Project Report

Exploratory Factor Analysis

Name: Arun Govind
Course: AI and ML
(Batch 4)

- **Problem Statement**

  Use the Airline Passenger Satisfaction dataset to perform factor analysis. Choose the best features possible that helps in dimensionality reduction, without much loss in information.

- **Prerequisites**

  1. Software:
     - Python 3 (Use anaconda as your python distributor as well)

  2. Tools:
     - Numpy
     - Sklearn
     - Seaborn
     - Matplotlib
     - Factor Analyzer
     - Pingouin

  3. Dataset used: Air passenger satisfaction dataset from Kaggle

- **Method Used**

  Exploratory factor analysis or EFA is a statistical technique used to reduce data to a smaller set of summary variables and to explore the underlying structure of a relatively large set of variables.

  It is used to identify the underlying relationships between measured variables. Each observed variable is considered as a potential measure of every factor, and the goal is to determine the strongest relationships.

- **Implementation:**

1. Code for Importing all Libraries to fetch and arrange data and to perform Exploratory Data Analysis on the dataset:

In [2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from factor_analyzer import FactorAnalyzer
```

In [ ]:
```
#Exploratory Data Analysis
```

In [10]:
```
df = pd.read_csv("train.csv")
```

In [11]:
```
df.head()
```

Out[11]:

	Unnamed: 0	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	...	Inflight entertainment	On-board service	Leg room service	Baggage handling	Checkir servic
0	0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	...	5	4	3	4	4
1	1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	...	1	1	5	3	[illegible]
2	2	110028	Female	Loyal Customer	26	Business travel	Business	1142	2	2	...	5	4	3	4	4
3	3	24026	Female	Loyal Customer	25	Business travel	Business	562	2	5	...	2	2	5	3	[illegible]
4	4	119299	Male	Loyal Customer	61	Business travel	Business	214	3	3	...	3	3	4	4	3

5 rows × 25 columns

2. Code for describing our dataset:

In [14]:
```
df.describe()
```

Out[14]:

	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Seat comfort	In entertain
count	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.00
mean	39.379706	1189.448375	2.729683	3.060296	2.756901	2.976883	3.202129	3.250375	3.439396	3.35
std	15.114964	997.147281	1.327829	1.525075	1.398929	1.277621	1.329533	1.349509	1.319088	1.33
min	7.000000	31.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	27.000000	414.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.00
50%	40.000000	843.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	4.000000	4.00
75%	51.000000	1743.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	5.000000	4.00
max	85.000000	4983.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.00

We see that our dataset contains 103,904 observations. Of the 23 columns we currently have, 14 seem to be representing responses, on a scale of 1 to 5, to a survey evaluating different aspects of the flights (Inflight wifi service, food and drink, online boarding, seat comfort, etc). These 14 columns will be very important for our upcoming factor analysis.

3. Code for computing average score for each class:

In [21]:
```
eco = df[df['Class']=='Eco'][df.columns[6:20]].mean().mean()
eco_plus = df[df['Class']=='Eco Plus'][df.columns[6:20]].mean().mean()
business = df[df['Class']=='Business'][df.columns[6:20]].mean().mean()
print(eco, eco_plus, business)
```

```
3.0670277951805396 3.0686835182431653 3.4301678388057124
```

As expected, Business class was better rated. Eco class and Eco Plus practically got the same grade, which indicates that customers that paid for Eco Plus don't feel they got their money's worth.

4. Code for rating every variable:

In [22]:
```
df.groupby('Class')[df.columns[6:20]].mean()
```

Out[22]:

Class	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	Clea
Business	2.775315	2.905910	2.913964	2.982926	3.323165	3.716541	3.760858	3.635437	3.679472	3.644498	3.842907	3.519178	3.844579	3
Eco	2.675067	3.199123	2.605241	2.971954	3.086277	2.812985	3.138838	3.098256	3.120355	3.085720	3.450551	3.122002	3.463921	3
Eco Plus	2.767948	3.217507	2.661996	2.967574	3.122631	2.889245	3.183747	3.141713	3.047638	3.061382	3.363758	3.017214	3.388444	3

This gives some great hindsight into what could be improved by the company. For instance, we see that Wifi, across all classes was poorly evaluated. We also see that the online boarding was more much less convenient for customers not in the business class.

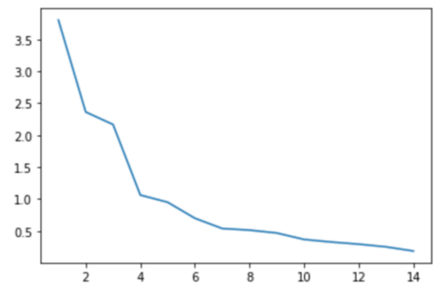
5. Code for evaluating the classes:

In [23]:
```
plt.subplot(1,2,1)
df.Class.value_counts().plot(kind='bar', figsize=(10,5))
plt.title('Observations per class')
plt.subplot(1,2,2)
df[df['satisfaction']==0].Class.value_counts().plot(kind='bar', figsize=(10,5))
plt.title('Neutral or dissatisfied per class')
```

6. Code for computing eigen values:

```
In [26]: #Factor Analysis
```

```
In [29]: #Subset of the data
         x =df[df.columns[6:20]]

         fa = FactorAnalyzer()
         fa.fit(x, 10)

         #Get Eigen values and plot
         ev, v = fa.get_eigenvalues()
         ev
         plt.plot(range(1,x.shape[1]+1),ev)

Out[29]: [<matplotlib.lines.Line2D at 0x7fb791992ee0>]
```



We will only use 3 factors here, given the big dropoff in eigenvalue after the 3rd factor. Let's see what factors are created, and what variables they contain. A loading cutoff of 0.5 will be used here.

7. Code for performing EFA:

```
In [31]: fa = FactorAnalyzer(3, rotation='varimax')
         fa.fit(x)
         loads = fa.loadings_
         print(loads)

         [[ 0.16826952  0.12827119  0.75809134]
          [-0.02950837  0.05968117  0.50138365]
          [ 0.03023106  0.02091436  0.93277525]
          [-0.0338282  -0.03231121  0.50404385]
          [ 0.75263893  0.01094635  0.00616734]
          [ 0.39545345  0.1138114   0.35906543]
          [ 0.78999048  0.08146326  0.02725824]
          [ 0.7456934   0.46674984  0.01203424]
          [ 0.09388069  0.70115382  0.02900913]
          [ 0.07445487  0.48144209  0.08065029]
          [ 0.02346305  0.76474833  0.02769279]
          [ 0.14351222  0.28418169  0.02888186]
          [ 0.01813146  0.79977083  0.01825226]
          [ 0.85842046  0.08814824 -0.00170807]]
```

Here are the 3 factors, the variables they contain and their possible "interpretability":

- Comfort: Food and Drink, Seat comfort, Inflight entertainment, Cleanliness
- Service: Onboard service, Baggage Handling, Inflight Service
- Convenience: In flight Wifi, Departure/Arrival time convenience, Online Booking, Gate Location.

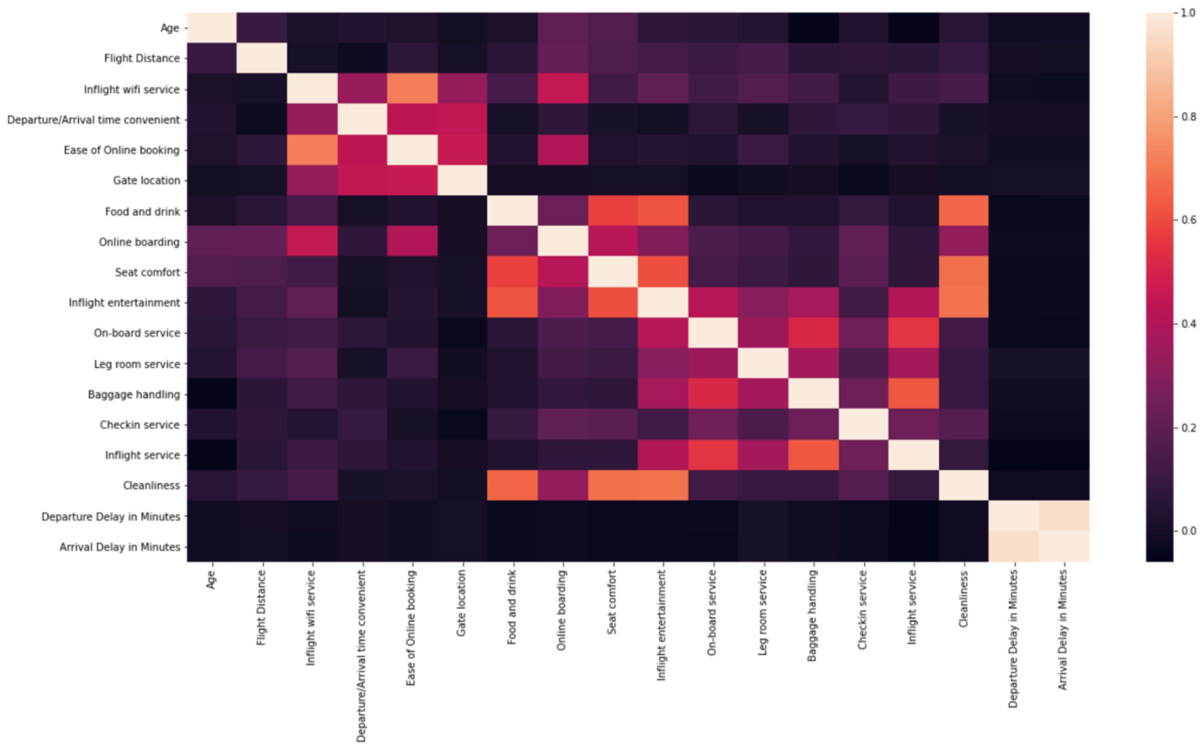
8. Code for finding the Cronbach alpha:

```
In [34]: #Create factors
         factor1 = df[['Food and drink', 'Seat comfort', 'Inflight entertainment', 'Cleanliness']]
         factor2 = df[['On-board service', 'Baggage handling', 'Inflight service']]
         factor3 = df[['Inflight wifi service', 'Departure/Arrival time convenient', 'Ease of Online booking', 'Gate location

         #Get cronbach alpha
         factor1_alpha = pg.cronbach_alpha(factor1)
         factor2_alpha = pg.cronbach_alpha(factor2)
         factor3_alpha = pg.cronbach_alpha(factor3)

         print(factor1_alpha, factor2_alpha, factor3_alpha)

         (0.8762877916624101, array([0.875, 0.878])) (0.794291693309021, array([0.792, 0.796])) (0.7679754211110685, array
         ([0.766, 0.77 ]))
```

The Cronbach alpha can be used to measure whether or not the variables of a factor form a "coherent" factor. A value above 0.6 for the alpha is in practice deemed acceptable.

- **Results:**

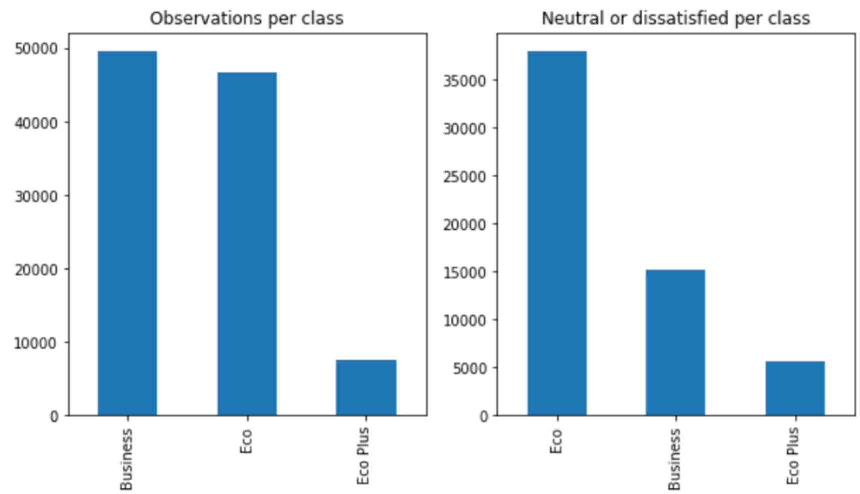
1. Heatmap of the dataset after EDA:



Some variables are quite highly correlated, especially the ones relating to what seems to be answers to a survey. However, what really stands out is the extremely high correlation (0.98) between the "Departure Delay in Minutes" and the "Arrival Delay in Minutes". That makes sense. If the plane leaves later than expected, it should arrive later as well. Considering this high correlation and the fact that we had 310 Nan in the "Arrival Delay in Minutes" column.

2. Graph for observing satisfaction per class:

```
Out[23]: Text(0.5, 1.0, 'Neutral or dissatisfied per class')
```



Results are clear. Customers in the "Eco" class are not as numerous as ones in the business class but they still had a very large chunk of unhappy customers.

```
In [24]: eco_proportion = len(df[df['Class']=='Eco'])/len(df)
         bad_proportion = len(df[df['Class']=='Eco']['satisfaction']==0)/len(df[df['satisfaction']==0])
         print(eco_proportion, bad_proportion)

0.449886433631044 0.7939163368943086
```

The "Eco" class customers accounted for about 45% of total customers, but for 79% of unhappy ones.

3. Cronbach Alpha score:

```
(0.876288, array([0.875, 0.878])) (0.794292, array([0.792, 0.796])) (0.767975, array([0.766, 0.77 ]))
```

The alphas are evaluated at 0.87, 0.79 and 0.76, which indicates that they are useful and coherent. We could use these new factors as variable for other analysis or for prediction.