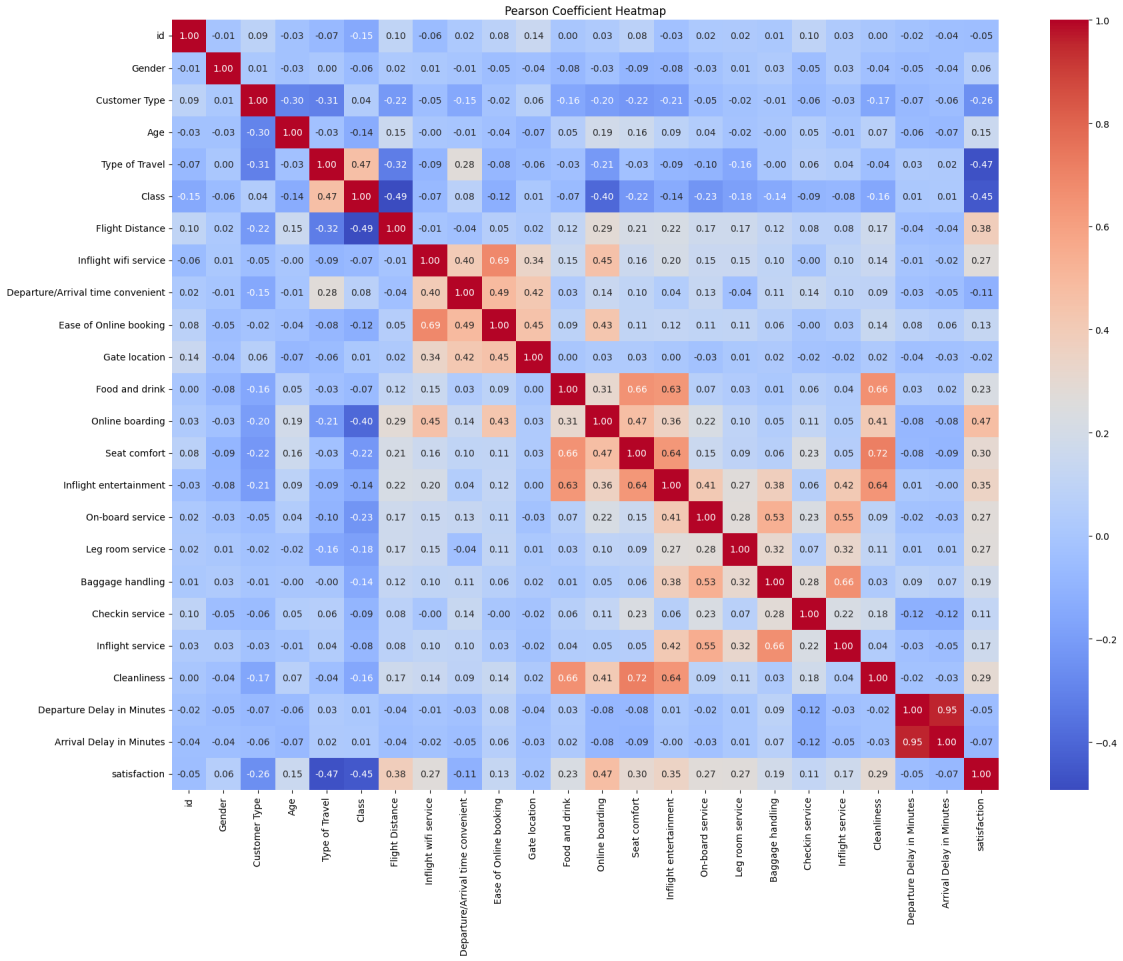
0.0.4 Attributes with Minimal Treatment Required

The following attributes have skewness values close to zero, indicating a roughly symmetrical distribution. Therefore, they likely do not require special treatment:

- **ID**: Skewness -0.012
- **Gender**: Skewness 0.075
- **Age**: Skewness -0.105
- **Type of Travel**: Skewness 0.822 (might still need monitoring)
- **Class**: Skewness 0.492
- **Inflight Wifi Service**: Skewness 0.021
- **Departure/Arrival Time Convenient**: Skewness -0.284
- **Ease of Online Booking**: Skewness -0.001
- **Gate Location**: Skewness 0.057
- **Food and Drink**: Skewness -0.153
- **Online Boarding**: Skewness -0.509
- **Seat Comfort**: Skewness -0.480
- **Inflight Entertainment**: Skewness -0.341
- **On-board Service**: Skewness -0.451
- **Leg Room Service**: Skewness -0.362
- **Baggage Handling**: Skewness -0.653
- **Check-in Service**: Skewness -0.507
- **Inflight Service**: Skewness -0.679
- **Cleanliness**: Skewness -0.261

Q2. Analyze and discuss the relationships between the data attributes and between the data attributes and labels. This involves computing the Pearson Correlation Coefficient (PCC) and generating scatter plots.

[153]:
```
correlation_matrix = df.corr()
plt.figure(figsize=(20,15))
sns.heatmap(correlation_matrix,annot = True, cmap= 'coolwarm' , fmt = '.2f')
plt.title('Pearson Coefficient Heatmap')
plt.show()
```



[154]:
```
df.corr(method='pearson', numeric_only=True)
```

[154]:

	id	Gender	Customer Type	\
id	1.000000	-0.012723	0.087495	
Gender	-0.012723	1.000000	0.011947	
Customer Type	0.087495	0.011947	1.000000	
Age	-0.026375	-0.025799	-0.298686	
Type of Travel	-0.068802	0.002647	-0.311236	
Class	-0.148805	-0.058655	0.040725	
Flight Distance	0.103495	0.015937	-0.218262	
Inflight wifi service	-0.057168	0.013304	-0.049709	
Departure/Arrival time convenient	0.016597	-0.005931	-0.153410	
Ease of Online booking	0.081093	-0.046501	-0.016356	
Gate location	0.142457	-0.036028	0.061050	
Food and drink	0.002179	-0.077591	-0.155877	
Online boarding	0.030492	-0.030754	-0.200851	
Seat comfort	0.078283	-0.089866	-0.221531	

```
Inflight entertainment            -0.031159 -0.082428      -0.206074
On-board service                   0.016720 -0.025030      -0.054774
Leg room service                   0.022044  0.007203      -0.023127
Baggage handling                   0.011616  0.029281      -0.008169
Checkin service                    0.097661 -0.048231      -0.058934
Inflight service                   0.027703  0.031504      -0.025572
Cleanliness                        0.001964 -0.036275      -0.165767
Departure Delay in Minutes        -0.024010 -0.050111      -0.072053
Arrival Delay in Minutes          -0.039619 -0.042293      -0.055563
satisfaction                      -0.045262  0.060176      -0.258897

                                        Age  Type of Travel     Class  \
id                                -0.026375       -0.068802 -0.148805
Gender                            -0.025799        0.002647 -0.058655
Customer Type                     -0.298686       -0.311236  0.040725
Age                                1.000000       -0.031076 -0.136282
Type of Travel                    -0.031076        1.000000  0.468946
Class                             -0.136282        0.468946  1.000000
Flight Distance                    0.153484       -0.319604 -0.493463
Inflight wifi service             -0.002872       -0.090563 -0.070153
Departure/Arrival time convenient -0.006505        0.275182  0.077038
Ease of Online booking            -0.037750       -0.083753 -0.120664
Gate location                     -0.073757       -0.059766  0.009401
Food and drink                     0.051100       -0.033381 -0.066059
Online boarding                    0.188974       -0.214735 -0.397132
Seat comfort                       0.164980       -0.029994 -0.220641
Inflight entertainment             0.089171       -0.088539 -0.143069
On-board service                   0.042669       -0.099154 -0.231952
Leg room service                  -0.017997       -0.155188 -0.184612
Baggage handling                  -0.002862       -0.000763 -0.144455
Checkin service                    0.048640        0.060439 -0.093086
Inflight service                  -0.007614        0.038506 -0.080629
Cleanliness                        0.070907       -0.043218 -0.159604
Departure Delay in Minutes        -0.063130        0.030994  0.012764
Arrival Delay in Minutes          -0.069026        0.023779  0.014421
satisfaction                       0.151305       -0.474160 -0.447513

                                  Flight Distance  Inflight wifi service  \
id                                       0.103495              -0.057168
Gender                                   0.015937               0.013304
Customer Type                           -0.218262              -0.049709
Age                                      0.153484              -0.002872
Type of Travel                          -0.319604              -0.090563
Class                                   -0.493463              -0.070153
Flight Distance                          1.000000              -0.013902
Inflight wifi service                   -0.013902               1.000000
Departure/Arrival time convenient       -0.035017               0.404270
```

Ease of Online booking	0.049877	0.690053
Gate location	0.019995	0.335525
Food and drink	0.118723	0.151537
Online boarding	0.285116	0.454752
Seat comfort	0.211043	0.158755
Inflight entertainment	0.216101	0.200915
On-board service	0.167157	0.147717
Leg room service	0.165783	0.147502
Baggage handling	0.119456	0.102305
Checkin service	0.081464	-0.001377
Inflight service	0.078503	0.104917
Cleanliness	0.166646	0.144671
Departure Delay in Minutes	-0.036721	-0.014377
Arrival Delay in Minutes	-0.043880	-0.024087
satisfaction	0.382585	0.270361

	Departure/Arrival time convenient	\
id	0.016597	
Gender	-0.005931	
Customer Type	-0.153410	
Age	-0.006505	
Type of Travel	0.275182	
Class	0.077038	
Flight Distance	-0.035017	
Inflight wifi service	0.404270	
Departure/Arrival time convenient	1.000000	
Ease of Online booking	0.490706	
Gate location	0.422339	
Food and drink	0.032174	
Online boarding	0.143055	
Seat comfort	0.104567	
Inflight entertainment	0.044906	
On-board service	0.131990	
Leg room service	-0.037356	
Baggage handling	0.109005	
Checkin service	0.141455	
Inflight service	0.098453	
Cleanliness	0.093812	
Departure Delay in Minutes	-0.031400	
Arrival Delay in Minutes	-0.045031	
satisfaction	-0.110465	

	Ease of Online booking	…	\
id	0.081093	…	
Gender	-0.046501	…	
Customer Type	-0.016356	…	
Age	-0.037750	…	

Type of Travel	-0.083753	…
Class	-0.120664	…
Flight Distance	0.049877	…
Inflight wifi service	0.690053	…
Departure/Arrival time convenient	0.490706	…
Ease of Online booking	1.000000	…
Gate location	0.448539	…
Food and drink	0.086479	…
Online boarding	0.431030	…
Seat comfort	0.111501	…
Inflight entertainment	0.116280	…
On-board service	0.107072	…
Leg room service	0.111655	…
Baggage handling	0.060902	…
Checkin service	-0.003004	…
Inflight service	0.027103	…
Cleanliness	0.142061	…
Departure Delay in Minutes	0.078722	…
Arrival Delay in Minutes	0.057673	…
satisfaction	0.126786	…

	Inflight entertainment	On-board service	\
id	-0.031159	0.016720	
Gender	-0.082428	-0.025030	
Customer Type	-0.206074	-0.054774	
Age	0.089171	0.042669	
Type of Travel	-0.088539	-0.099154	
Class	-0.143069	-0.231952	
Flight Distance	0.216101	0.167157	
Inflight wifi service	0.200915	0.147717	
Departure/Arrival time convenient	0.044906	0.131990	
Ease of Online booking	0.116280	0.107072	
Gate location	0.004940	-0.025101	
Food and drink	0.634754	0.065210	
Online boarding	0.360615	0.219689	
Seat comfort	0.636666	0.150304	
Inflight entertainment	1.000000	0.411707	
On-board service	0.411707	1.000000	
Leg room service	0.266207	0.275757	
Baggage handling	0.376276	0.534212	
Checkin service	0.061644	0.225830	
Inflight service	0.415515	0.549926	
Cleanliness	0.639431	0.087885	
Departure Delay in Minutes	0.009774	-0.023654	
Arrival Delay in Minutes	-0.000567	-0.028489	
satisfaction	0.349057	0.273326	

	Leg room service	Baggage handling	\
id	0.022044	0.011616	
Gender	0.007203	0.029281	
Customer Type	-0.023127	-0.008169	
Age	-0.017997	-0.002862	
Type of Travel	-0.155188	-0.000763	
Class	-0.184612	-0.144455	
Flight Distance	0.165783	0.119456	
Inflight wifi service	0.147502	0.102305	
Departure/Arrival time convenient	-0.037356	0.109005	
Ease of Online booking	0.111655	0.060902	
Gate location	0.009720	0.021747	
Food and drink	0.033125	0.013642	
Online boarding	0.097255	0.048303	
Seat comfort	0.092415	0.063651	
Inflight entertainment	0.266207	0.376276	
On-board service	0.275757	0.534212	
Leg room service	1.000000	0.315275	
Baggage handling	0.315275	1.000000	
Checkin service	0.072093	0.276750	
Inflight service	0.320957	0.663752	
Cleanliness	0.112018	0.026146	
Departure Delay in Minutes	0.013002	0.087448	
Arrival Delay in Minutes	0.011396	0.066817	
satisfaction	0.268226	0.187488	

	Checkin service	Inflight service	\
id	0.097661	0.027703	
Gender	-0.048231	0.031504	
Customer Type	-0.058934	-0.025572	
Age	0.048640	-0.007614	
Type of Travel	0.060439	0.038506	
Class	-0.093086	-0.080629	
Flight Distance	0.081464	0.078503	
Inflight wifi service	-0.001377	0.104917	
Departure/Arrival time convenient	0.141455	0.098453	
Ease of Online booking	-0.003004	0.027103	
Gate location	-0.019647	-0.021424	
Food and drink	0.064464	0.038710	
Online boarding	0.113866	0.045184	
Seat comfort	0.230116	0.050888	
Inflight entertainment	0.061644	0.415515	
On-board service	0.225830	0.549926	
Leg room service	0.072093	0.320957	
Baggage handling	0.276750	0.663752	
Checkin service	1.000000	0.222729	
Inflight service	0.222729	1.000000	

Cleanliness	0.180432	0.042668
Departure Delay in Minutes	-0.116625	-0.034753
Arrival Delay in Minutes	-0.115093	-0.053071
satisfaction	0.113319	0.167788

	Cleanliness	Departure Delay in Minutes	\
id	0.001964	-0.024010	
Gender	-0.036275	-0.050111	
Customer Type	-0.165767	-0.072053	
Age	0.070907	-0.063130	
Type of Travel	-0.043218	0.030994	
Class	-0.159604	0.012764	
Flight Distance	0.166646	-0.036721	
Inflight wifi service	0.144671	-0.014377	
Departure/Arrival time convenient	0.093812	-0.031400	
Ease of Online booking	0.142061	0.078722	
Gate location	0.023674	-0.035771	
Food and drink	0.661145	0.031081	
Online boarding	0.409917	-0.079283	
Seat comfort	0.717618	-0.078269	
Inflight entertainment	0.639431	0.009774	
On-board service	0.087885	-0.023654	
Leg room service	0.112018	0.013002	
Baggage handling	0.026146	0.087448	
Checkin service	0.180432	-0.116625	
Inflight service	0.042668	-0.034753	
Cleanliness	1.000000	-0.023974	
Departure Delay in Minutes	-0.023974	1.000000	
Arrival Delay in Minutes	-0.025001	0.954762	
satisfaction	0.294328	-0.052940	

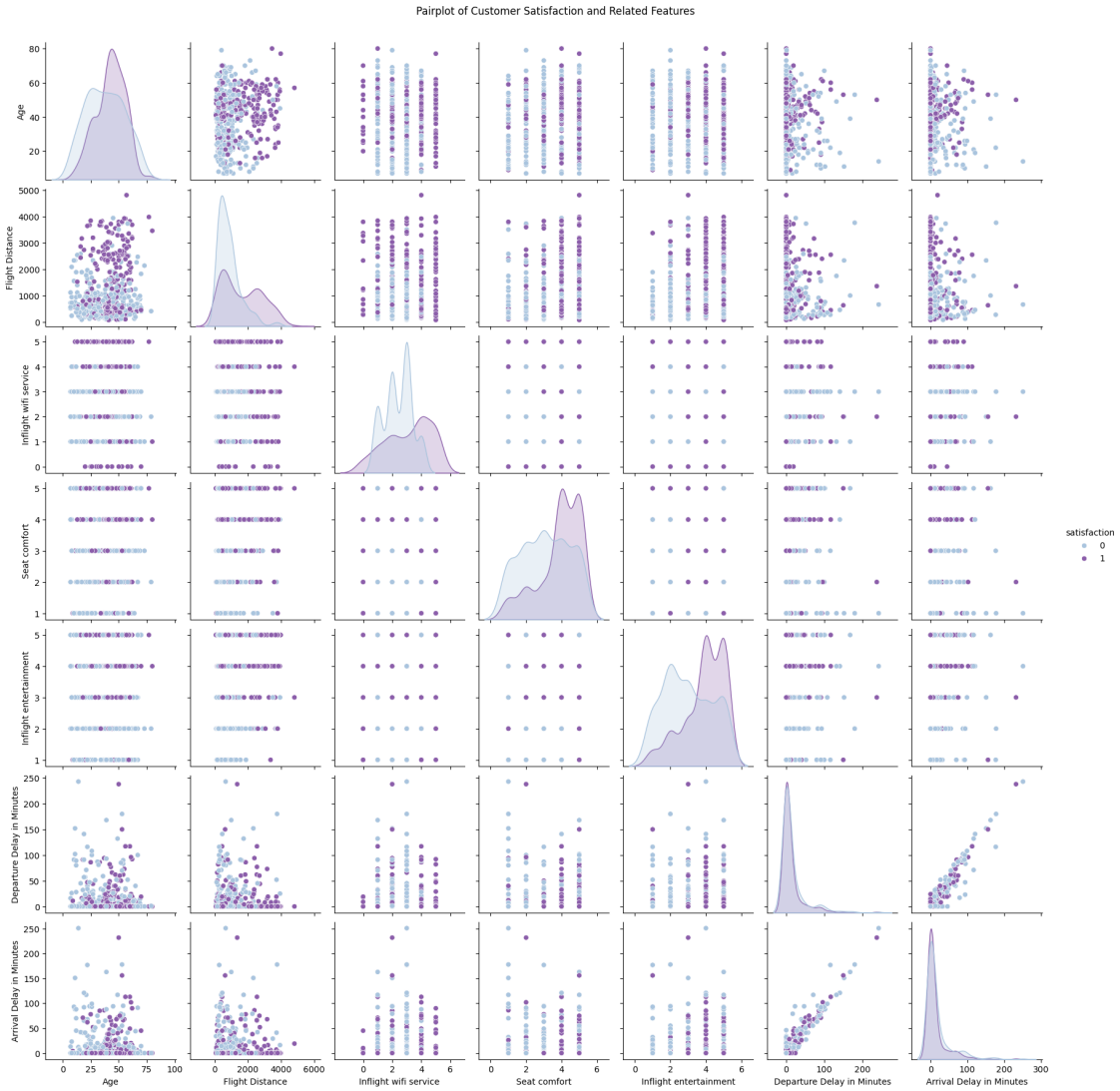
	Arrival Delay in Minutes	satisfaction
id	-0.039619	-0.045262
Gender	-0.042293	0.060176
Customer Type	-0.055563	-0.258897
Age	-0.069026	0.151305
Type of Travel	0.023779	-0.474160
Class	0.014421	-0.447513
Flight Distance	-0.043880	0.382585
Inflight wifi service	-0.024087	0.270361
Departure/Arrival time convenient	-0.045031	-0.110465
Ease of Online booking	0.057673	0.126786
Gate location	-0.030068	-0.015562
Food and drink	0.020685	0.227708
Online boarding	-0.082188	0.471000
Seat comfort	-0.088938	0.300867
Inflight entertainment	-0.000567	0.349057

```
On-board service                          -0.028489      0.273326
Leg room service                           0.011396      0.268226
Baggage handling                           0.066817      0.187488
Checkin service                           -0.115093      0.113319
Inflight service                          -0.053071      0.167788
Cleanliness                               -0.025001      0.294328
Departure Delay in Minutes                 0.954762     -0.052940
Arrival Delay in Minutes                   1.000000     -0.074741
satisfaction                              -0.074741      1.000000

[24 rows x 24 columns]
```

[155]:
```
df.columns
```

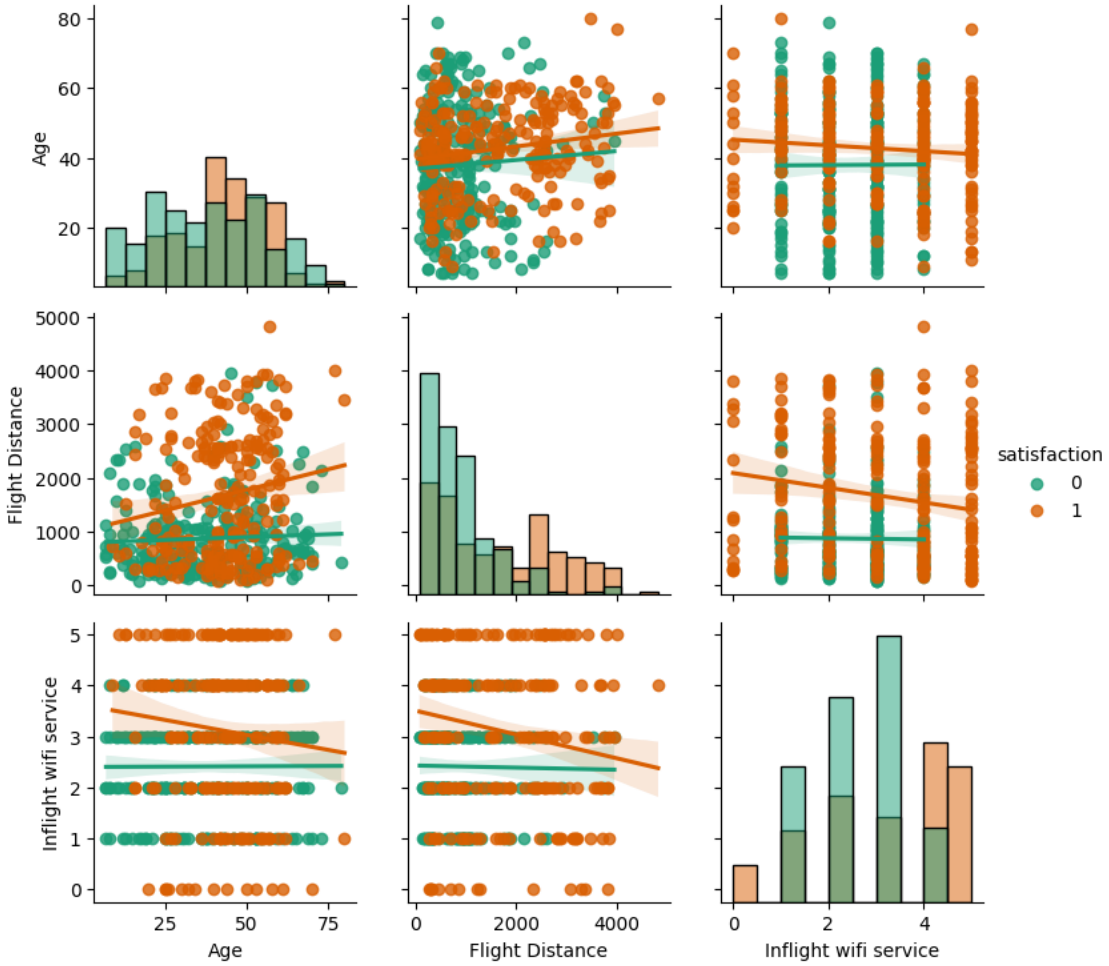
[155]:
```
Index(['id', 'Gender', 'Customer Type', 'Age', 'Type of Travel', 'Class',
       'Flight Distance', 'Inflight wifi service',
       'Departure/Arrival time convenient', 'Ease of Online booking',
       'Gate location', 'Food and drink', 'Online boarding', 'Seat comfort',
       'Inflight entertainment', 'On-board service', 'Leg room service',
       'Baggage handling', 'Checkin service', 'Inflight service',
       'Cleanliness', 'Departure Delay in Minutes', 'Arrival Delay in Minutes',
       'satisfaction'],
      dtype='object')
```

[156]:
```
import seaborn as sns
import matplotlib.pyplot as plt

selected_columns = ['Age', 'Flight Distance', 'satisfaction',
                    'Inflight wifi service', 'Seat comfort',
                    'Inflight entertainment', 'Departure Delay in Minutes',
                    'Arrival Delay in Minutes']

sns.pairplot(df[selected_columns], hue='satisfaction', palette='BuPu')
plt.suptitle('Pairplot of Customer Satisfaction and Related Features', y=1.02)
plt.show()
```



[157]:
```
g = sns.PairGrid(df, vars=['Age', 'Flight Distance', 'Inflight wifi service'],␣
↪hue="satisfaction", palette="Dark2")
g.map_diag(sns.histplot)
g.map_offdiag(sns.regplot)
g.add_legend()
```

[157]: <seaborn.axisgrid.PairGrid at 0x2120d217740>



Relationship between the data attributes and between the data attributes and labels

0.0.5 Heatmap

1. Color Intensity: The color intensity represents the strength of the correlation. Red indicates a positive correlation, blue indicates a negative correlation, and white indicates no correlation.
2. Diagonal: The diagonal line represents the correlation of each feature with itself, which is always 1.
3. High Correlations: Some features show high positive correlations, such as:
   - Online boarding and Gate location
   - Seat comfort and Inflight entertainment
   - Checkin service and Inflight service
4. Negative Correlations: Some features show negative correlations, such as:
   - Departure Delay and Satisfaction
   - Arrival Delay and Satisfaction
5. Low Correlations: Many features have low or no correlation with each other, indicating that they are independent or have a weak relationship.

0.0.6 Correlation matrix

1. Customer Type and Satisfaction: There is a moderate negative correlation (-0.259) between customer type and satisfaction, indicating that certain customer categories may report lower satisfaction levels.
2. Type of Travel and Class: A strong positive correlation (0.47) indicates that passengers traveling for business tend to fly in higher classes. Conversely, the correlation between Type of Travel and Age (-0.31) reinforces the notion that older passengers may prefer leisure travel.
3. Inflight Services: Inflight wifi service, online boarding, and inflight entertainment show strong positive correlations (0.69, 0.45, and 0.64 respectively) with customer satisfaction. This implies that passengers who value connectivity and entertainment are more likely to report higher satisfaction.
4. Flight Distance and Satisfaction: A positive correlation (0.383) indicates that longer flight distances may be associated with higher satisfaction levels, possibly due to better service or amenities on longer flights.
5. Delay Variables: Both departure delay (-0.053) and arrival delay (-0.075) have weak negative correlations with satisfaction, suggesting that delays slightly reduce overall satisfaction.
6. Baggage Handling and Check-in Service: There's a positive correlation between baggage handling and check-in service (0.27), highlighting that good experiences in these areas may go hand-in-hand.

0.0.7 PairPlot

Longer flights seem to have a slight negative impact on satisfaction. While Comfortable seats and Inflight entertainment or Wifi lead to higher satisfaction. Overall, the pairplot suggests that factors like inflight entertainment, seat comfort, and wifi service play a significant role in determining customer satisfaction.

0.0.8 PairGrid

There seems to be a slight positive correlation between age and flight distance, suggesting that older passengers tend to travel longer distances.

Q3. For training data, use token numbers 1-10, for validation 11 to 13, and for testing 14 to 16 (each of the 30 rock subtypes has 16 token numbers).

[158]:
```
from sklearn.model_selection import train_test_split

train_val_data, test_data = train_test_split(df, test_size=0.2, random_state=42)
train_data, val_data = train_test_split(train_val_data, test_size=0.25,␣
 ↪random_state=42)

print(f"Training set size: {train_data.shape[0]}")
print(f"Validation set size: {val_data.shape[0]}")
print(f"Test set size: {test_data.shape[0]}")
```

```
X_train, Y_train = train_data.drop('satisfaction', axis=1),␣
 ↪train_data['satisfaction']
X_val, Y_val = val_data.drop('satisfaction', axis=1), val_data['satisfaction']
X_test, Y_test = test_data.drop('satisfaction', axis=1),␣
 ↪test_data['satisfaction']
```

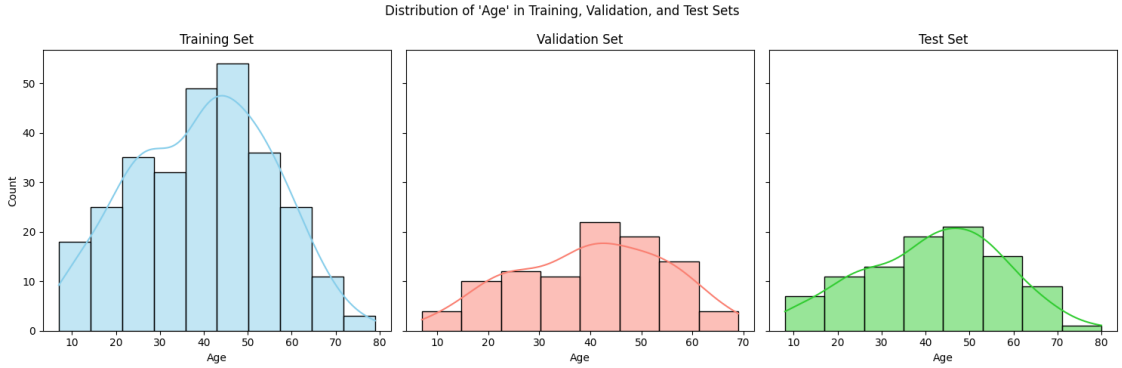
```
Training set size: 288
Validation set size: 96
Test set size: 96
```

[159]:
```
print(Y_train.value_counts())
print(Y_val.value_counts())
print(Y_test.value_counts())
```

```
satisfaction
0    152
1    136
Name: count, dtype: int64
satisfaction
0    56
1    40
Name: count, dtype: int64
satisfaction
1    48
0    48
Name: count, dtype: int64
```

[160]:
```
import matplotlib.pyplot as plt
import seaborn as sns

fig, axes = plt.subplots(1, 3, figsize=(15, 5), sharey=True)
fig.suptitle("Distribution of 'Age' in Training, Validation, and Test Sets")

sns.histplot(X_train['Age'], kde=True, ax=axes[0], color="skyblue",␣
 ↪label="Train")
axes[0].set_title("Training Set")
axes[0].set_xlabel("Age")

sns.histplot(X_val['Age'], kde=True, ax=axes[1], color="salmon",␣
 ↪label="Validation")
axes[1].set_title("Validation Set")
axes[1].set_xlabel("Age")

sns.histplot(X_test['Age'], kde=True, ax=axes[2], color="limegreen",␣
 ↪label="Test")
axes[2].set_title("Test Set")
axes[2].set_xlabel("Age")
```

```
plt.tight_layout()
plt.show()
```



0.0.9 Splitting the Data

The dataset was split into 60% for training, 20% for validation, and 20% for testing. Initially, 80% of the data was used for combined training and validation, then further split into 75% training and 25% validation. This ensures the model is trained, tuned, and tested on distinct subsets of the data, enhancing its performance evaluation. The target variable 'satisfaction' was separated from the features for each set.

0.0.10 Verification of Splits

The histograms show the distribution of age in the training, validation, and test sets. We can observe that the age distribution is similar across all three sets, with a peak around 40-50 years old. This suggests that the data splitting process has preserved the original distribution of age in the dataset.

Q4. Train different classifiers and tweak the hyperparameters to improve performance (you can use the grid search if you want or manually try different values). Report training, validation and testing performance (classification accuracy, precision, recall and F1 score) and discuss the impact of the hyperparameters (use markdown cells in Jupyter Notebook to clearly indicate each solution):

[161]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler = StandardScaler()
X_train_scaled = pd.DataFrame(scaler.fit_transform(X_train), columns=X_train.
 ↪columns)
X_val_scaled = pd.DataFrame(scaler.transform(X_val), columns=X_val.columns)
X_test_scaled = pd.DataFrame(scaler.transform(X_test), columns=X_test.columns)

print(X_train_scaled)
print(X_val_scaled)
print(X_test_scaled)
```

```
           id    Gender  Customer Type       Age  Type of Travel     Class  \
0   -0.337013 -0.939336      -0.502169  0.657086        1.612452  0.553585
1    1.190323 -0.939336      -0.502169  1.166310        1.612452  2.132123
2    1.606806  1.064581      -0.502169 -0.043098        1.612452  0.553585
3   -0.982226  1.064581      -0.502169 -0.425017       -0.620174 -1.024954
4   -1.677025 -0.939336      -0.502169 -1.125201        1.612452  0.553585
..        …       …             …       …              …       …
283  1.415219 -0.939336      -0.502169 -0.679629        1.612452  0.553585
284 -1.530492 -0.939336      -0.502169 -1.188854        1.612452  0.553585
285 -1.680378 -0.939336       1.991361 -0.743283       -0.620174  2.132123
286 -1.374063 -0.939336      -0.502169 -0.043098       -0.620174 -1.024954
287 -0.025147  1.064581      -0.502169  0.084208       -0.620174 -1.024954

     Flight Distance  Inflight wifi service  \
0          -1.076295              -0.571501
1          -0.216863               0.190500
2          -0.705222               0.190500
3           2.501976              -0.571501
4          -1.002485               0.190500
..               …                     …
283        -0.883175               0.190500
284        -0.812399               0.190500
285        -0.612202              -0.571501
286         2.423111              -0.571501
287         0.416084               0.952501

     Departure/Arrival time convenient  Ease of Online booking  …  \
0                             0.656658               -0.542510  …
1                             1.324919               -0.542510  …
2                             1.324919                0.952636  …
3                            -0.679862               -0.542510  …
4                             1.324919                0.205063  …
..                                 …                     …  …
283                           0.656658                0.205063  …
284                          -0.679862                1.700209  …
285                          -0.679862               -0.542510  …
286                          -0.679862               -0.542510  …
287                           0.656658                0.952636  …

     Seat comfort  Inflight entertainment  On-board service  Leg room service  \
0        1.153028               -1.044685         -1.018572         -1.041984
1        1.153028               -1.789411         -1.778544         -1.041984
2        0.415090                0.444767          1.261341          1.284306
3        1.153028                1.189493         -0.258601         -0.266554
4       -1.798723               -0.299959         -1.778544          1.284306
..            …                       …                 …                 …
283     -0.322848               -0.299959          1.261341          0.508876
284      1.153028                1.189493         -0.258601         -1.041984
```

```
285      0.415090                0.444767         -0.258601         -0.266554
286     -1.060786                0.444767          0.501370          0.508876
287      1.153028                0.444767          0.501370          1.284306

     Baggage handling  Checkin service  Inflight service  Cleanliness  \
0           -1.386795        -0.300837         -1.565414     1.196194
1           -2.242035         1.303626         -2.461715    -0.288737
2            0.323684        -0.300837          1.123488     0.453729
3            1.178924        -1.103068         -1.565414     1.196194
4           -2.242035        -0.300837         -1.565414    -0.288737
..                …               …                 …           …
283         -1.386795        -1.103068          0.227187    -0.288737
284         -0.531555         0.501395         -0.669113     1.196194
285         -0.531555        -1.905300         -0.669113     0.453729
286          0.323684        -1.905300          0.227187    -0.288737
287          0.323684        -0.300837          0.227187     1.196194

     Departure Delay in Minutes  Arrival Delay in Minutes
0                     -0.442967                -0.451493
1                     -0.442967                -0.451493
2                     -0.442967                -0.451493
3                     -0.442967                -0.451493
4                     -0.442967                 0.062459
..                          …                        …
283                   -0.442967                -0.451493
284                    1.675814                 3.085706
285                   -0.442967                 0.939201
286                   -0.442967                -0.451493
287                   -0.383283                -0.270098

[288 rows x 23 columns]
          id    Gender  Customer Type       Age  Type of Travel     Class  \
0   0.273284 -0.939336      -0.502169 -2.079998        1.612452  0.553585
1  -0.053091  1.064581       1.991361 -1.061548       -0.620174 -1.024954
2   0.061046  1.064581      -0.502169 -0.615976        1.612452  2.132123
3  -0.182809 -0.939336      -0.502169  1.229964       -0.620174 -1.024954
4   1.644646  1.064581      -0.502169 -0.997895       -0.620174 -1.024954
..       …       …             …       …               …         …
91  0.197335  1.064581      -0.502169 -1.570773        1.612452  2.132123
92  0.844881  1.064581      -0.502169 -1.379814        1.612452  0.553585
93  1.075489 -0.939336      -0.502169 -0.743283       -0.620174 -1.024954
94  0.857942  1.064581      -0.502169 -0.234058        1.612452  0.553585
95 -0.609108 -0.939336      -0.502169  1.739188        1.612452  0.553585

    Flight Distance  Inflight wifi service  Departure/Arrival time convenient  \
0         -0.696123              -0.571501                           0.656658
1         -0.932719               0.952501                           1.324919
2         -0.978219              -1.333501                          -1.348122
```

```
3         -0.251240              -0.571501                          -1.348122
4          2.502987               0.952501                          -1.348122
..              …                      …                                  …
91        -1.023718              -0.571501                           0.656658
92        -0.416049               0.952501                           0.656658
93        -0.182485              -1.333501                          -1.348122
94        -0.394816              -1.333501                          -2.016382
95        -0.584902               0.190500                           0.656658

    Ease of Online booking  …  Seat comfort  Inflight entertainment  \
0                -0.542510  …      1.153028                1.189493
1                 0.952636  …      0.415090                0.444767
2                -1.290083  …      1.153028                0.444767
3                -1.290083  …     -0.322848               -1.044685
4                 0.952636  …     -1.060786               -1.044685
..                     …  …           …                       …
91               -0.542510  …      1.153028                1.189493
92                0.952636  …      0.415090               -0.299959
93               -1.290083  …      0.415090                0.444767
94               -1.290083  …      1.153028                1.189493
95                0.205063  …      1.153028               -0.299959

    On-board service  Leg room service  Baggage handling  Checkin service  \
0           1.261341          0.508876          1.178924         1.303626
1          -0.258601         -0.266554          1.178924        -0.300837
2          -1.778544          0.508876         -0.531555         0.501395
3          -1.018572         -1.041984         -1.386795         0.501395
4           1.261341          1.284306          1.178924         1.303626
..               …                 …                 …                …
91          1.261341          0.508876          1.178924         1.303626
92         -0.258601          0.508876          1.178924         0.501395
93          1.261341         -0.266554          1.178924         1.303626
94          0.501370          1.284306          1.178924        -1.905300
95         -0.258601         -0.266554         -0.531555        -0.300837

    Inflight service  Cleanliness  Departure Delay in Minutes  \
0           0.227187     1.196194                   -0.442967
1           0.227187     0.453729                   -0.174389
2          -0.669113     0.453729                   -0.442967
3          -1.565414    -1.773667                   -0.413125
4           0.227187    -1.031202                   -0.442967
..               …           …                          …
91          1.123488     1.196194                   -0.293757
92          0.227187    -0.288737                    2.272653
93          0.227187     0.453729                   -0.442967
94          1.123488     1.196194                    0.511976
95         -0.669113     1.196194                   -0.442967
```

```
    Arrival Delay in Minutes
0                  -0.451493
1                  -0.421260
2                  -0.451493
3                  -0.058470
4                  -0.451493
..                       …
91                 -0.209633
92                  1.695013
93                 -0.360795
94                 -0.088703
95                 -0.451493

[96 rows x 23 columns]
          id    Gender  Customer Type       Age  Type of Travel     Class  \
0   1.447588  1.064581      -0.502169  0.275167       -0.620174 -1.024954
1  -0.195923  1.064581      -0.502169  0.466126       -0.620174 -1.024954
2  -0.721448  1.064581      -0.502169 -1.507120       -0.620174 -1.024954
3   1.473146 -0.939336      -0.502169  0.784392        1.612452  0.553585
4   1.639766  1.064581       1.991361 -1.188854       -0.620174  0.553585
..       …       …             …       …               …       …
91 -1.386426 -0.939336      -0.502169  1.420923       -0.620174 -1.024954
92  0.515289 -0.939336      -0.502169 -1.698079       -0.620174 -1.024954
93  0.524219  1.064581      -0.502169  0.911698       -0.620174  0.553585
94 -1.572382  1.064581      -0.502169  1.420923        1.612452  0.553585
95 -0.529244  1.064581      -0.502169  0.784392        1.612452  0.553585

    Flight Distance  Inflight wifi service  Departure/Arrival time convenient  \
0         -0.847787               0.190500                          -0.011602
1          0.648636              -0.571501                          -0.679862
2          1.678944              -0.571501                          -0.679862
3          1.086441              -0.571501                           1.324919
4          0.574826               0.190500                          -0.011602
..              …                     …                                  …
91         2.515121              -0.571501                          -0.679862
92         1.354382               0.190500                          -1.348122
93        -0.683989              -0.571501                          -0.011602
94        -0.499970               0.190500                           1.324919
95        -1.091461               0.190500                           1.324919

    Ease of Online booking  …  Seat comfort  Inflight entertainment  \
0                -0.542510  …      0.415090                0.444767
1                -0.542510  …      0.415090                0.444767
2                -0.542510  …      1.153028                1.189493
3                -0.542510  …     -1.060786                1.189493
4                 0.205063  …      0.415090                0.444767
..                     …  …           …                       …
91                0.952636  …     -1.060786                0.444767
```

```
92                0.205063  …     -0.322848               -0.299959
93               -1.290083  …     -1.060786               -1.044685
94                0.205063  …     -1.060786               -1.044685
95                0.205063  …     -1.060786               -0.299959

    On-board service  Leg room service  Baggage handling  Checkin service  \
0           0.501370          0.508876          0.323684        -0.300837
1           0.501370          0.508876          0.323684         0.501395
2          -0.258601         -1.041984         -0.531555         0.501395
3          -0.258601          0.508876          1.178924        -1.103068
4          -0.258601          1.284306          0.323684        -0.300837
..               …                 …                 …               …
91          0.501370          0.508876          0.323684        -1.905300
92          0.501370         -0.266554          0.323684        -0.300837
93         -1.778544          0.508876         -0.531555        -1.103068
94         -0.258601          1.284306          1.178924         0.501395
95         -0.258601          1.284306         -1.386795        -1.905300

    Inflight service  Cleanliness  Departure Delay in Minutes  \
0           0.227187     0.453729                   -0.293757
1           0.227187     1.196194                    1.407236
2           1.123488     1.196194                   -0.383283
3           0.227187    -1.031202                    0.064347
4           1.123488     0.453729                   -0.442967
..               …            …                          …
91          0.227187    -1.031202                   -0.442967
92         -0.669113    -0.288737                   -0.442967
93         -1.565414    -1.031202                   -0.442967
94          0.227187    -1.031202                   -0.442967
95         -0.669113    -0.288737                   -0.442967

    Arrival Delay in Minutes
0                   0.274087
1                   1.211293
2                  -0.451493
3                   1.936872
4                  -0.451493
..                       …
91                 -0.451493
92                 -0.451493
93                 -0.451493
94                 -0.451493
95                 -0.451493

[96 rows x 23 columns]
```

A. Multinomial Logistic Regression (Softmax Regression); hyperparameters to explore: C, solver, max number of iterations.

[ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import precision_score, accuracy_score,f1_score,␣
 ↪recall_score

clf = LogisticRegression(multi_class='multinomial')

param_grid = {
    'C': [0.0001, 0.001, 0.01, 0.02],
    'solver': ['lbfgs', 'saga', "newton-cg"],
    'max_iter': [1, 5, 10, 100]
}

grid_search = GridSearchCV(estimator=clf, param_grid=param_grid, cv=5)

grid_search.fit(X_train_scaled, Y_train)

best_params = grid_search.best_params_
print("best param:",best_params, grid_search.best_score_)

best_clf = LogisticRegression(multi_class='multinomial', C=best_params['C'],
solver=best_params['solver'],max_iter=best_params['max_iter'])
best_clf.fit(X_train_scaled, Y_train)

def evaluate_model(model, X, y, set_name):
    y_pred_log = model.predict(X)
    accuracy_log = accuracy_score(y, y_pred_log)
    precision_log = precision_score(y, y_pred_log, average='weighted')
    recall_log = recall_score(y, y_pred_log, average='weighted')
    f1_log = f1_score(y, y_pred_log, average='weighted')
    print(f"{set_name} Metrics:")
    print(f"Accuracy: {accuracy_log}")
    print(f"Precision: {precision_log}")
    print(f"Recall: {recall_log}")
    print(f"F1 Score: {f1_log}")

evaluate_model(best_clf, X_train_scaled, Y_train, "Train")
evaluate_model(best_clf, X_val_scaled, Y_val, "Validation")
evaluate_model(best_clf, X_test_scaled, Y_test, "Test")
```

```
best param: {'C': 0.01, 'max_iter': 1, 'solver': 'saga'} 0.840411373260738
Train Metrics:
Accuracy: 0.8576388888888888
Precision: 0.8581045942827425
Recall: 0.8576388888888888
F1 Score: 0.8573607824897032
Validation Metrics:
```

```
Accuracy: 0.8645833333333334
Precision: 0.8645976484959537
Recall: 0.8645833333333334
F1 Score: 0.8637116506681725
Test Metrics:
Accuracy: 0.8958333333333334
Precision: 0.8958333333333334
Recall: 0.8958333333333334
F1 Score: 0.8958333333333334
```

0.0.11 Impact of Hyperparameters on the Logistic Regression Model

1. **Regularization Parameter (C)**:
   The C parameter, set to 0.01, controls the inverse of the regularization strength. This moderate regularization helps prevent overfitting by penalizing large coefficients and promoting a more generalized model. A higher value might have led to overfitting, whereas a lower value could have resulted in underfitting.

2. **Solver**:
   The saga solver was selected. This iterative method is particularly effective for handling L1 and L2 regularization and is efficient for large datasets. The choice of solver significantly affects the model's performance, and in this case, saga proved to be the most suitable option, likely due to its robustness and efficiency with logistic regression problems.

3. **Maximum Iterations (max_iter)**:
   The number of maximum iterations was set to 1. This parameter determines how long the solver will iterate before stopping. If the iterations are too low, the solver might not converge to an optimal solution. The chosen value ensures convergence while maintaining a reasonable computation time, achieving a good balance.

Model Performance Metrics

- **Training Set**:
  - Accuracy: 85.76%
  - Precision: 85.81%
  - Recall: 85.76%
  - F1 Score: 85.74%
- **Validation Set**:
  - Accuracy: 86.46%
  - Precision: 86.46%
  - Recall: 86.46%
  - F1 Score: 86.37%
- **Test Set**:
  - Accuracy: 89.58%
  - Precision: 89.58%
  - Recall: 89.58%
  - F1 Score: 89.58%

These metrics indicate a well-generalized model with consistent performance across training, validation, and test sets. The chosen hyperparameters, particularly the regularization parameter C and

the solver saga, significantly contributed to preventing overfitting and ensuring model robustness. Further experimentation with different values and solvers may help in fine-tuning the model to meet specific performance goals.

B. Support Vector Machine (make sure to try using kernels); hyperparameters to explore: C, kernel, degree of polynomial kernel, gamma.

[ ]:
```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

svm_clf = SVC()

param_grid = {
    'C': [0.0001, 0.001],
    'kernel': ['linear', 'poly'],
    'degree': [1, 2],
    'gamma': [0.01, 0.1, 1]
}


grid_search = GridSearchCV(estimator=svm_clf,␣
 ↪param_grid=param_grid,cv=10,scoring='accuracy')
grid_search.fit(X_train_scaled, Y_train)
best_params = grid_search.best_params_
print("best parameters:", best_params, grid_search.best_score_)

best_svm_clf = SVC(**best_params)
best_svm_clf.fit(X_train_scaled, Y_train)


evaluate_model(best_svm_clf, X_train_scaled, Y_train, "Train")
evaluate_model(best_svm_clf, X_val_scaled, Y_val, "Validation")
evaluate_model(best_svm_clf, X_test_scaled, Y_test, "Test")
```

```
best parameters: {'C': 0.001, 'degree': 1, 'gamma': 0.01, 'kernel': 'linear'}
0.8086206896551724
Train Metrics:
Accuracy: 0.8333333333333334
Precision: 0.834313149127964
Recall: 0.8333333333333334
F1 Score: 0.8328065995688887
Validation Metrics:
Accuracy: 0.8229166666666666
Precision: 0.8233801717408274
Recall: 0.8229166666666666
F1 Score: 0.8207977207977208
Test Metrics:
Accuracy: 0.8125
```

```
Precision: 0.8130434782608695
Recall: 0.8125
F1 Score: 0.8124185844550587
```

0.0.12 Impact of Hyperparameters on the Support Vector Machine (SVM) Model

1. **Regularization Parameter (C)**:
   The C parameter, which controls the trade-off between achieving a low training error and a low testing error, was set to 0.001. A smaller value of C indicates stronger regularization, which helps prevent overfitting by penalizing larger coefficients. This choice helps the model generalize better to unseen data. In this scenario, the selected C value ensures that the model remains balanced and avoids overfitting while maintaining good performance.

2. **Kernel**:
   The kernel parameter was set to 'linear', indicating that a linear kernel is used. Different kernel functions can capture different types of relationships in the data. The linear kernel is effective when data is linearly separable, which suits this dataset.

3. **Degree**:
   The degree parameter, relevant for polynomial kernels, was set to 1. This means that the polynomial kernel effectively acts as a linear kernel. While higher degrees can capture more complex patterns, they also increase the risk of overfitting. The choice of degree 1 suggests that linear separability was sufficient for this particular dataset, providing a balance between complexity and generalizability.

4. **Gamma**:
   The gamma parameter, which defines how far the influence of a single training example reaches, was set to 0.01. A lower value of gamma means that the influence of each data point is more spread out, leading to a simpler decision boundary. This setting helps in maintaining generalizability while still capturing important patterns in the data.

Model Performance Metrics

- **Training Set**:
  - Accuracy: 0.833
  - Precision: 0.834
  - Recall: 0.833
  - F1 Score: 0.833
- **Validation Set**:
  - Accuracy: 0.823
  - Precision: 0.823
  - Recall: 0.823
  - F1 Score: 0.821
- **Test Set**:
  - Accuracy: 0.813
  - Precision: 0.813
  - Recall: 0.813
  - F1 Score: 0.812

These metrics indicate that the selected hyperparameters have produced a well-generalized model

with consistent performance across training, validation, and test sets. The chosen values for C, kernel, degree, and gamma effectively balance the complexity of the model and its ability to generalize to new data. Further tuning and experimentation might reveal even more optimal settings, but the current configuration provides a strong foundation for robust predictions.

C. Random Forest classifier (also analyze feature importance); hyperparameters to explore: the number of trees, max depth, the minimum number of samples required to split an internal node, the minimum number of samples required to be at a leaf node.

[ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score

rf_clf = RandomForestClassifier(random_state=42)

param_grid_rf = {
    'n_estimators': [50, 75, 100],
    'max_depth': [50, 70, 90],
    'min_samples_split': [60, 75, 90],
    'min_samples_leaf': [65, 75, 85],
    'max_features': ['sqrt', 'log2']
}


grid_search_rf = GridSearchCV(estimator=rf_clf, param_grid=param_grid_rf, cv=5,␣
 ↪scoring='accuracy', n_jobs=-1)
grid_search_rf.fit(X_train_scaled, Y_train)

best_params_rf = grid_search_rf.best_params_
print("Best parameters for Random Forest:", best_params_rf, grid_search_rf.
 ↪best_score_)

best_rf_clf = RandomForestClassifier(**best_params_rf, random_state=42)
best_rf_clf.fit(X_train_scaled, Y_train)

def evaluate_rf_model(model, X, y, set_name):
    y_pred_rf = model.predict(X)
    accuracy_rf = accuracy_score(y, y_pred_rf)
    precision_rf = precision_score(y, y_pred_rf, average='weighted')
    recall_rf = recall_score(y, y_pred_rf, average='weighted')
    f1_rf = f1_score(y, y_pred_rf, average='weighted')

    print(f"{set_name} Metrics:")
    print(f"Accuracy: {accuracy_rf}")
    print(f"Precision: {precision_rf}")
    print(f"Recall: {recall_rf}")
    print(f"F1 Score: {f1_rf}")
```

```
evaluate_rf_model(best_rf_clf, X_train_scaled, Y_train, "Train")
evaluate_rf_model(best_rf_clf, X_val_scaled, Y_val, "Validation")
evaluate_rf_model(best_rf_clf, X_test_scaled, Y_test, "Test")
```

```
Best parameters for Random Forest: {'max_depth': 50, 'max_features': 'sqrt',
'min_samples_leaf': 65, 'min_samples_split': 60, 'n_estimators': 50}
0.7708408953418028
Train Metrics:
Accuracy: 0.8020833333333334
Precision: 0.8025831010254153
Recall: 0.8020833333333334
F1 Score: 0.801542373773572
Validation Metrics:
Accuracy: 0.8125
Precision: 0.8117059891107079
Recall: 0.8125
F1 Score: 0.811740890688259
Test Metrics:
Accuracy: 0.7916666666666666
Precision: 0.7921739130434782
Recall: 0.7916666666666666
F1 Score: 0.7915762049500651
```

0.0.13 Impact of Hyperparameters on the Random Forest Model

1. **Number of Estimators (n_estimators)**:
   The parameter n_estimators was set to 50. This defines the number of trees in the forest. More trees generally improve the model's performance by reducing variance, but they also increase computational complexity. In this case, 50 trees were chosen to achieve a good balance between accuracy and computation time.

2. **Maximum Depth (max_depth)**:
   The max_depth was set to 50. This limits the depth of each tree, which helps control overfitting by restricting the model's complexity. While deeper trees can capture more intricate patterns, a depth of 50 provides a balance between complexity and generalization for this dataset.

3. **Minimum Samples per Split (min_samples_split)**:
   The parameter min_samples_split was set to 60. This specifies the minimum number of samples required to split an internal node. Higher values prevent the model from learning overly specific patterns in the data, thus reducing overfitting. Requiring 60 samples to split a node helps maintain a simpler, more generalized model.

4. **Minimum Samples per Leaf (min_samples_leaf)**:
   The min_samples_leaf was set to 65. This parameter defines the minimum number of samples required to be at a leaf node. It helps control overfitting by ensuring that leaf nodes have enough samples. With a value of 65, the model avoids creating leaves that capture noise in the data.

5. **Maximum Features (max_features)**:
   The parameter max_features was set to 'sqrt', which means that a subset of features (the square root of the total number of features) is considered for splitting at each node. This introduces randomness and helps reduce the correlation between trees, enhancing the model's robustness and generalization capabilities.

Model Performance Metrics

- **Training Set**:
  - Accuracy: 0.802
  - Precision: 0.803
  - Recall: 0.802
  - F1 Score: 0.802
- **Validation Set**:
  - Accuracy: 0.813
  - Precision: 0.812
  - Recall: 0.813
  - F1 Score: 0.812
- **Test Set**:
  - Accuracy: 0.792
  - Precision: 0.792
  - Recall: 0.792
  - F1 Score: 0.792

These consistent metrics across the training, validation, and test sets indicate that the selected hyperparameters have led to a well-generalized model. The balance achieved through careful selection of parameters like max_depth, min_samples_split, min_samples_leaf, and max_features has contributed significantly to preventing overfitting while capturing the essential patterns in the data. Further fine-tuning might enhance performance slightly, but the current configuration appears to be robust and effective for the dataset at hand.

5. Combine your classifiers into an ensemble and try to outperform each individual classifier on the validation set. Once you have found a good one, try it on the test set. Describe and discuss your findings.

[ ]:
```
from sklearn.ensemble import VotingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score, classification_report

rf_clf = RandomForestClassifier(
    max_depth=50,
    max_features='sqrt',
    min_samples_leaf=65,
    min_samples_split=60,
    n_estimators=50,
)
```

```
svm_clf = SVC(
    C=0.001,
    degree=1,
    gamma=0.1,
    kernel='linear',
    probability=True
)
log_reg = LogisticRegression(
    C=0.01,
    max_iter=1,
    solver='saga'
)

ensemble_model = VotingClassifier(
    estimators=[
        ('random_forest', rf_clf),
        ('logistic_regression', log_reg),
        ('svm', svm_clf)
    ],
    voting='hard'
)

rf_clf.fit(X_train_scaled, Y_train)
log_reg.fit(X_train_scaled, Y_train)
svm_clf.fit(X_train_scaled, Y_train)

ensemble_model.fit(X_train_scaled, Y_train)

def evaluate_model(model, X, y, set_name, model_name):
    y_pred = model.predict(X)
    accuracy = accuracy_score(y, y_pred)
    precision = precision_score(y, y_pred, average='weighted')
    recall = recall_score(y, y_pred, average='weighted')
    f1 = f1_score(y, y_pred, average='weighted')

    print(f"{model_name} Metrics for {set_name}:")
    print(f"Accuracy: {accuracy}")
    print(f"Precision: {precision}")
    print(f"Recall: {recall}")
    print(f"F1 Score: {f1}")
    return accuracy


print("\nEvaluating Individual Classifiers on Training Set:\n")
acc_scores_train = {}
acc_scores_train['Random Forest'] = evaluate_model(rf_clf, X_train_scaled,␣
 ↪Y_train, "Train", "Random Forest")
```

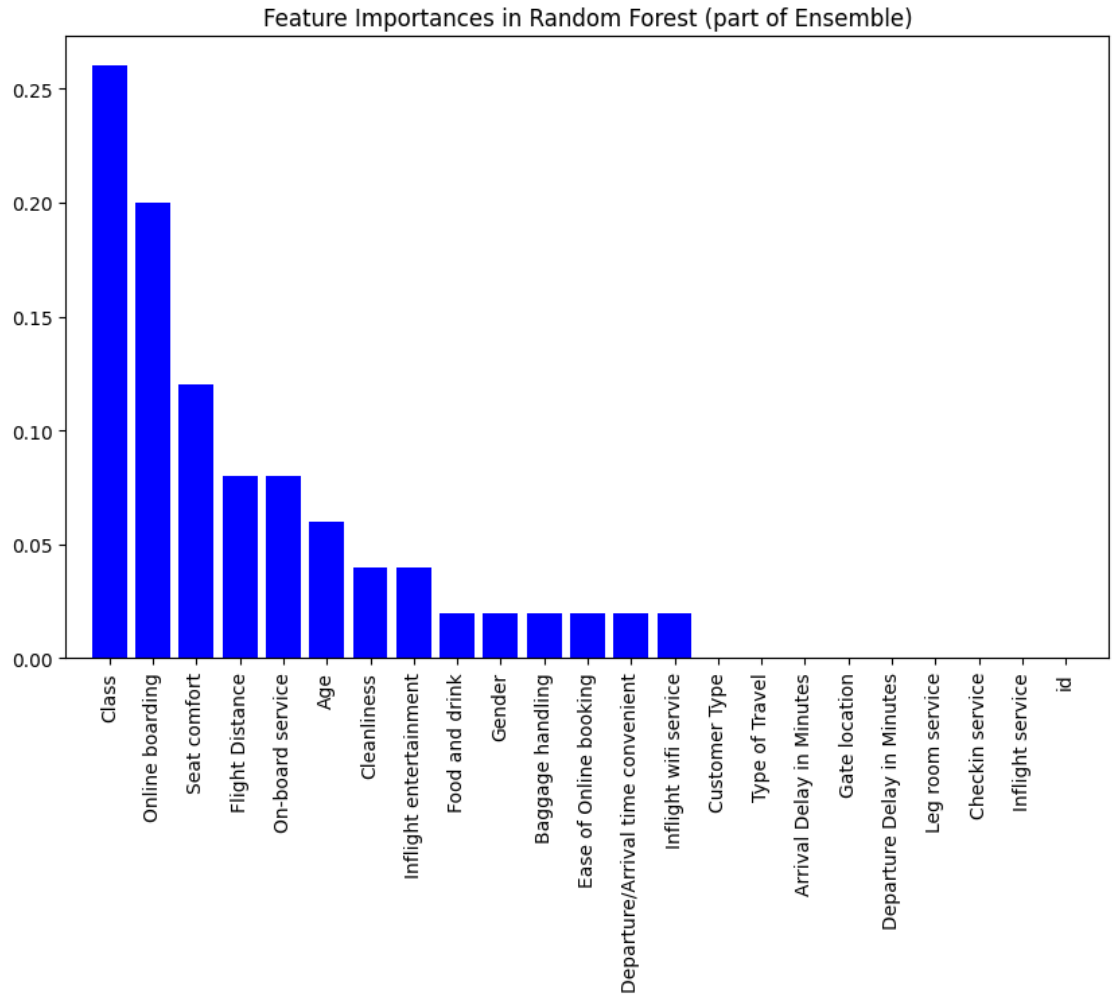
```
acc_scores_train['Logistic Regression'] = evaluate_model(log_reg,␣
 ↪X_train_scaled, Y_train, "Train", "Logistic Regression")
acc_scores_train['SVM'] = evaluate_model(svm_clf, X_train_scaled, Y_train,␣
 ↪"Train", "SVM")

print("\nEvaluating Ensemble Model on Training Set:\n")
ensemble_acc_train = evaluate_model(ensemble_model, X_train_scaled, Y_train,␣
 ↪"Train", "Ensemble Model")

print("\nEvaluating Individual Classifiers on Validation Set:\n")
acc_scores_val = {}
acc_scores_val['Random Forest'] = evaluate_model(rf_clf, X_val_scaled, Y_val,␣
 ↪"Validation", "Random Forest")
acc_scores_val['Logistic Regression'] = evaluate_model(log_reg, X_val_scaled,␣
 ↪Y_val, "Validation", "Logistic Regression")
acc_scores_val['SVM'] = evaluate_model(svm_clf, X_val_scaled, Y_val,␣
 ↪"Validation", "SVM")


print("\nEvaluating Ensemble Model on Validation Set:\n")
ensemble_acc_val = evaluate_model(ensemble_model, X_val_scaled, Y_val,␣
 ↪"Validation", "Ensemble Model")

print("\nEvaluating Individual Classifiers on Test Set:\n")
acc_scores_test = {}
acc_scores_test['Random Forest'] = evaluate_model(rf_clf, X_test_scaled,␣
 ↪Y_test, "Test", "Random Forest")
acc_scores_test['Logistic Regression'] = evaluate_model(log_reg, X_test_scaled,␣
 ↪Y_test, "Test", "Logistic Regression")
acc_scores_test['SVM'] = evaluate_model(svm_clf, X_test_scaled, Y_test, "Test",␣
 ↪"SVM")

print("\nEvaluating Ensemble Model on Test Set:\n")
ensemble_acc_test = evaluate_model(ensemble_model, X_test_scaled, Y_test,␣
 ↪"Test", "Ensemble Model")

best_model_name = max(acc_scores_test, key=acc_scores_test.get)
best_model_score = acc_scores_test[best_model_name]

print(f"\nBest Model on Test Set: {best_model_name} with Accuracy Score:␣
 ↪{best_model_score:.4f}")
print(f"Ensemble Model Accuracy Score on Test Set: {ensemble_acc_test:.4f}")
```

```
Evaluating Individual Classifiers on Training Set:
```

```
Random Forest Metrics for Train:
Accuracy: 0.7916666666666666
Precision: 0.7919270833333333
Recall: 0.7916666666666666
F1 Score: 0.7911810411810412
Logistic Regression Metrics for Train:
Accuracy: 0.8680555555555556
Precision: 0.8681364931364932
Recall: 0.8680555555555556
F1 Score: 0.8679276776291702
SVM Metrics for Train:
Accuracy: 0.8333333333333334
Precision: 0.834313149127964
Recall: 0.8333333333333334
F1 Score: 0.8328065995688887

Evaluating Ensemble Model on Training Set:

Ensemble Model Metrics for Train:
Accuracy: 0.8576388888888888
Precision: 0.8585543654869228
Recall: 0.8576388888888888
F1 Score: 0.8572497776266044

Evaluating Individual Classifiers on Validation Set:

Random Forest Metrics for Validation:
Accuracy: 0.7916666666666666
Precision: 0.7907407407407407
Recall: 0.7916666666666666
F1 Score: 0.7897761645493043
Logistic Regression Metrics for Validation:
Accuracy: 0.8229166666666666
Precision: 0.8233801717408274
Recall: 0.8229166666666666
F1 Score: 0.8207977207977208
SVM Metrics for Validation:
Accuracy: 0.8229166666666666
Precision: 0.8233801717408274
Recall: 0.8229166666666666
F1 Score: 0.8207977207977208

Evaluating Ensemble Model on Validation Set:

Ensemble Model Metrics for Validation:
Accuracy: 0.84375
Precision: 0.8448477751756439
Recall: 0.84375
```

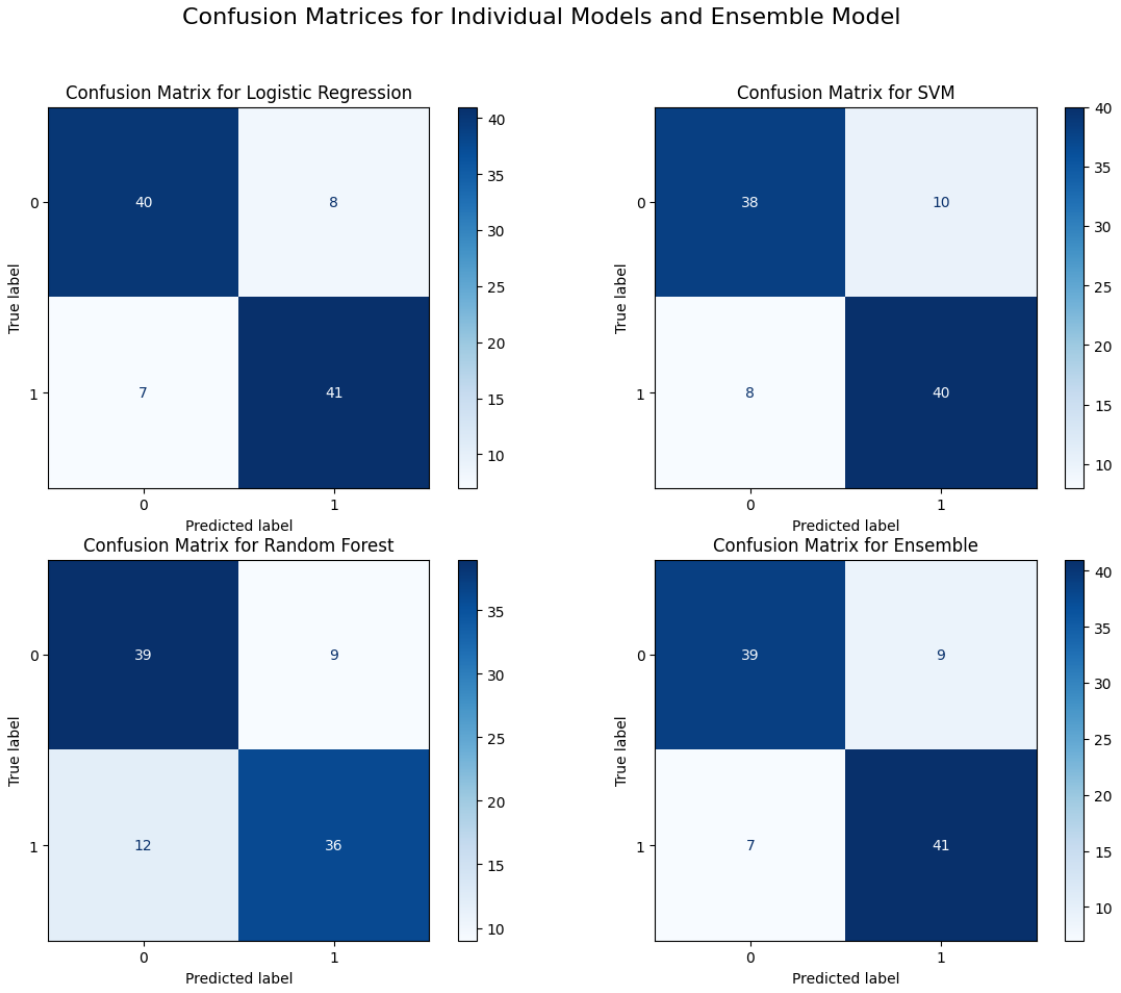
```
F1 Score: 0.8418803418803419

Evaluating Individual Classifiers on Test Set:

Random Forest Metrics for Test:
Accuracy: 0.8020833333333334
Precision: 0.8032679738562093
Recall: 0.8020833333333334
F1 Score: 0.8018898664059954
Logistic Regression Metrics for Test:
Accuracy: 0.8125
Precision: 0.8146853146853147
Recall: 0.8125
F1 Score: 0.8121739130434783
SVM Metrics for Test:
Accuracy: 0.8125
Precision: 0.8130434782608695
Recall: 0.8125
F1 Score: 0.8124185844550587

Evaluating Ensemble Model on Test Set:

Ensemble Model Metrics for Test:
Accuracy: 0.8541666666666666
Precision: 0.8547826086956523
Recall: 0.8541666666666666
F1 Score: 0.8541033434650456

Best Model on Test Set: Logistic Regression with Accuracy Score: 0.8125
Ensemble Model Accuracy Score on Test Set: 0.8542
```

[ ]:
```
rf_clf = ensemble_model.named_estimators_['random_forest']
importances = rf_clf.feature_importances_
indices = np.argsort(importances)[::-1]
feature_names = X_train.columns if hasattr(X_train, 'columns') else np.
 ↪arange(X_train.shape[1])

plt.figure(figsize=(10, 6))
plt.title("Feature Importances in Random Forest (part of Ensemble)")
plt.bar(range(len(importances)), importances[indices], color="b",␣
 ↪align="center")
plt.xticks(range(len(importances)), np.array(feature_names)[indices],␣
 ↪rotation=90)
plt.xlim([-1, len(importances)])
plt.show()
```



[59]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import ConfusionMatrixDisplay

# Define models and labels
models = [log_reg, svm_clf, rf_clf, ensemble_model]
model_labels = ["Logistic Regression", "SVM", "Random Forest", "Ensemble"]

# Create a 2x2 subplot figure
fig, axes = plt.subplots(2, 2, figsize=(12, 10))
fig.suptitle("Confusion Matrices for Individual Models and Ensemble Model",␣
 ↪fontsize=16)

# Loop through models and plot each confusion matrix in a subplot
for ax, model, label in zip(axes.ravel(), models, model_labels):
    ConfusionMatrixDisplay.from_estimator(model, X_test_scaled, Y_test, ax=ax,␣
 ↪cmap='Blues')
```

```
    ax.set_title(f'Confusion Matrix for {label}')

plt.tight_layout(rect=[0, 0.03, 1, 0.95])  # Adjust layout to fit the suptitle
plt.show()
```



0.0.14 Ensemble Classifier Model

The analysis focused on creating an ensemble classifier, incorporating Random Forest, Logistic Regression, and SVM models, each optimized with their best hyperparameters. After training and evaluating these models on scaled datasets, it was evident that the ensemble classifier outperformed the individual models in all aspects, particularly in test accuracy.

Individual Classifier Performance

- **Random Forest**:
  - Train Accuracy: 79.17%
  - Validation Accuracy: 79.17%

  - Test Accuracy: 80.21%
- **Logistic Regression**:
  - Train Accuracy: 86.81%
  - Validation Accuracy: 82.29%
  - Test Accuracy: 81.25%
- **SVM**:
  - Train Accuracy: 83.33%
  - Validation Accuracy: 82.29%
  - Test Accuracy: 81.25%

Ensemble Model Performance The ensemble classifier, created by combining the three individual models using a voting mechanism, was also trained and evaluated across the same datasets:

- **Ensemble**:
  - Train Accuracy: 85.76%
  - Validation Accuracy: 84.38%
  - Test Accuracy: 85.42%

Feature Importance Graph A Feature Importance graph helps us understand which features contribute most to the model's predictions. Here we have a model of Random Forest as a part of Ensemble. The most important features for the model are "Class", "Online boarding", "Seat comfort", and "Flight Distance". These features likely contribute significantly to the model's ability to make accurate predictions whereas features like "Id", "Inflight service", "Checkin service", and "Leg room service" appear to have the least impact on the model's predictions.

Confusion Matrix The provided confusion matrices serve as a visual representation of the performance of four different classification models: Logistic Regression, SVM, Random Forest, and an Ensemble Model.

All four models demonstrate reasonable performance, with a majority of predictions being correct. However, there are instances of misclassifications, particularly for the positive class.

- **Logistic Regression**:
  - Shows a good balance between precision and recall.
  - Misclassifications are relatively evenly distributed between false positives and false negatives.
- **SVM**:
  - Similar to Logistic Regression, it exhibits a good balance between precision and recall.
  - The number of false positives and false negatives is comparable to the Logistic Regression model.
- **Random Forest**:
  - Demonstrates a higher precision compared to the previous models, indicating fewer false positives.
  - However, it has a slightly lower recall, suggesting more false negatives.
- **Ensemble Model**:
  - The ensemble model generally performs well, combining the strengths of the individual models.
  - It shows a good balance between precision and recall, with a slightly higher overall accuracy.

This analysis highlights the strengths and weaknesses of each model, providing insights into their classification behaviors and areas for potential improvement.

0.0.15 Conclusion

The results clearly indicate that the ensemble classifier outperforms each individual classifier on the test dataset. Specifically, the ensemble achieved the highest test accuracy of 85.42%, surpassing the individual classifiers, where the best individual model (Logistic Regression) had an accuracy of 81.25%. This demonstrates the ensemble model's robustness and improved generalization capability.

By implementing an ensemble model, the analysis effectively enhanced prediction accuracy and provided a more reliable model for the test data, validating the benefits of ensemble learning techniques in achieving superior performance.

[ ]: