In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
data_preprocessed = pd.read_csv('flight_satisfaction_preprocessed.csv')
```

In [3]:
```
data_preprocessed.head()
```

Out[3]:

| | Gender | Customer Type | Type of Travel | Business Class | Economy Class | Economy Plus Class | Flight Distance | Departure Delay | Arrival Delay |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | True | False | False | 821 | 2 | 5 |
| **1** | 1 | 1 | 0 | True | False | False | 821 | 26 | 39 |
| **2** | 0 | 1 | 0 | True | False | False | 853 | 0 | 0 |
| **3** | 0 | 1 | 0 | True | False | False | 1905 | 0 | 0 |
| **4** | 1 | 1 | 0 | True | False | False | 3470 | 0 | 1 |

5 rows × 24 columns

In [4]:
```
pd.options.display.max_rows = None
pd.options.display.max_columns = None
```

# CREATING TARGETS

In [6]:
```
targets = data_preprocessed['Satisfaction'].to_numpy()
```

In [7]:
```
targets[:300]
```

Out[7]:
```
array([0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0,
       0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
       1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0], dtype=int64)
```

In [8]:
```
targets.sum()/targets.shape
```

Out[8]:
```
array([0.42256091])
```

In [9]:
```
data_with_targets = data_preprocessed.copy()
```

# INPUTS FOR REGRESSION

In [11]:
```
unscaled_inputs = data_with_targets.iloc[:,:-1]
```

In [12]:
```
unscaled_inputs.head()
```

Out[12]:

| | Gender | Customer Type | Type of Travel | Business Class | Economy Class | Economy Plus Class | Flight Distance | Departure Delay | Arrival Delay |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | True | False | False | 821 | 2 | 5 |
| **1** | 1 | 1 | 0 | True | False | False | 821 | 26 | 39 |
| **2** | 0 | 1 | 0 | True | False | False | 853 | 0 | 0 |
| **3** | 0 | 1 | 0 | True | False | False | 1905 | 0 | 0 |
| **4** | 1 | 1 | 0 | True | False | False | 3470 | 0 | 1 |

# STANDARDIZING DATA

In [14]:
```
from sklearn.base import BaseEstimator, TransformerMixin
from sklearn.preprocessing import StandardScaler
```

In [15]:
```
class CustomScaler(BaseEstimator,TransformerMixin):


    # init or what information we need to declare a CustomScaler object

    # and what is calculated/declared as we do


    def __init__(self,columns):

        # scaler is nothing but a Standard Scaler object
```

```
        self.scaler = StandardScaler()
        # with some columns 'twist'
        self.columns = columns



    # the fit method, which, again based on StandardScale
    def fit(self, X, y=None):
        self.scaler.fit(X[self.columns], y)
        self.mean_ = np.mean(X[self.columns])
        self.var_ = np.var(X[self.columns])
        return self


    # the transform method which does the actual scaling
    def transform(self, X, y=None):
        # record the initial order of the columns
        init_col_order = X.columns


        # scale all features that you chose when creating the instance of the class
        X_scaled = pd.DataFrame(self.scaler.transform(X[self.columns]), columns=sel


        # declare a variable containing all information that was not scaled
        X_not_scaled = X.loc[:,~X.columns.isin(self.columns)]


        # return a data frame which contains all scaled features and all 'not scale
        # use the original order (that you recorded in the beginning)
        return pd.concat([X_not_scaled, X_scaled], axis=1)[init_col_order]
```

In [16]:
```
unscaled_inputs.columns.values
```

Out[16]:
```
array(['Gender', 'Customer Type', 'Type of Travel', 'Business Class',
       'Economy Class', 'Economy Plus Class', 'Flight Distance',
       'Departure Delay', 'Arrival Delay',
       'Departure and Arrival Time Convenience', 'Ease of Online Booking',
       'Check-in Service', 'Online Boarding', 'Gate Location',
       'On-board Service', 'Seat Comfort', 'Leg Room Service',
       'Cleanliness', 'Food and Drink', 'In-flight Service',
       'In-flight Wifi Service', 'In-flight Entertainment',
       'Baggage Handling'], dtype=object)
```

In [17]:
```
columns_to_omit = ['Gender', 'Customer Type', 'Type of Travel', 'Business Class',
       'Economy Class', 'Economy Plus Class']
```

In [18]:
```
columns_to_scale = [x for x in unscaled_inputs.columns.values if x not in columns_t
```

In [19]:
```
columns_to_scale
```

Out[19]:
```
['Flight Distance',
 'Departure Delay',
 'Arrival Delay',
 'Departure and Arrival Time Convenience',
 'Ease of Online Booking',
 'Check-in Service',
 'Online Boarding',
 'Gate Location',
 'On-board Service',
 'Seat Comfort',
 'Leg Room Service',
 'Cleanliness',
 'Food and Drink',
 'In-flight Service',
 'In-flight Wifi Service',
 'In-flight Entertainment',
 'Baggage Handling']
```

In [20]:
```
satisfaction_scaler = CustomScaler(columns_to_scale)
```

In [21]:
```
satisfaction_scaler.fit(unscaled_inputs)
```

```
D:\anaconda\Lib\site-packages\numpy\core\fromnumeric.py:3785: FutureWarning: The beh
avior of DataFrame.var with axis=None is deprecated, in a future version this will r
educe over both axes and return a scalar. To retain the old behavior, pass axis=0 (o
r do not pass axis)
  return var(axis=axis, dtype=dtype, out=out, ddof=ddof, **kwargs)
```

Out[21]:

```
CustomScaler(columns=['Flight Distance', 'Departure Delay', 'Arrival Dela
y',
                      'Departure and Arrival Time Convenience',
                      'Ease of Online Booking', 'Check-in Service',
                      'Online Boarding', 'Gate Location', 'On-board Servic
e',
                      'Seat Comfort', 'Leg Room Service', 'Cleanliness',
                      'Food and Drink', 'In-flight Service',
                      'In-flight Wifi Service', 'In-flight Entertainment',
```

In [22]:
```
scaled_inputs = satisfaction_scaler.transform(unscaled_inputs)
```

In [23]:
```
scaled_inputs.shape
```

Out[23]: (123878, 23)

# Training data

In [24]:
```
from sklearn.model_selection import train_test_split
```

In [25]:
```
x_train,x_test,y_train,y_test = train_test_split(scaled_inputs,targets, train_size
```

In [26]:
```
print(x_train.shape,x_test.shape)
print(y_train.shape, y_test.shape)
```

```
(99102, 23) (24776, 23)
(99102,) (24776,)
```

In [27]:
```
from sklearn.linear_model import LogisticRegression
```

In [28]:
```
from sklearn import metrics
```

In [29]:
```
reg = LogisticRegression()
```

In [30]:
```
reg.fit(x_train,y_train)
```

Out[30]:

```
LogisticRegression()
```

In [31]:
```
reg.score(x_train,y_train)
```

Out[31]: 0.8990030473653408

In [32]:
```
model_output = reg.predict(x_train)
```

In [33]:
```
(model_output == y_train).sum()
```

Out[33]: 89093

# Creating summary table

In [36]:
```
feature_name = unscaled_inputs.columns.values
```

In [37]:
```
summary_table = pd.DataFrame(columns = ['Feature Name'], data = feature_name)
```

In [38]:
```
summary_table['Coefficient'] = np.transpose(reg.coef_)
```

In [39]:
```
summary_table
```

Out[39]:

| | Feature Name | Coefficient |
|---|---|---|
| **0** | Gender | -0.050510 |
| **1** | Customer Type | 2.799021 |
| **2** | Type of Travel | -3.379182 |
| **3** | Business Class | 0.017500 |
| **4** | Economy Class | -0.818203 |
| **5** | Economy Plus Class | -1.037394 |
| **6** | Flight Distance | 0.028454 |
| **7** | Departure Delay | 0.132630 |
| **8** | Arrival Delay | -0.295444 |
| **9** | Departure and Arrival Time Convenience | -0.444186 |
| **10** | Ease of Online Booking | 0.478332 |
| **11** | Check-in Service | 0.455739 |
| **12** | Online Boarding | 1.176902 |
| **13** | Gate Location | -0.348778 |
| **14** | On-board Service | 0.435585 |
| **15** | Seat Comfort | 0.015011 |
| **16** | Leg Room Service | 0.396078 |
| **17** | Cleanliness | 0.328842 |
| **18** | Food and Drink | -0.059350 |
| **19** | In-flight Service | 0.198217 |
| **20** | In-flight Wifi Service | 0.992210 |
| **21** | In-flight Entertainment | 0.123287 |
| **22** | Baggage Handling | 0.185474 |

In [40]:
```
summary_table.index = summary_table.index + 1
```

In [42]:
```
summary_table.loc[0] = ['Intercept',reg.intercept_[0]]
```

In [43]:
```
summary_table = summary_table.sort_index()
```

In [45]:
```
summary_table['Odds_ratio'] = np.exp(summary_table.Coefficient)
```

In [46]:
```
summary_table.sort_values('Odds_ratio', ascending=False)
```

Out[46]:

| | Feature Name | Coefficient | Odds_ratio |
|---|---|---|---|
| **2** | Customer Type | 2.799021 | 16.428561 |
| **13** | Online Boarding | 1.176902 | 3.244308 |
| **21** | In-flight Wifi Service | 0.992210 | 2.697190 |
| **11** | Ease of Online Booking | 0.478332 | 1.613381 |
| **12** | Check-in Service | 0.455739 | 1.577339 |
| **15** | On-board Service | 0.435585 | 1.545867 |
| **17** | Leg Room Service | 0.396078 | 1.485986 |
| **18** | Cleanliness | 0.328842 | 1.389359 |
| **20** | In-flight Service | 0.198217 | 1.219227 |
| **23** | Baggage Handling | 0.185474 | 1.203788 |
| **8** | Departure Delay | 0.132630 | 1.141828 |
| **22** | In-flight Entertainment | 0.123287 | 1.131209 |
| **7** | Flight Distance | 0.028454 | 1.028863 |
| **4** | Business Class | 0.017500 | 1.017654 |
| **16** | Seat Comfort | 0.015011 | 1.015125 |
| **1** | Gender | -0.050510 | 0.950744 |
| **19** | Food and Drink | -0.059350 | 0.942377 |
| **9** | Arrival Delay | -0.295444 | 0.744201 |
| **14** | Gate Location | -0.348778 | 0.705550 |
| **10** | Departure and Arrival Time Convenience | -0.444186 | 0.641346 |
| **5** | Economy Class | -0.818203 | 0.441224 |
| **6** | Economy Plus Class | -1.037394 | 0.354377 |
| **0** | Intercept | -1.846844 | 0.157734 |
| **3** | Type of Travel | -3.379182 | 0.034075 |

# Testing

In [47]:
```
reg.score(x_test,y_test)
```

Out[47]:
```
0.9000645786244753
```

In [48]:
```
predicted_proba = reg.predict_proba(x_test)
```

In [49]:
```
predicted_proba
```

Out[49]:
```
array([[9.84569167e-01, 1.54308330e-02],
       [2.29370214e-02, 9.77062979e-01],
       [3.76850133e-01, 6.23149867e-01],
       ...,
       [7.16213424e-01, 2.83786576e-01],
       [9.99415552e-01, 5.84447554e-04],
       [6.57622415e-01, 3.42377585e-01]])
```

In [50]:
```
predicted_proba.shape
```

Out[50]:
```
(24776, 2)
```

In [51]:
```
predicted_proba[:,1]
```

Out[51]:
```
array([1.54308330e-02, 9.77062979e-01, 6.23149867e-01, ...,
       2.83786576e-01, 5.84447554e-04, 3.42377585e-01])
```

# Saving model and scaler

In [52]:
```
import pickle
```

In [53]:
```
with open('modell','wb') as file:
    pickle.dump(reg,file)
```

In [54]:
```
with open('scalerr','wb') as file:
    pickle.dump(satisfaction_scaler,file)
```