MACHINE LEARNING PROJECT

Flight Satisfaction

ANDREEA DIANA GHERGHESCU
XIUYUE ZHANG

Project Goal

Build a reliable classification model that can **predict** whether a passenger is satisfied or unsatisfied, based on various features captured from a post-flight survey.

Data Understanding

DATA PROFILING

Infer some information about the structure and the context of the data.

DATA DISTRIBUTION

Analysis of how values are spread or dispersed within a particular column or dataset.

DATA PROCESSING

Extract and keep only the relevant features for modeling. Make sure their values are standardized to match the models.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103904 entries, 0 to 103903
Data columns (total 25 columns):
 #   Column                             Non-Null Count   Dtype  
---  ------                             --------------   -----  
 0   Unnamed: 0                         103904 non-null  int64  
 1   id                                 103904 non-null  int64  
 2   Gender                             103904 non-null  object 
 3   Customer Type                      103904 non-null  object 
 4   Age                                103904 non-null  int64  
 5   Type of Travel                     103904 non-null  object 
 6   Class                              103904 non-null  object 
 7   Flight Distance                    103904 non-null  int64  
 8   Inflight wifi service              103904 non-null  int64  
 9   Departure/Arrival time convenient  103904 non-null  int64  
 10  Ease of Online booking             103904 non-null  int64  
 11  Gate location                      103904 non-null  int64  
 12  Food and drink                     103904 non-null  int64  
 13  Online boarding                    103904 non-null  int64  
 14  Seat comfort                       103904 non-null  int64  
 15  Inflight entertainment             103904 non-null  int64  
 16  On-board service                   103904 non-null  int64  
 17  Leg room service                   103904 non-null  int64  
 18  Baggage handling                   103904 non-null  int64  
 19  Checkin service                    103904 non-null  int64  
 20  Inflight service                   103904 non-null  int64  
 21  Cleanliness                        103904 non-null  int64  
 22  Departure Delay in Minutes         103904 non-null  int64  
 23  Arrival Delay in Minutes           103594 non-null  float64
 24  satisfaction                       103904 non-null  object 
dtypes: float64(1), int64(19), object(5)
```

Data Profiling

- We got to know the structure of the dataset, the semantics of each feature, the data type of each column, and whether it contains missing values or not. It is a critical step for data understanding.

Data distribution

```
df.describe()
```

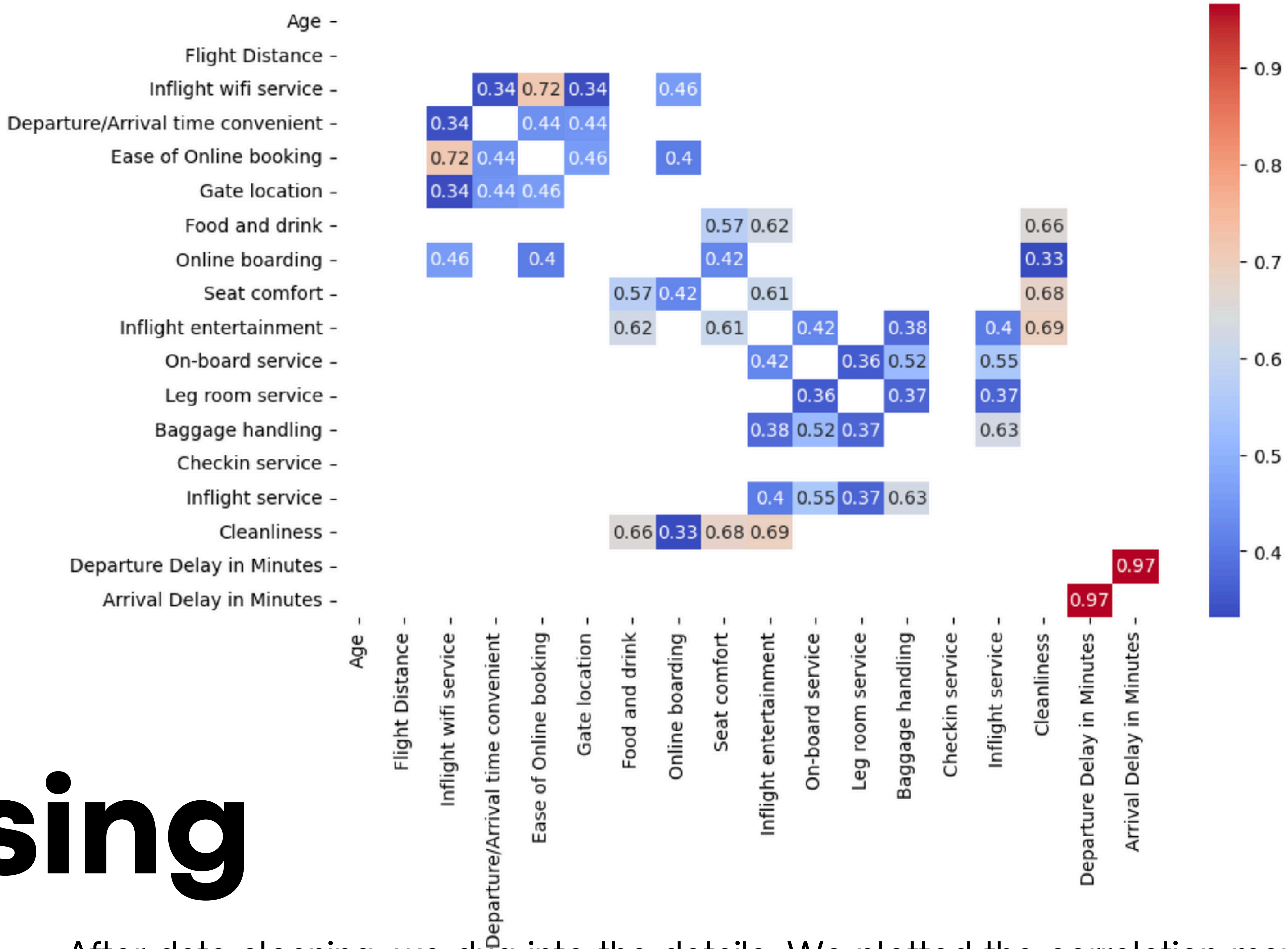
	Unnamed: 0	id	Age	Flight Distance	Inflight wifi service	Departure/Arrival time convenient	Ease of Online booking	Gate location	Food and drink	Online boarding	Seat comfort	Inflight entertainment
count	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000	103904.000000
mean	51951.500000	64924.210502	39.379706	1189.448375	2.729683	3.060296	2.756901	2.976883	3.202129	3.250375	3.439396	3.358158
std	29994.645522	37463.812252	15.114964	997.147281	1.327829	1.525075	1.398929	1.277621	1.329533	1.349509	1.319088	1.332991
min	0.000000	1.000000	7.000000	31.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	25975.750000	32533.750000	27.000000	414.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.000000
50%	51951.500000	64856.500000	40.000000	843.000000	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000	4.000000	4.000000
75%	77927.250000	97368.250000	51.000000	1743.000000	4.000000	4.000000	4.000000	4.000000	4.000000	4.000000	5.000000	4.000000
max	103903.000000	129880.000000	85.000000	4983.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000

```
df[:5]
```

	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	Checkin service	Inflight service	Cleanliness
0	0	70172	Male	Loyal Customer	13	Personal Travel	Eco Plus	460	3	4	...	5	4	3	4	4	5	5
1	1	5047	Male	disloyal Customer	25	Business travel	Business	235	3	2	...	1	1	5	3	1	4	1
2	2	110028	Female	Loyal	26	Business	Business	1142	2	2	...	5	4	3	4	4	4	5

Data Distribution.

By checking the distribution of the data, we can get some useful information (like the mean age or the maximum flight distance), so we have a better understanding of different variables.



Data Processing

After data cleaning, we dug into the details. We plotted the correlation map, from which we inferred some useful information, such as the correlation between arrival delay time and departure delay time.

Splitting Training Data and Testing Data

```
[ ] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

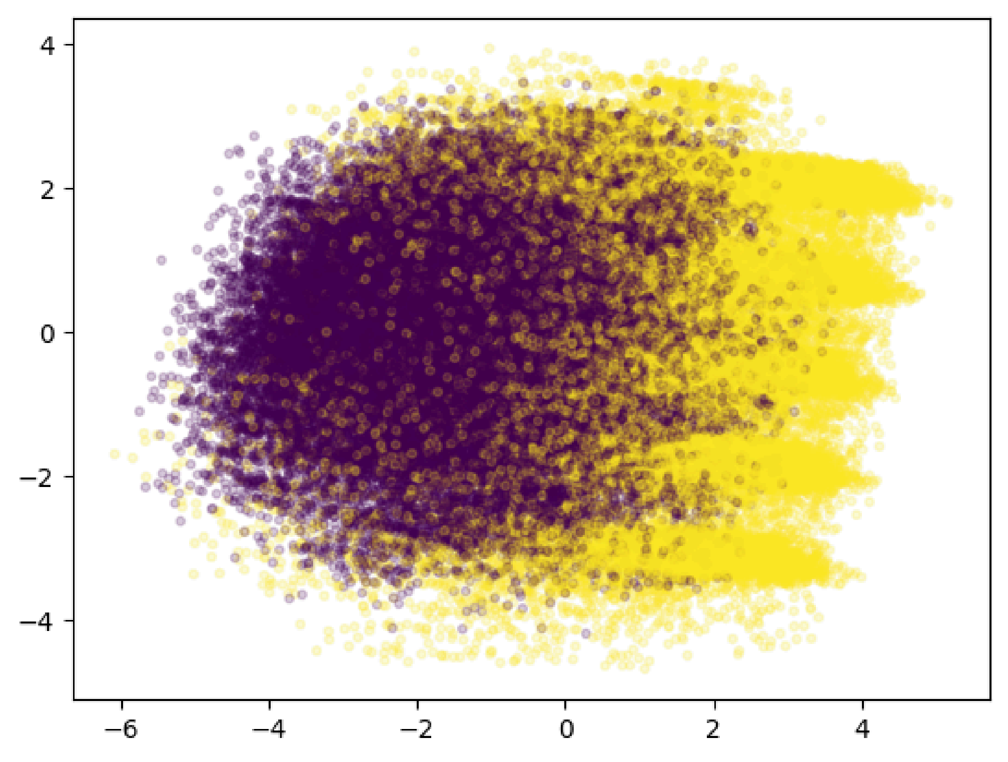
```
[ ] feature_names = X_train.columns
```

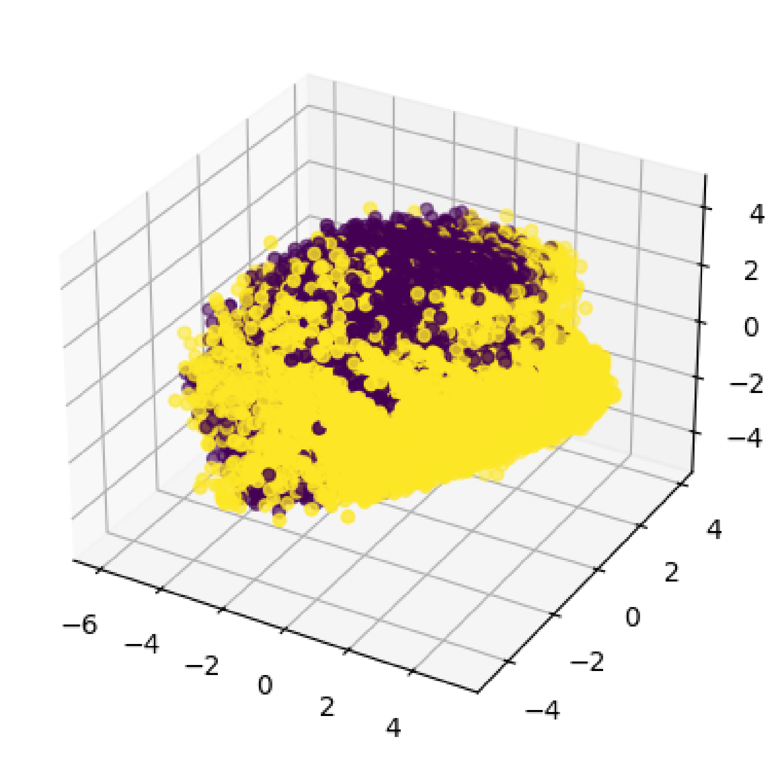
Use the StandardScaler instead of the manual normalization.

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In order to build reliable, unbiased, and generalizable machine learning models that perform well, we split the data into a training set (80%) and a testing set (20%).

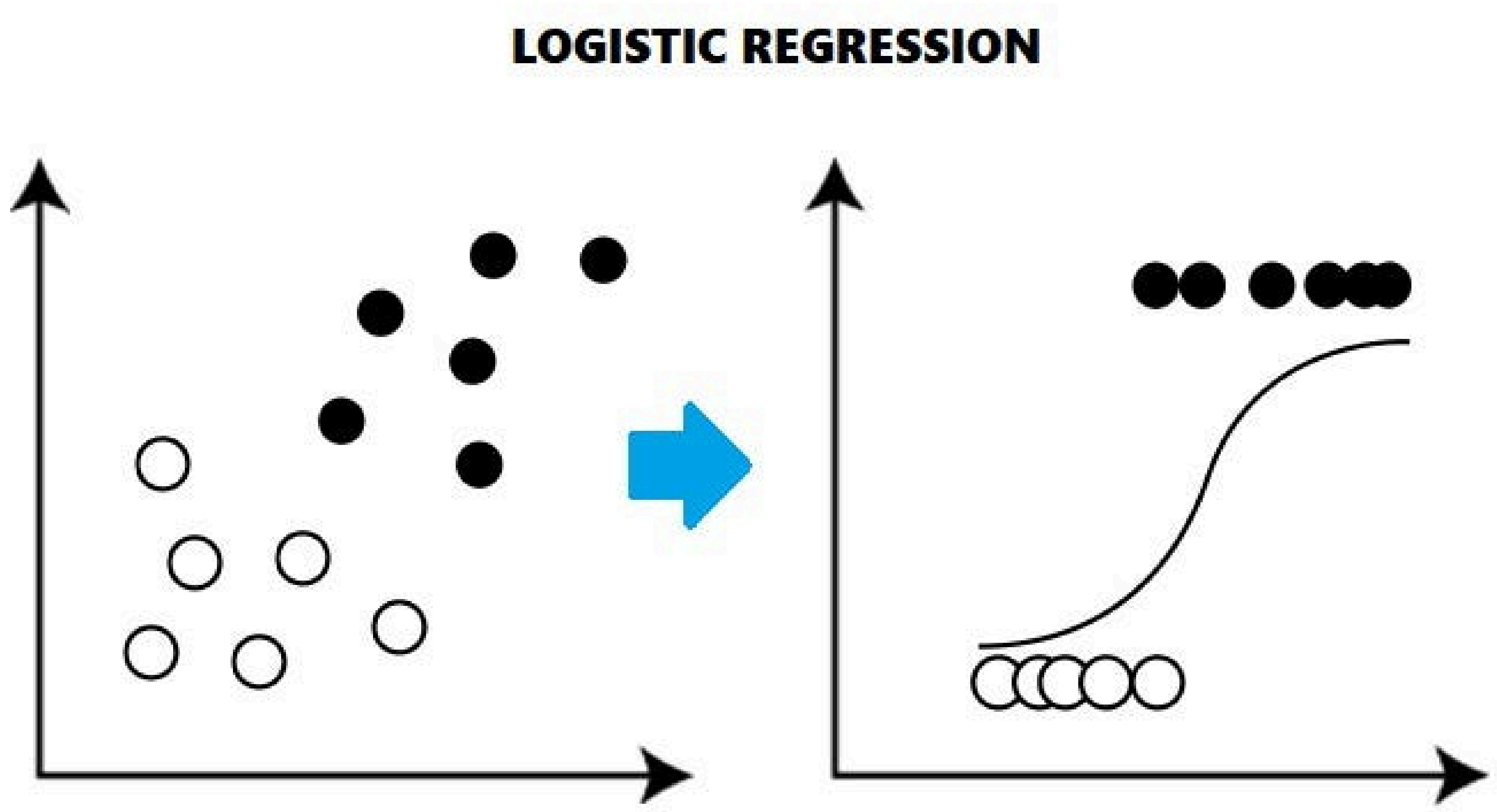
- training set - used for model learning
- testing set - used for model evaluation





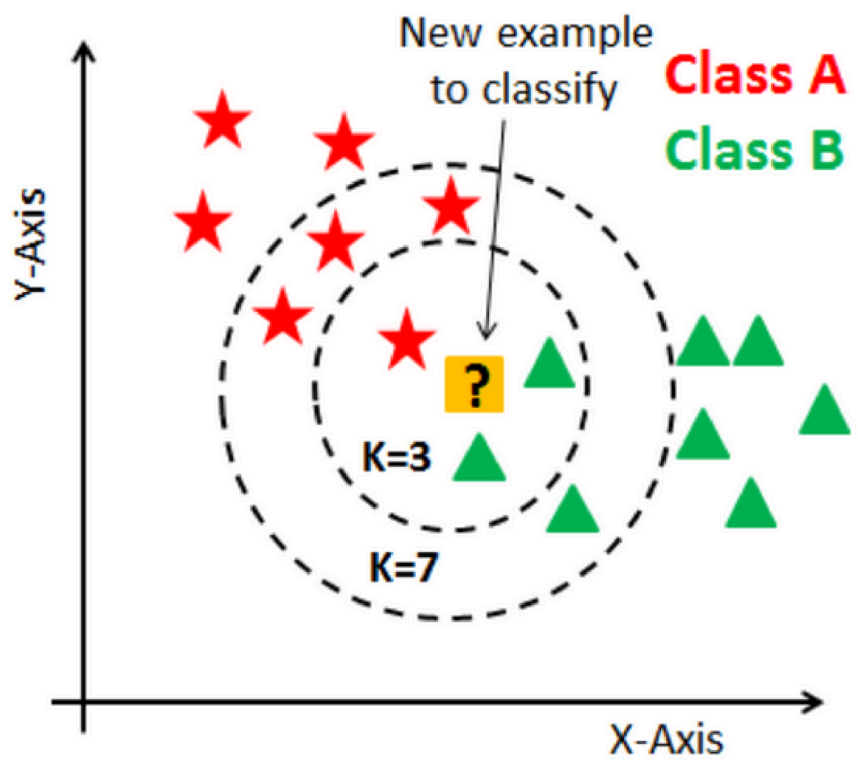
Data Vizualization in 2D & 3D

We attempted to visualize the data in space, both in 2D and 3D, using **Principal Component Analysis** to infer some information about how the data is distributed across the principal features. Unfortunately, due to the high dimensionality of the original data, no relevant information was obtained.



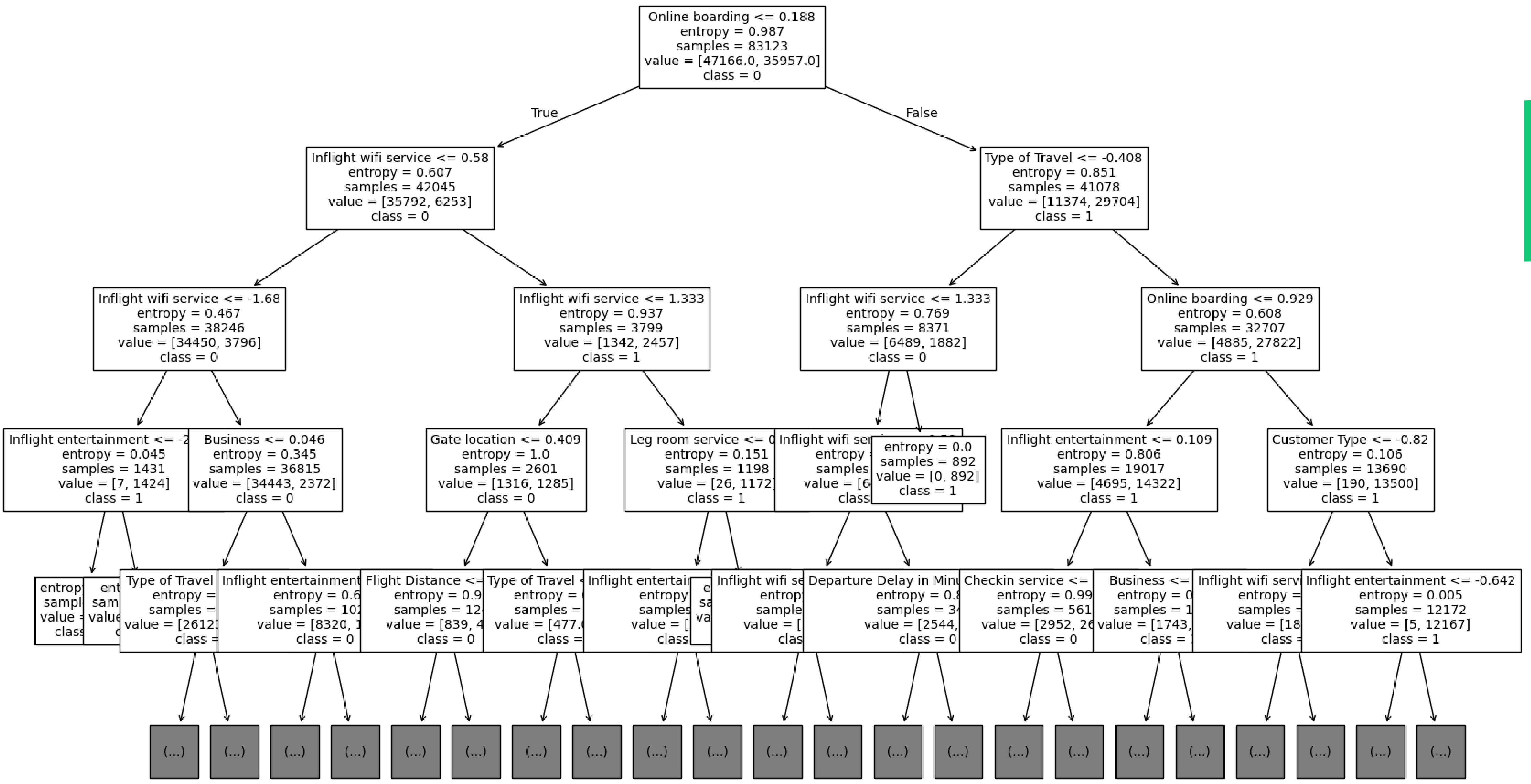
Logistic Regression

The first model we tried estimates the probability that a given input belongs to a particular class. It uses the logistic (**sigmoid**) function to map predicted values to probabilities between 0 and 1. The accuracy obtained is **0.87**, making it the worst-performing of our models.



K-Nearest Neighbors

KNN assigns a label to a data point based on the **majority label of its k closest neighbors** in the feature space.
To get a larger coverage over the space of the k parameter, we used a **randomized search**, which concluded that the best configuration for **k is 13**, with an **accuracy of 0.92.**



Decision Tree.

The Decision Tree performed better than expected, with an **accuracy value of 0.949**, outperforming both the Regressor and the KNN.

Online boarding	0.201621
Inflight wifi service	0.148053
Business	0.086439
Type of Travel	0.085017
Eco	0.061371
Inflight entertainment	0.055681
Seat comfort	0.039635
Ease of Online booking	0.038306
Leg room service	0.035475
Customer Type	0.033994
On-board service	0.029957
Cleanliness	0.025998
Flight Distance	0.023840
Baggage handling	0.023500
Age	0.023253
Checkin service	0.019488
Inflight service	0.018332
Departure/Arrival time convenient	0.012619
Gate location	0.011489
Food and drink	0.010446
Departure Delay in Minutes	0.008176
Eco Plus	0.003960
Gender	0.003349

Random Forest

```
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [5, 10, 15],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}
```

The Random Forest Classifier builds multiple Decision Trees and combines their outputs to make more accurate and stable predictions. Each tree is trained on a **random subset of the data** and considers **a random subset of features** when splitting nodes, which helps *reduce overfitting and improves generalization*. The final prediction is made by majority voting among the trees.

First accuracy obtained is **0.959**.

```
Optimal number of features: 18
```

```
param_grid = {
  'n_estimators': [50, 100, 150, 200],
  'max_depth': [5, 10, 15, 20, 25],
  'min_samples_split': [2, 5],
  'min_samples_leaf': [1, 2, 4]
}
```

Accuracy: `0.9629950435493961`

Random Forest

We used **Recursive Feature Elimination** to recursively remove features and evaluate the model's performance using cross-validation at each step. The goal is to reduce the number of features while maintaining or improving the model's performance.

After obtaining the new set of features, we proceeded to retrain the forest with the best parameters, as well as search through a larger grid.

SVC works by finding the hyperplane that best separates the data into different classes with the largest margin, being effective in high-dimensional spaces.
We applied a randomized search and obtained a **0.958 accuracy score**, which is better than how the Decision Tree performed, but it is worse than all the Random Forests we tried.

Support Vector Classifier

```python
automl_settings = {
  "time_budget": 360,  # in seconds
  "metric": "accuracy",
  "task": "classification",
}
```

LGBMClassifier

```
LGBMClassifier(colsample_bytree=np.float64(0.6578347391758362),
               learning_rate=np.float64(0.04179074535827166), max_bin=1023,
               min_child_samples=4, n_estimators=589, n_jobs=-1, num_leaves=82,
               reg_alpha=np.float64(0.007704104902643929),
               reg_lambda=np.float64(0.020229013206102948), verbose=-1)
```

AutoML

Accuracy: 0.964

Multi-Layer Perceptron Classifier

```
mlp = MLPClassifier(
  hidden_layer_sizes=(128, 64, 32),
  activation='relu',
  solver='adam',
  alpha=0.01,
  learning_rate='adaptive',
  max_iter=500,
  random_state=42,
  verbose=True
)
```

→ 3 layer architecture
→ introduce non-linearity
→ moderate level of regularization
→ lr adjusts based on performance

Accuracy: 0.951

MLPC consists of one or more hidden layers and learns complex patterns by adjusting weights through backpropagation.

```
param_grid = {
  'hidden_layer_sizes': [(256, 128, 64, 32), (128, 64, 32)],
  'activation': ['relu', 'tanh'],
  'alpha': [0.0001, 0.001, 0.01],
  'solver': ['adam'],
  'learning_rate': ['adaptive'],
  'early_stopping': [True]
}
```

Accuracy: 0.958

Light Gradient Boosting Machine Classifier is a fast, efficient gradient boosting framework based on decision trees. Given that it was the model suggested by AutoML, we tried improving it through a grid search.

```
param_grid = {
  'n_estimators': [250, 300],
  'max_depth': [5, 10, 15, -1],
  'learning_rate': [0.01, 0.1],
  'num_leaves': [31, 41],
  'min_child_samples': [5, 10, 15],
  'reg_alpha': [0, 0.1],
  'reg_lambda': [0, 0.1],
}
```

Accuracy: **0.964**, same as the one obtained by AutoML, and the **best** we got out of all models.

Improving LGBM

Thank You

Andreea Diana Gherghescu
Xiuyue Zhang