Probabilistic Models Final Project

Modeling and Inferring Airline Passenger Satisfaction

In [1]:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import OneHotEncoder

%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings("ignore")

import zipfile
import os
# Plot settings
plt.rcParams['figure.figsize'] = (12, 9)
plt.rcParams['font.size'] = 12
```

In [2]:

```
#loading the data
data=pd.read_csv("train.csv").drop('Unnamed: 0', axis=1)
data_raw=data.copy()
```

In [3]:

```
data['Gender']=data['Gender'].map(dict({'Male': 0, 'Female': 1}))
data['Customer Type']=data['Customer Type'].map(dict({'disloyal Customer': 0, 'Loyal Cus
data['Type of Travel']=data['Type of Travel'].map(dict({'Personal Travel': 0, 'Business
#data['Class']=data['Class'].map(dict({'Eco Plus': 0, 'Business': 1, 'Eco': 2}))
data['satisfaction']=data['satisfaction'].map(dict({'neutral or dissatisfied': 0, 'satis
```

In [4]:

```
data
```

Out[4]:

	id	Gender	Customer Type	Age	Type of Travel	Class	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	...	ent
0	70172	0	1	13	0	Eco Plus	460	3	4	3	...	
1	5047	0	0	25	1	Business	235	3	2	3	...	
2	110028	1	1	26	1	Business	1142	2	2	2	...	
3	24026	1	1	25	1	Business	562	2	5	5	...	
4	119299	0	1	61	1	Business	214	3	3	3	...	
...	...	...	...	...	...	...	...	...	...	...	...	
103899	94171	1	0	23	1	Eco	192	2	1	2	...	
103900	73097	0	1	49	1	Business	2347	4	4	4	...	
103901	68825	0	0	30	1	Business	1995	1	1	1	...	
103902	54173	1	0	22	1	Eco	1000	1	1	1	...	
103903	62567	0	1	27	1	Business	1723	1	3	3	...	

103904 rows × 24 columns

In [5]:

```
def oheColumn(oheData, columnName):
    oneHotEnc = OneHotEncoder(dtype=int, handle_unknown='ignore')
    oheDataColumn = oneHotEnc.fit_transform(oheData[[columnName]]).toarray()

    oheData[oneHotEnc.categories_[0]] = oheDataColumn

    for catCol in oneHotEnc.categories_[0]:
        oheData.rename(columns = {catCol:columnName+'_'+catCol}, inplace = True)

    return oheData
```

In [6]:

```
data = oheColumn(data, 'Class').drop('Class', axis=1)
```

In [7]:

```
data
```

Out[7]:

	id	Gender	Customer Type	Age	Type of Travel	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	Ba ha
0	70172	0	1	13	0	460	3	4	3	1	...	
1	5047	0	0	25	1	235	3	2	3	3	...	
2	110028	1	1	26	1	1142	2	2	2	2	...	
3	24026	1	1	25	1	562	2	5	5	5	...	
4	119299	0	1	61	1	214	3	3	3	3	...	
...	...	...	...	...	...	...	...	...	...	...	...	
103899	94171	1	0	23	1	192	2	1	2	3	...	
103900	73097	0	1	49	1	2347	4	4	4	4	...	
103901	68825	0	0	30	1	1995	1	1	1	3	...	
103902	54173	1	0	22	1	1000	1	1	1	5	...	
103903	62567	0	1	27	1	1723	1	3	3	3	...	

103904 rows × 26 columns

In [8]:

```
print(data.iloc[:,0:27].isna().sum())
```

```
id                                     0
Gender                                 0
Customer Type                          0
Age                                    0
Type of Travel                         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             310
```

```
satisfaction                         0
Class_Business                       0
Class_Eco                            0
Class_Eco Plus                       0
dtype: int64
```

In [9]:
```
#Filling in nan values
data['Arrival Delay in Minutes']=data['Arrival Delay in Minutes'].fillna(0)
```

In [10]:
```
print(data.iloc[:,0:27].isna().sum())
```

```
id                                  0
Gender                              0
Customer Type                       0
Age                                 0
Type of Travel                      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
Class_Business                      0
Class_Eco                           0
Class_Eco Plus                      0
dtype: int64
```

In [11]:
```
data=data.drop_duplicates()
```

In [12]:
```
data
```

Out[12]:

	id	Gender	Customer Type	Age	Type of Travel	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	...	Ba ha
0	70172	0	1	13	0	460	3	4	3	1	...	
1	5047	0	0	25	1	235	3	2	3	3	...	
2	110028	1	1	26	1	1142	2	2	2	2	...	
3	24026	1	1	25	1	562	2	5	5	5	...	
4	119299	0	1	61	1	214	3	3	3	3	...	
...	...	...	...	...	...	...	...	...	...	...	...	
103899	94171	1	0	23	1	192	2	1	2	3	...	
103900	73097	0	1	49	1	2347	4	4	4	4	...	
103901	68825	0	0	30	1	1995	1	1	1	3	...	
103902	54173	1	0	22	1	1000	1	1	1	5	...	

103903	62567	0	1	27	1	1723	1	3	3	3	...

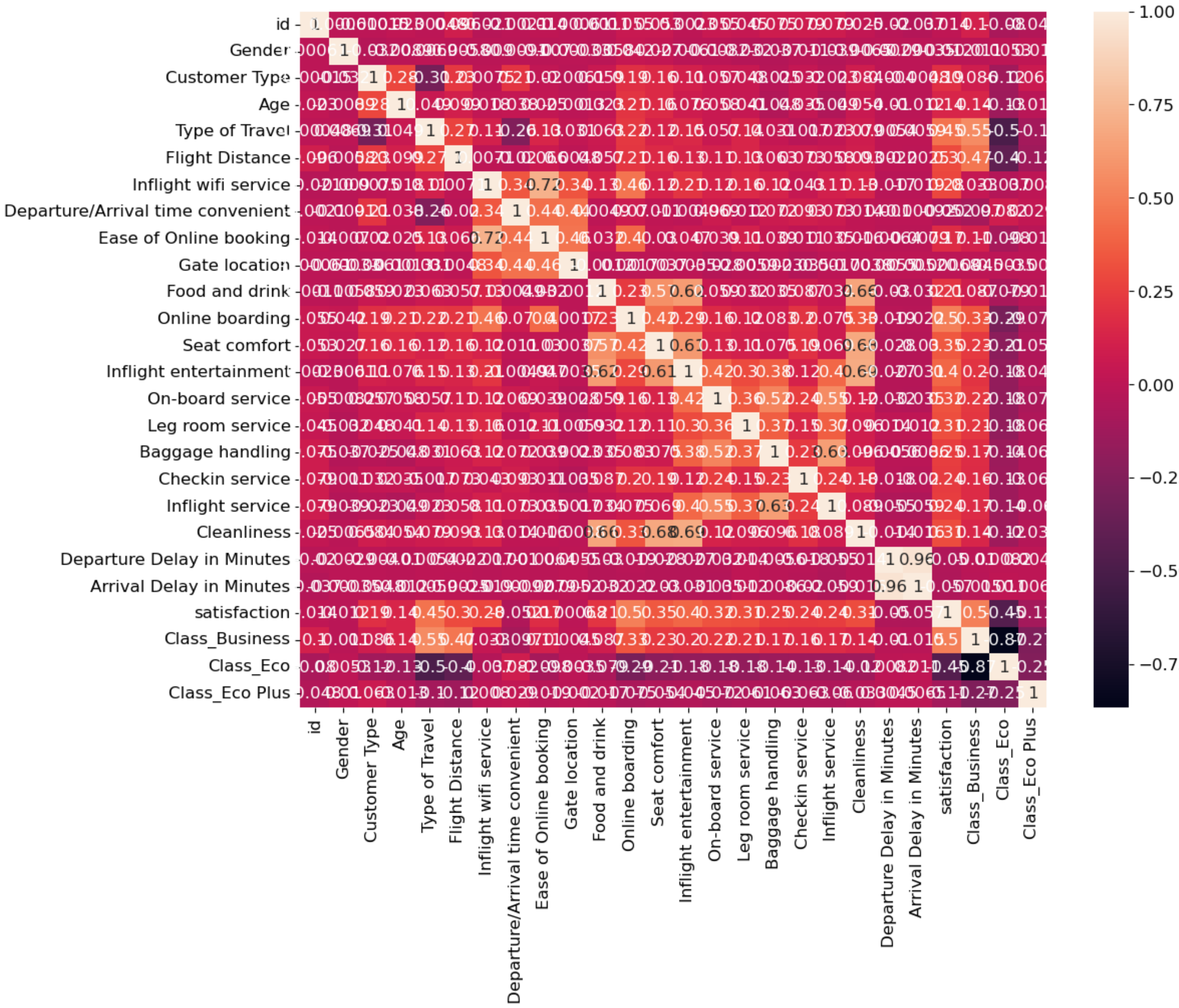
103904 rows × 26 columns

In [13]:

```
corrMatrix = data.corr()
#print(corrMatrix)
sns.heatmap(corrMatrix, annot=True,)
```

Out[13]:

```
<AxesSubplot:>
```



In [15]:

```
data1=data_raw.loc[:, ['Type of Travel', 'Inflight wifi service',
       'Ease of Online booking', 'Food and drink', 'Online boarding', 'On-board service'
       'Baggage handling', 'Inflight service', 'Cleanliness',
       'Departure Delay in Minutes', 'Arrival Delay in Minutes',
       'satisfaction', 'Class','Inflight entertainment','Seat comfort']]#data.drop("id",
```

In [16]:

```
data1
```

Out[16]:

	Type of Travel	Inflight wifi service	Ease of Online booking	Food and drink	Online boarding	On-board service	Baggage handling	Inflight service	Cleanliness	Departure Delay in Minutes	Arrival Delay in Minutes
0	Personal Travel	3	3	5	3	4	4	5	5	25	18.0
1	Business travel	3	3	1	3	1	3	4	1	1	6.0

2	Business travel	2	2	5	5	4	4	4	5	0	0.0
3	Business travel	2	5	2	2	2	3	4	2	11	9.0
4	Business travel	3	3	4	5	3	4	3	3	0	0.0
...	...	...	...	...	...	...	...	...	...	...	...
103899	Business travel	2	2	2	2	3	4	3	2	3	0.0
103900	Business travel	4	4	2	4	5	5	5	4	0	0.0
103901	Business travel	1	1	4	1	3	4	5	4	7	14.0
103902	Business travel	1	1	1	1	4	1	4	1	0	0.0
103903	Business travel	1	3	1	1	1	4	3	1	0	0.0

103904 rows × 15 columns

In [17]:

```
from pgmpy.estimators import PC
from pgmpy.estimators.CITests import chi_square
est = PC(data1)
print(est.estimate(significance_level=0.01).edges())
```

```
  0%|          | 0/5 [00:00<?, ?it/s]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\anaconda3\lib\site-packages\pgmpy\estimators\CITests.py in power_divergence(X, Y, Z, d
ata, boolean, lambda_, **kwargs)
    549                 try:
--> 550                     c, _, d, _ = stats.chi2_contingency(
    551                         df.groupby([X, Y]).size().unstack(Y, fill_value=0), lambda_=
lambda_

~\anaconda3\lib\site-packages\scipy\stats\contingency.py in chi2_contingency(observed, c
orrection, lambda_)
    268     if observed.size == 0:
--> 269         raise ValueError("No data; `observed` has size 0.")
    270

ValueError: No data; `observed` has size 0.

During handling of the above exception, another exception occurred:

TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_25332\1097962197.py in <module>
      2 from pgmpy.estimators.CITests import chi_square
      3 est = PC(data1)
----> 4 print(est.estimate(significance_level=0.01).edges())

~\anaconda3\lib\site-packages\pgmpy\estimators\PC.py in estimate(self, variant, ci_test,
 max_cond_vars, return_type, significance_level, n_jobs, show_progress, **kwargs)
    167
    168         # Step 1: Run the PC algorithm to build the skeleton and get the separat
ing sets.
--> 169         skel, separating_sets = self.build_skeleton(
    170             ci_test=ci_test,
    171                 max_cond_vars=max_cond_vars,

~\anaconda3\lib\site-packages\pgmpy\estimators\PC.py in build_skeleton(self, ci_test, max_cond_vars, significance_level, variant, n_jobs, show_progress, **kwargs)
    317                             # If a conditioning set exists remove the edge, store the
    318                             # separating set and move on to finding conditioning set for next edge.
--> 319                             if ci_test(
    320                                 u,
    321                                 v,

~\anaconda3\lib\site-packages\pgmpy\estimators\CITests.py in chi_square(X, Y, Z, data, boolean, **kwargs)
     93     False
     94     """
---> 95     return power_divergence(
     96         X=X, Y=Y, Z=Z, data=data, boolean=boolean, lambda_="pearson", **kwargs
     97     )

~\anaconda3\lib\site-packages\pgmpy\estimators\CITests.py in power_divergence(X, Y, Z, data, boolean, lambda_, **kwargs)
    561                     else:
    562                         z_str = ", ".join(
--> 563                             [f"{var}={state}" for var, state in zip(Z, z_state)]
    564                         )
    565                         logging.info(

TypeError: 'int' object is not iterable
```

In [18]:

```
from pgmpy.estimators import HillClimbSearch
from pgmpy.estimators import BDeuScore, K2Score, BicScore
from pgmpy.models import BayesianModel
hc = HillClimbSearch(data1)
best_model = hc.estimate(scoring_method=BicScore(data1))
print(best_model.edges())
```

```
  0%|          | 0/1000000 [00:00<?, ?it/s]
[('Type of Travel', 'Ease of Online booking'), ('Inflight wifi service', 'Ease of Online booking'), ('Inflight wifi service', 'satisfaction'), ('Inflight wifi service', 'Class'), ('Inflight wifi service', 'On-board service'), ('Food and drink', 'Inflight entertainment'), ('Food and drink', 'Inflight wifi service'), ('Online boarding', 'Inflight wifi service'), ('Online boarding', 'satisfaction'), ('Online boarding', 'Ease of Online booking'), ('Online boarding', 'Class'), ('On-board service', 'Inflight service'), ('On-board service', 'Baggage handling'), ('Baggage handling', 'Type of Travel'), ('Inflight service', 'Inflight entertainment'), ('Inflight service', 'Baggage handling'), ('Inflight service', 'Type of Travel'), ('Cleanliness', 'Inflight entertainment'), ('Cleanliness', 'Food and drink'), ('Cleanliness', 'Seat comfort'), ('Cleanliness', 'Online boarding'), ('satisfaction', 'Class'), ('satisfaction', 'On-board service'), ('satisfaction', 'Type of Travel'), ('satisfaction', 'Baggage handling'), ('satisfaction', 'Inflight service'), ('satisfaction', 'Ease of Online booking'), ('Class', 'Type of Travel'), ('Class', 'Inflight service'), ('Class', 'On-board service'), ('Seat comfort', 'Online boarding'), ('Seat comfort', 'Food and drink'), ('Seat comfort', 'Inflight wifi service'), ('Seat comfort', 'satisfaction')]
```

In [19]:

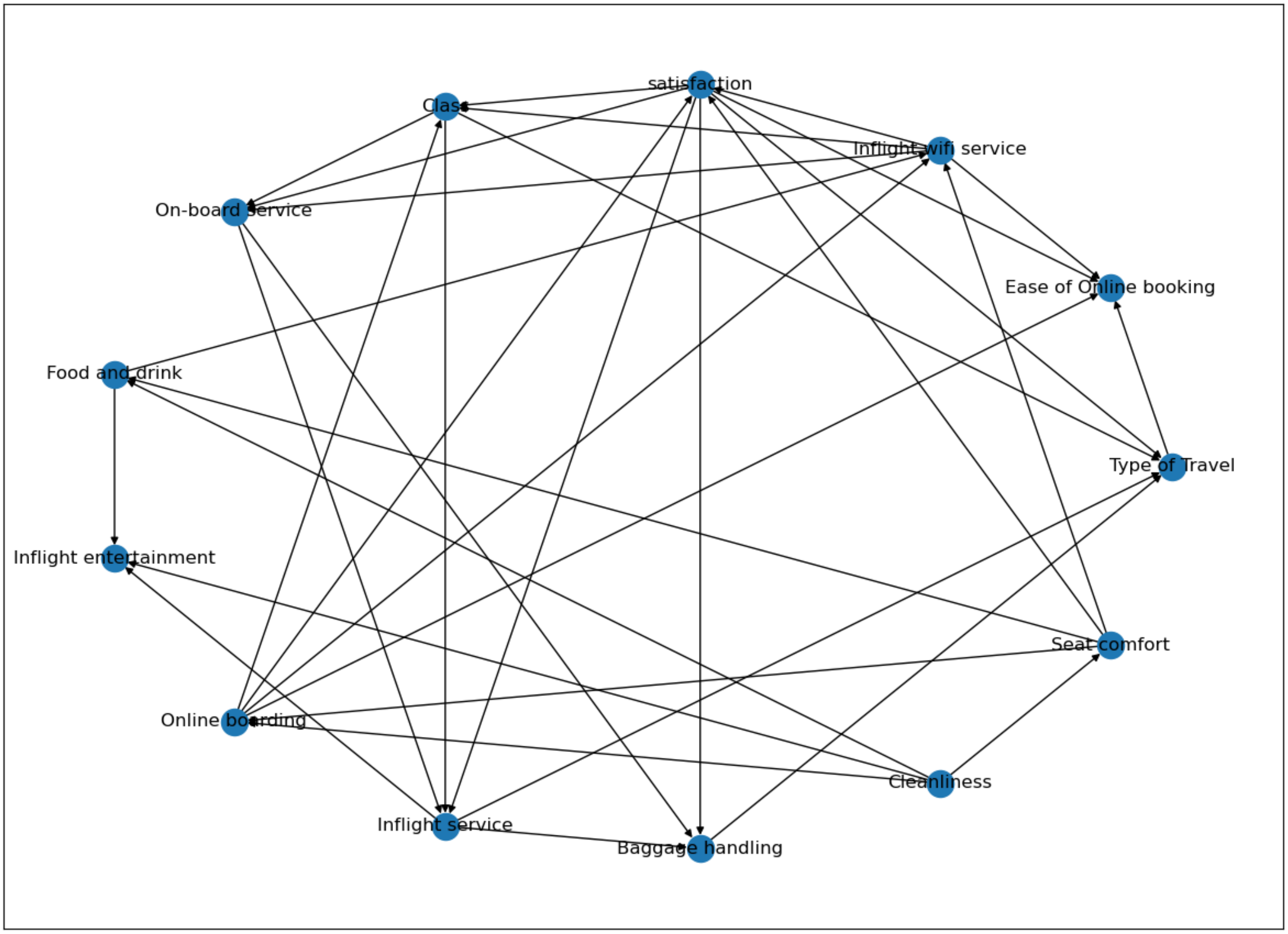
```
import networkx as nx
import matplotlib.pyplot as plt


G = nx.DiGraph(best_model.edges())

for layer, nodes in enumerate(nx.topological_generations(G)):
    # `multipartite_layout` expects the layer as a node attribute, so add the
    # numeric layer value as a node attribute
    for node in nodes:
        G.nodes[node]["layer"] = layer
```

```
# Compute the multipartite_layout using the "layer" node attribute
# pos = nx.multipartite_layout(G, subset_key="layer")
pos=nx.circular_layout(G, scale=1, center=None, dim=2)
# nx.spring_layout(G)#nx.nx_pydot.graphviz_layout(G)
#
fig, ax = plt.subplots()
nx.draw_networkx(G, pos=pos, ax=ax)
ax.set_title("DAG layout in topological order")
fig.tight_layout()
plt.show()
```

DAG layout in topological order



the features which we are inferring are\

1)Satisfaction\ 2)Type of Travel

In [20]:

```
#Maximum likelihood estimator
```

In [21]:

```
from pgmpy.models import BayesianModel

model = BayesianModel(best_model.edges())
```

In [22]:

```
from pgmpy.estimators import MaximumLikelihoodEstimator
mle = MaximumLikelihoodEstimator(model, data1)
print(mle.estimate_cpd('satisfaction'))  # unconditional
mle.estimate_cpd('satisfaction')
```

```
+----------------------------------------+-----+---------------------------+--------------
------------+
| Inflight wifi service                  | ... | Inflight wifi service(5) | Inflight wifi
service(5) |
+---------------------------------------+-----+--------------------------+--------------
-------------+
| Online boarding                       | ... | Online boarding(5)       | Online boardi
ng(5)       |
+---------------------------------------+-----+--------------------------+--------------
-------------+
| Seat comfort                          | ... | Seat comfort(4)          | Seat comfort
(5)          |
+---------------------------------------+-----+--------------------------+--------------
-------------+
| satisfaction(neutral or dissatisfied) | ... | 0.0                      | 0.00058892815
07656066    |
+---------------------------------------+-----+--------------------------+--------------
-------------+
| satisfaction(satisfied)               | ... | 1.0                      | 0.99941107184
92344       |
+---------------------------------------+-----+--------------------------+--------------
-------------+
```

Out[22]:

```
<TabularCPD representing P(satisfaction:2 | Inflight wifi service:6, Online boarding:6,
Seat comfort:6) at 0x23f15507ac0>
```

In [23]:

```
print(mle.estimate_cpd('Class'))
```

```
+-----------------------+-----+--------------------------+
| Inflight wifi service | ... | Inflight wifi service(5) |
+-----------------------+-----+--------------------------+
| Online boarding       | ... | Online boarding(5)       |
+-----------------------+-----+--------------------------+
| satisfaction          | ... | satisfaction(satisfied)  |
+-----------------------+-----+--------------------------+
| Class(Business)       | ... | 0.524873398868037        |
+-----------------------+-----+--------------------------+
| Class(Eco)            | ... | 0.3873994638069705       |
+-----------------------+-----+--------------------------+
| Class(Eco Plus)       | ... | 0.08772713732499256      |
+-----------------------+-----+--------------------------+
```

In [ ]: