

**CMP6202**  
**AI and Machine Learning Project**  
**2023–2024**

Individual Report

**Hotel booking cancelation  
prediction with XGBClassifier**

Artificial Intelligence and Machine Learning  
Thu Ha Ngo  
24104128

## Table of Contents

1	Report Introduction.....	3
1.1	Motivations .....	3
1.2	Dataset identification .....	3
1.3	Supervised learning task identification .....	4
1.4	Team Identification .....	5
2	Exploratory Data Analysis.....	5
2.1	Question(s) identification.....	5
2.2	Splitting the dataset.....	6
2.3	Exploratory Data Analysis process and results.....	6
2.3.1	Overview of dataset .....	6
2.3.2	Univariate analysis - target variable .....	7
2.3.3	Univariate analysis – categorical features.....	7
2.3.4	Univariate analysis – continuous values .....	8
2.3.5	Multivariate analysis .....	8
2.3.6	Correlation analysis .....	10
2.4	EDA conclusions .....	12
3	Experimental Design .....	12
3.1	Identification of your chosen supervised learning algorithm(s) .....	12
	The chosen algorithm is XGBClassifier in the open-source software library.....	12
	XGBoost, which stands for Extreme Gradient Boosting. ....	12
3.2	Identification of appropriate evaluation techniques.....	13
3.3	Data cleaning and Pre-processing transformations.....	14
3.3.1	Drop outliers .....	14
3.3.2	Feature engineering .....	15
3.3.3	Impute missing values.....	16
3.3.4	handling imbalanced dataset.....	17
3.3.6	Feature Scaling .....	18
3.4	Limitations and Options.....	20
4	Predictive Modelling / Model Development .....	19
4.1	The predictive modelling process .....	19
4.2	Evaluation results on “seen” data .....	19
5	Evaluation and further modelling improvements .....	20
5.1	Initial evaluation comparison.....	20
5.2	Research team comparison .....	20
5.3	Further modelling improvements attempted .....	21
5.4	Fine-tune the model .....	23

5.5	Final Evaluation results.....	24
6	Conclusion .....	25
6.1	Summary of results .....	25
6.2	Suggested further improvements to the model development process.....	25
6.3	Reflection on Research Team .....	25
6.4	Reflection on Individual Learning .....	26
7	References .....	26

# 1 Report Introduction

This report presents a complete machine learning pipeline which includes dataset identification, exploratory data analysis (EDA), pre-processing, modelling, training, evaluation and tuning of a Hotel booking dataset. The problem solved is predicting whether a booking will be cancelled or not based on booking data.

## 1.1 Motivations

Predicting hotel booking cancellation can serve the following purposes:

- **Maximize revenue through OverBook:** During peak season, it is ideal to have all rooms occupied. Unfortunately, cancellations happen for various reasons. If the hotel know which bookings are highly likely cancelled, they can allow additional bookings for these rooms (Dong and Ling, 2015, p. 11696).
- **Customer retention and satisfaction:** Knowing customers are going to cancel, the hotel can proactively reach out to them for additional support/offers (such as room upgrade or dining vouchers) to encourage them keep their bookings (Sim et al, 2006, p. 1). **Feature importance** will be calculated later to help hotels better customize their offers (example, if the booking is cancelled and the "Number of Children" feature is of high importance, the hotel can offer babysit services or airport pickup with children car seats)
- **Operational efficiency and resource allocation:** The hotel can better schedule staff and manage inventory (fresh ingredients) according to number of actual reservations. For example, if there are chances that 5 bookings will be cancelled on a day, the hotel can put one staff in on-call mode (in case the customers show up) to save cost. Furthermore, if a booking is likely to be cancelled, other rooms will be prioritized for cleaning first.

## 1.2 Dataset identification

The dataset was found on Kaggle – a data science competition platform and online community of data scientists and machine learning practitioners under Google LLC ('Kaggle', 2023). The dataset represents 36285 hotel bookings collected from real-

world scenarios - stated by the author of the dataset Youssef Aboelwafa (2023) without specifying the identities.

```
df.shape
(36285, 17)
```

Figure 1: the dataset has 36285 rows (records) and 17 columns (attributes)

The dataset was chosen due to its moderate size, which is suitable for the scope of this AI and Machine Learning project. Furthermore, the dataset shows high usability score of 10.00 (highest usability score on Kaggle) with substantial number of views and downloads - 7910 and 1896 accordingly ('Hotel Booking Cancellation Prediction', 2023). The following table displays the list of attributes in the dataset, detailing name, data type and brief description for each.

Name	Data type	Brief description
Booking_ID	numerical	Unique identifier for each booking
number of adults	numerical	Number of adults included in the booking
number of children	numerical	Number of children included in the booking
number of weekend nights	numerical	Number of weekend nights included in the booking
type of meal	categorical	type of meal included in the booking
car parking space	numerical	Indicates whether a car parking space was requested or included in the booking
room type	categorical	Type of room booked
lead time	numerical	Number of days between the booking date and the arrival date
market segment type	categorical	Type of market segment associated with the booking
repeated	numerical	Indicates whether the booking is a repeat booking
P-C	numerical	Number of previous bookings that were cancelled by the customer prior to the current booking
P-not-C	numerical	Number of previous bookings not cancelled by the customer prior to the current booking
average price	numerical	Average price associated with the booking
special requests	numerical	Number of special requests made by the guest
date of reservation	string	Date of the reservation
booking status	categorical	Status of the booking (cancelled or not cancelled)

Table 1 List of attributes identified in the hotel booking cancellation prediction dataset

### 1.3 Supervised learning task identification

Because the target attribute (booking status) is of type categorical, the predictive task for this project is classification (Brownlee, 2019). Further investigation shows that there are two values in the booking status column, which narrows the classification tasks to binary classification (Sarker, 2021).

In machine learning, the term “ground truth” refers to labelled outcome that a model is training and testing itself against (Towards data science, 2018). The target variable – booking status has been labelled and will be used as ground truth for this project.

## 1.4 Team Identification

Forename	Surname	Student ID	Model(s) developed
Thu Ha	Ngo	24104128	XGBClassifier
Mai Uyen Nhi	Do	24104122	Random Forest Classifier
Quynh Duyen	Nguyen	24104132	KNN Classifier
Tuan Nam	Tran	24104140	CatBoostClassifier

Table 2: Team member details and their models

# 2 Exploratory Data Analysis

## 2.1 Question(s) identification

The questions about dataset will be divided into two main groups: data exploration and domain-specific (hotel booking).

Data exploration
1. What is the distribution of variables in the dataset? Are they normally distributed, skewed, or have other patterns?
2. Are there missing values in the dataset?
3. Are there any outliers in the dataset?
4. Which variables are likely to be most important in predicting whether a booking is cancelled or not?
5. Is the dataset imbalanced?
6. How many unique values of each categorical attribute?
7. Do I have to perform feature engineering for any attribute?
8. Which attributes have highest correlation so I can reduce number of features.

Table 3: data exploration questions

Domain-specific
1. How does number of adults affect booking status? Single, couple, or group are most likely to cancel their booking?
2. Are bookings with children more likely to be cancelled?
3. Are repeated booking is more reliable (less cancellation)?
4. Are booking with more special requests less likely to be cancelled?
5. Which market segment types are least likely to cancel bookings?
6. How does lead time affect booking status? Is it true that longer lead time means more chances of cancelling?
7. How does average prices relate to lead time? The shorter lead time, the higher price?
8. Does the average prices affect booking status? Which category, higher prices or lower prices, is more likely to lead to booking cancellations?
9. Which room type is most likely to be cancelled?

Table 4: Domain-specific questions

## 2.2 Splitting the dataset

Generalization is an important factor when evaluating a computational model. To avoid poor generalization by over-training, it is a common practice to hold out data for cross validation (Reitermanova, 2010). Furthermore, a dataset should be divided into “seen” and “unseen” subsets to avoid “data snooping bias” (Géron, 2022, p. 106). Realizing the need for cross validation and bias avoidance, the research team has decided to split the data into train and test sets before performing exploratory data analysis. Many practitioners use the 80:20 split, which is justified from Pareto principle, but it is just a rule of thumb (Joseph, 2022). The research team agreed on the 9:1 ratio, which means 90% of the dataset was used for training and 10% was reserved for testing, with considerations about dataset characteristics: moderate sized (more than 30000 records) and low number of features (17).

Data splitting was done with sklearn train\_test\_split function, which performs random splitting. The random\_state parameter controls data shuffling so the team set a same random\_state of 11 to ensure the data is randomised in same manner for each. The test\_size parameter determines how many percent of the dataset is provisioned for test set, which is set to 0.1 (10%) as mention earlier. Thanks to these protocols, the team has same subsets of data for model development and evaluation.

```
X_train, X_test, y_train, y_test = train_test_split(
    X, target, test_size=0.1, random_state=11)
```

Figure 2 Splitting data set into train and test subsets

## 2.3 Exploratory Data Analysis process and results

After splitting data, EDA was performed on train test only to avoid data leakage and bias made by human such as selecting a particular model because of some interesting pattern they saw from the test set (Géron, 2022).

Based on questions defined in section 2.1, data was visualized in different formats to explore and recognize insights. Seaborn, a python data visualization library based on matplotlib (seaborn, n.d), was used for this task.

### 2.3.1 Overview of dataset

	number of adults	number of children	number of weekend nights	number of week nights	car parking space	lead time	repeated	P-C	P-not-C	average price	special requests
count	36185.000000	36185.000000	36285.000000	36285.000000	36285.000000	36285.000000	36285.000000	36285.000000	36285.000000	36285.000000	36285.000000
mean	1.844742	0.105292	0.810693	2.204602	0.030977	85.239851	0.025630	0.023343	0.153369	103.421636	0.619733
std	0.518926	0.402665	0.870590	1.410946	0.173258	85.938796	0.158032	0.368281	1.753931	35.086469	0.786262
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	2.000000	0.000000	0.000000	1.000000	0.000000	17.000000	0.000000	0.000000	0.000000	80.300000	0.000000
50%	2.000000	0.000000	1.000000	2.000000	0.000000	57.000000	0.000000	0.000000	0.000000	99.450000	0.000000
75%	2.000000	0.000000	2.000000	3.000000	0.000000	126.000000	0.000000	0.000000	0.000000	120.000000	1.000000
max	4.000000	10.000000	7.000000	17.000000	1.000000	443.000000	1.000000	13.000000	58.000000	540.000000	5.000000

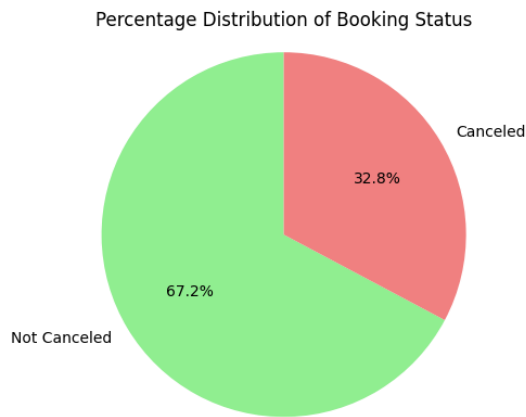
Figure 3: Summary of the central tendency, dispersion, and shape of the distribution of numerical data

From figure 3, some conclusions can be drawn as follows:

- 75% of bookings are for 2 adults and 0 child.
- 75% for 2 weekend nights and 3 weeknights

- 75% doesn't require car parking
- Most booking are not repeated therefore the columns about P-C and P-not-C (previous bookings cancelled or not cancelled by the customers)
- 50% doesn't have any special request

### 2.3.2 Univariate analysis - target variable



```
df['booking status'].value_counts()
Not_Canceled    24396
Canceled         11889
Name: booking status, dtype: int64
```

Figure 5: number of records per class

Figure 4: percentage distribution of target variable

From figure 4 and figure 5, the problem of imbalanced dataset was recognised. Imbalanced dataset means one class has a lot less instances than the other(s) or vice versa (Kotsiantis et al, 2006). In this case, the number of “Cancelled” bookings does not even make up half of its counterpart even though it is often a greater concern for hotel management. Kotsiantis et al (2006) also emphasised that imbalanced dataset can cause classifiers to ignore the minor class which is often more useful. Two solutions for this issues was suggested by Kotsiantis: using algorithms which are capable of dealing with imbalanced data or pre-processing data to make data balanced. In this project, although XGBClassifier model allowed setting “scale\_pos\_wight” parameters to tune algorithm behaviour for imbalanced dataset, the pre-processing approach was preferred due to its higher performance (Filho, 2023). Further details will be discussed in section 3.3 Data cleaning and Pre-processing transformations.

### 2.3.3 Univariate analysis – categorical features

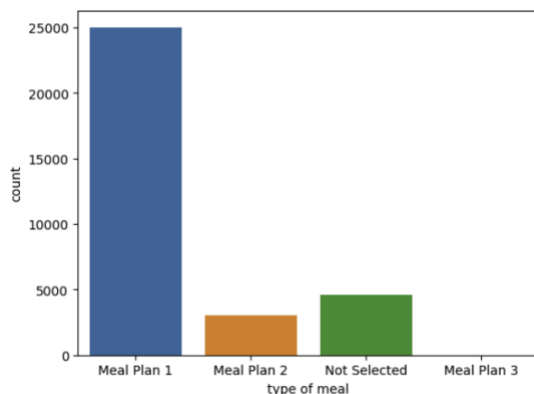


Figure 6: “type of meal” feature - data distribution

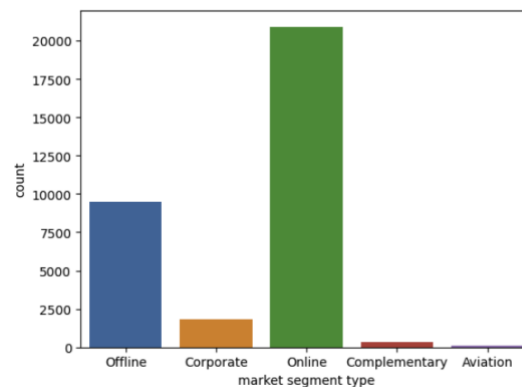


Figure 7: “market segment type” feature – data distribution

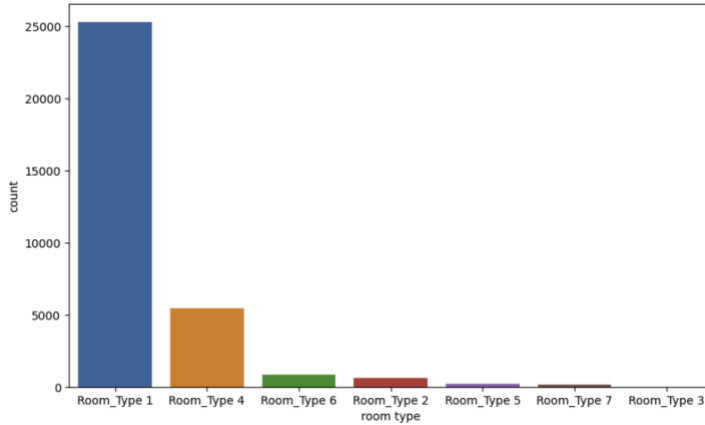


Figure 8: “room type” feature – data distribution

Figures 6 shows that most bookings has “meal plan 1” selected, “meal 3” is chosen by a tiny number of bookings and is considered as an outlier which will be removed later in pre-processing step. Figure 7 illustrates a highest number of bookings reserved “online” and the lowest number is “aviation”. Figure 8 represents “room type 1” as dominant option and “room type 3” is probably an outlier.

#### 2.3.4 Univariate analysis – continuous values

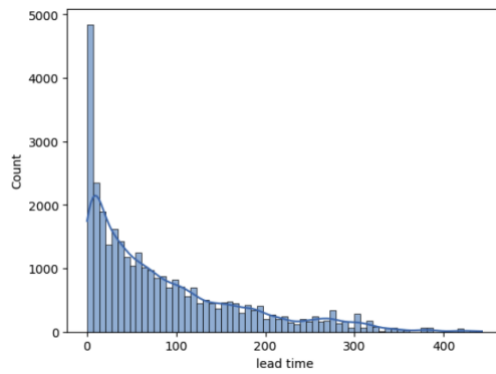


Figure 9: “lead time” attribute – data distribution

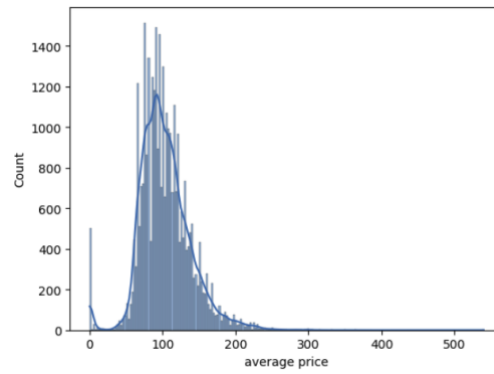


Figure 10: “average price” attribute – data distribution

The data distribution of “lead time” attribute in figure 9 shows a right skewed pattern, where most of data is on the lower end (lead time < 100). Cheryl (2020) noted that skewed data can degrade the model’s ability to generalize because it learnt from rare cases of extreme values. But she also emphasised that it was only the case for regression-based models but not tree-based models such as XGBClassifier in this project.

Figure 10 represents a normal distribution of “average price” attribute where data was distributed symmetrically and characterized by its mean and standard deviation.

#### 2.3.5 Multivariate analysis



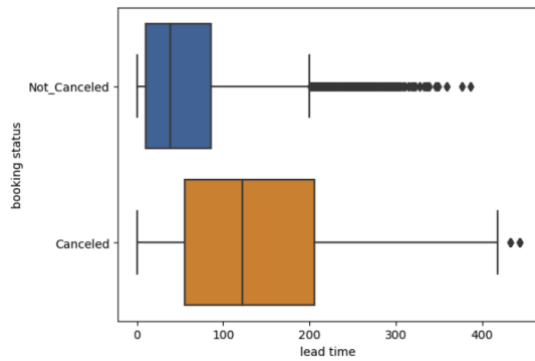


Figure 11: Boxplots of lead time and booking status

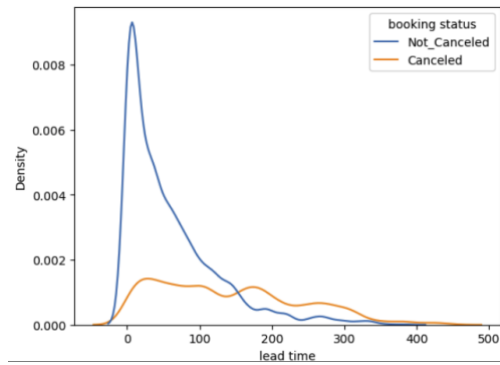


Figure 12: line graph of lead time and booking status

Figure 11 suggests that “Not\_cancelled” bookings has lower lead time ,approximately from 20-80 days, compared to “Cancelled” bookings which has average lead time ranging from 50-200 days. Figure 12 shows that a much higher number of bookings that had lower lead time and was not cancelled. This answers a question defined in section 2.1 about the relationship between lead time and booking status – the longer the lead time, the more likely that a booking will be cancelled.

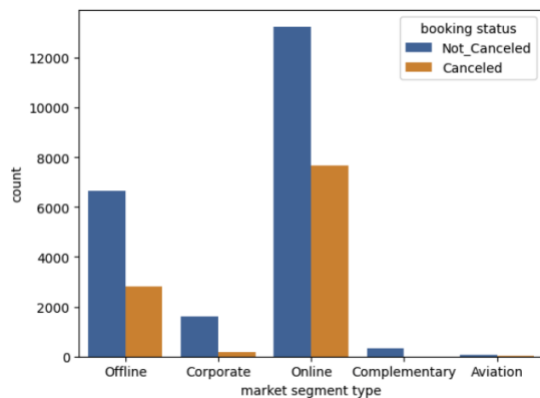


Figure 13: bar chart of market segment types and booking status

Figure 13 proves that even though “corporate”, “complementary”, and “aviation” only make up a small portion of all bookings, the chances of cancellation are very small for these types. The “offline” and “online” segments have much higher share but also have higher rate of cancellation.

Almost all of repeated bookings will not be cancelled while a third of non-repeated bookings have a chance of cancellation (figure 14).

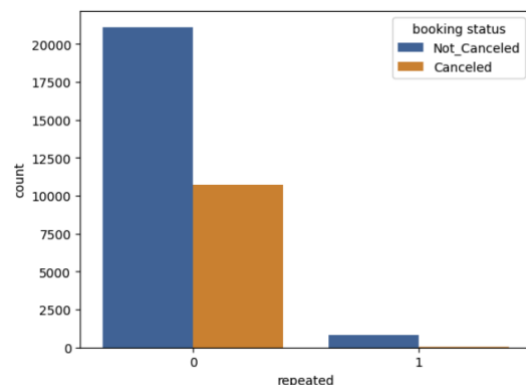


Figure 14: bar chart of “repeated” attribute and booking status

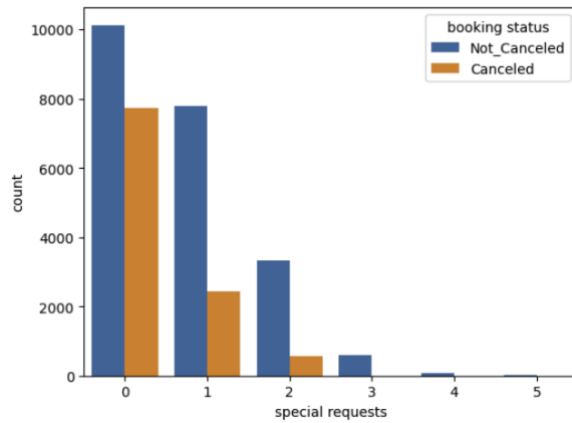


Figure 15: bar chart of ‘special requests’ attribute and booking status

There are certain months that have higher average price, which are consider peak season (from May to September). It is interesting to note that “Cancelled” bookings always (in every month) have higher average price than their counterpart (figure 16).

Bookings with 3 or more special request are guaranteed to be kept. Generally, number of special request has negative correlation with booking cancellation (more request, less chances of cancellation).

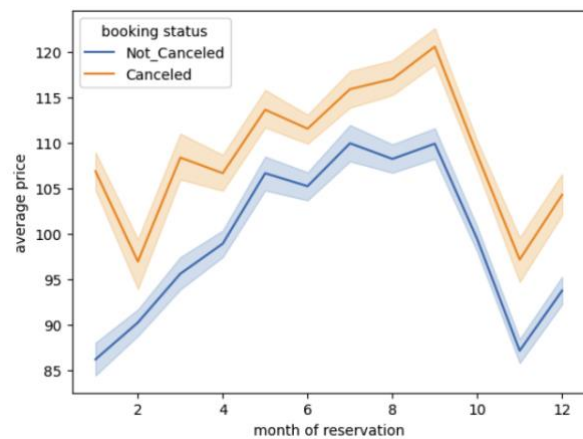


Figure 16: line graph about “month of reservation”, “average price” and booking status

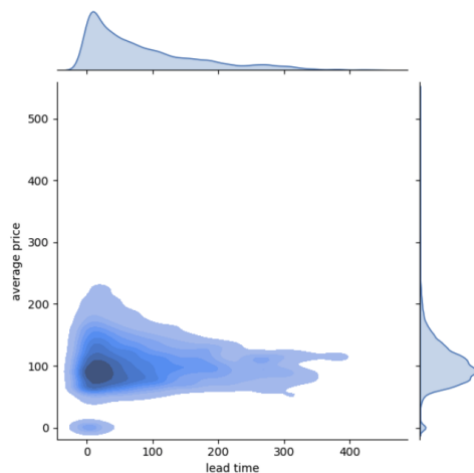


Figure 17: pair plot of “average price” and “lead time”

Most bookings have average price of \$80 - \$100 and lead time of 10 – 40 days.

### 2.3.6 Correlation analysis

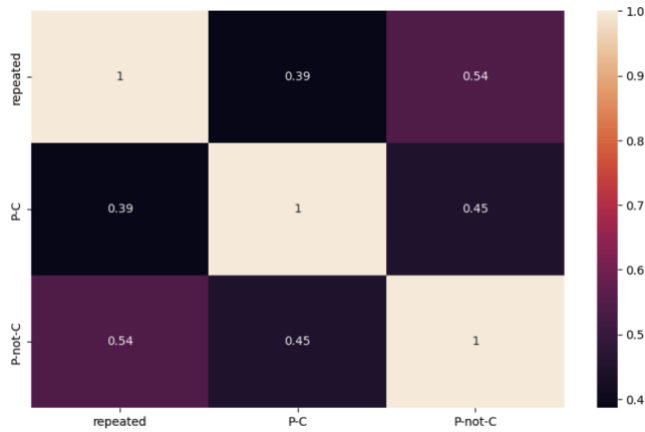


Figure 18: heat map to show correlation of “P-C”, “P-not-C”, and “repeated”

High correlations between these features suggest feature reduction to reduce computational cost (Géron, 2022, p.116).

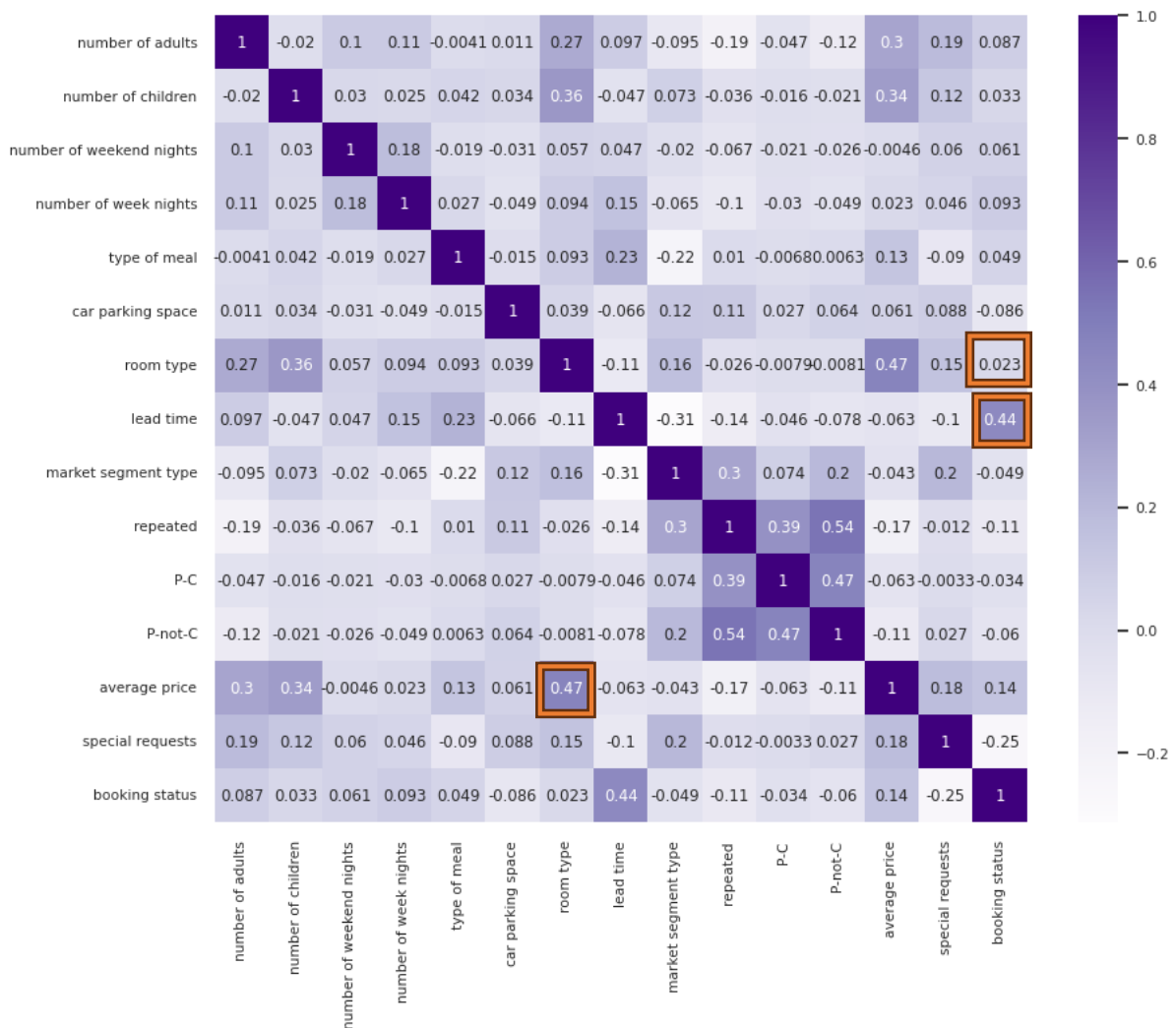


Figure 19: heat map to show correlation of features and target

Looking at the heat map, lead time has the highest correlation with booking status (score of 0.44). It is suggested that “lead time” is the most important feature in

predicting hotel booking cancellation while “room type” has least influence on booking status with the correlation score of only 0.023.

Other interesting correlation between features can be seen such as “room type” and “average price” has high correlation, which answered a domain-specific question defined in section 2.1.

## 2.4 EDA conclusions

Performing Exploratory data analysis allows the author to obtain valuable insights about the dataset and domain-specific business rules. Overall, the dataset has a moderate size of 36285 rows and 17 columns, which well represent necessary features for the defined classification problem. Investigating target values bring the issue of imbalanced dataset to light, which will be further processed before model training to yield better result. Univariate data analysis shows data distributions of both categorical and numerical attributes, in which most features has uneven data distribution and outliers. While skewed data does not affect the classification model, outliers may reduce the model’s performance and should be removed before model training. Multivariate data analysis exposes many interesting relationships between features and target as well as features and features. Most notable relationship is between “lead time” and booking status, which suggests longer lead time means more chances of booking cancellation. Analysing correlation scores between features and target confirms that “lead time” is the most important feature in hotel booking cancellation prediction.

## 3 Experimental Design

### 3.1 Identification of your chosen supervised learning algorithm(s)

The chosen algorithm is XGBClassifier in the open-source software library XGBoost, which stands for Extreme Gradient Boosting.

XGBoost implements gradient boosted decision trees to achieve better speed and performance, which has gained a good reputation in many machine learning competitions (Brownlee, 2021). A gradient boosted decision trees (GBDT) is a decision tree ensemble learning algorithm, which is comparable to random forest. XGBoost builds trees in parallel instead of sequentially like GBDT so it allows scalability and pushes the computing limitations (Nvidia, n.d).

XGBoost was chosen for this project because of the following reasons:

- High accuracy: XGBoost is well known for its high accuracy, which is achieved by combining multiple decision trees (Mandal, 2023). Therefore, it is suitable for a high level of precision required task like the hotel booking cancellation prediction in this project.
- Speed: XGBoost is designed to perform in a timely manner, even for large datasets (Mandal, 2023). Considering the size of the hotel booking cancellation prediction dataset, xgboost is chosen to provide a quick and efficient solution.

- Good reputation: XGBoost has helped many competitors win in the world of data science, such as Kaggle competition or ML contests. Seeing its proven potential makes the author want to experiment the algorithm on a practical dataset as hotel booking cancelation prediction.

Beside XGBoost's good reputation, the algorithm also poses challenges:

- Complexity: high accuracy and scalable algorithm comes with a cost of complexity that requires a certain level of understanding to work with xgboost. There are a large number of hyperparameters involved in xgboost algorithm, which is time-consuming to tune and configure.
- Overfitting: because of its complexity, the algorithm is more likely to be overfitted. The author has experienced the issue in the first training round, further details will be discussed later.
- Memory usage: XGBoost can be memory-intensive, which lead to long training time or even crash for memory limited computers.

In conclusion, XGBClassifier was chosen for this project due to its high accuracy, speed, and good reputation. The limitations of complexity, overfitting, and memory usage can easily be addressed by time dedication to understand the algorithm, feature selection, and a powerful computer.

## 3.2 Identification of appropriate evaluation techniques

To compare model performance in a fair manner, the team has agreed to follow the same evaluation protocol.

First, the team used sklearn train\_test\_split function with same test\_size and random\_state to split the dataset, which yielded same train and test subsets for each member (discussed in section 2.2).

Second, StratifiedKFold from sklearn library was used to cross-validate after training the model. K-fold cross-validation means splitting the training set into k folds (in this case, k = 5), then training the model k times, reserving a different fold each time for evaluation (Géron, 2022, p.185). The team set same parameters for StratifiedKFold function: n\_splits=5, random\_state=11, shuffle=True). The result is more reliable than evaluating on a "validation" set only because the algorithm is trained and evaluated multiple times on different data (Brownlee, 2020).

The metrics used to evaluate model performance are F1 score and ROC AUC (Receiver Operating Characteristic Area Under the Curve). The accuracy score does not well reflect model performance when dataset is skewed (Géron, 2022, p.185), so it is not chosen for this project. When working with classification problem, confusion matrix gives more information about model performance such as the number of times instances of class A are classified as class B or vice versa. However, to make comparison between models easier, a single metric - F1 score is used. F1 score is the harmonic mean of precision (accuracy of positive predictions) and recall (ratio of positive instances that are correctly detected by the classifier). Harmonic mean is different to regular mean; it does not consider all values equally but gives much more weight to low values. Hence, a classification model will only achieve a high F1 score when both precision and recall are high

(Géron, 2022, p.190). A good balance of precision and recall is especially important in this project because both type of errors (false positive – a not-cancelled booking is categorized as cancelled results in extra costs to rearrange or compensate for show-up customers and false negative – a cancelled booking is classified as not-cancelled causes revenue loss) has significant consequences for hotel management. As for ROC AUC, it is a common metric for evaluating classifier (especially binary classifier) because it can represent the degree of separability (Narkhede, 2018). Additionally, the imbalanced dataset in this project makes ROC AUC more suitable than accuracy score to demonstrate model's ability in differentiating between the classes.

Finally, prediction is made on test set to compare each member model performance in terms of F1 score.

### 3.3 Data cleaning and Pre-processing transformations

Data pre-processing is an important step in predictive model development. Raw data often contains missing values, data in different formats (such as datetime or categorical) and other inconsistencies that a machine learning model cannot process. Furthermore, skewed and/or noisy data can decrease model's performance (Singh et al, 2021). The author has applied some techniques to pre-process raw data from Hotel Booking cancellation prediction dataset before feeding to xgbclassifier.

#### 3.3.1 Drop outliers

To detect outliers, box plots were created for attributes of type numerical.

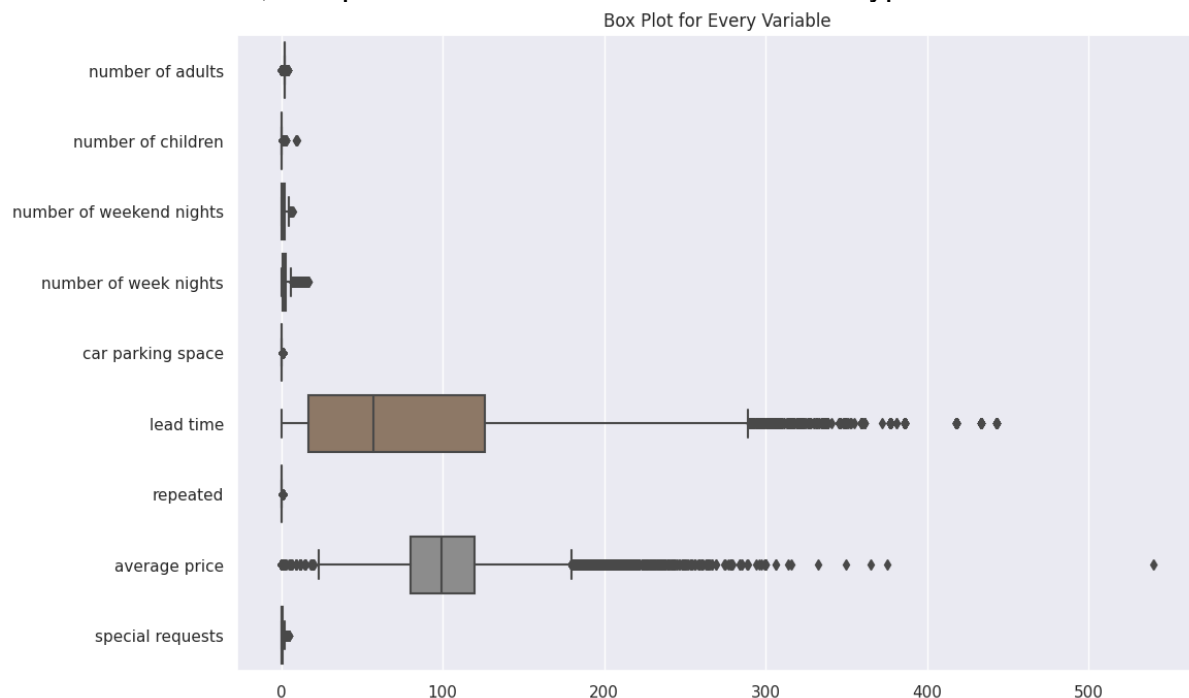


Figure 20: box plots to detect outliers

Looking at figure 20, outliers can be seen on “lead time” and “average price” features. The author then wrote python code with conditions to remove outliers. Lower bound was calculated as first quartile (q1) minus 1.5 times the interquartile

range (IQR) and Upper bound was calculated as third quartile (q3) plus 1.5 times the IQR.

```
outliers_cols = ["lead time", "average price"]
for column in outliers_cols:
    if booking[column].dtype in ["int64", "float64"]:
        q1 = booking[column].quantile(0.25)
        q3 = booking[column].quantile(0.75)
        iqr = q3 - q1
        lower_bound = q1 - 1.5 * iqr
        upper_bound = q3 + 1.5 * iqr
        booking = booking[
            (booking[column] >= lower_bound) & (booking[column] <= upper_bound)
        ]
```

Figure 21: Python code to remove outliers in “lead time” and “average price” attributes

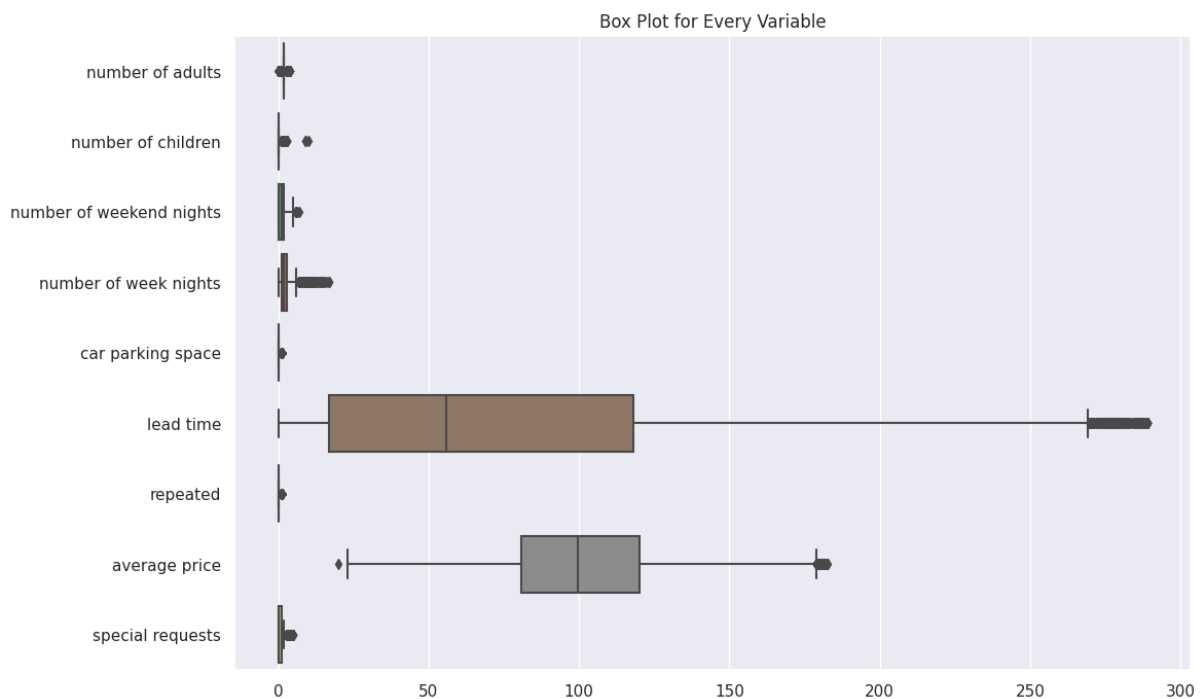


Figure 22: Box plots of attributes after removing outliers

### 3.3.2 Feature engineering

Feature engineering is a process that transform data to create new variables that is of better use for model development (Patel, 2024). During EDA stage, the author recognised “date of reservation” is in date time format that is not accepted by the chosen algorithm – xgbclassifier. Furthermore, the day and year information in “date of reservation” are not very helpful in business problem of hotel booking management. The month of reservation is more relevant in this case, which can provide information about peak season and how it affects booking status. Therefore, the “date of reservation” was transformed to represent the month attribute in the integer format (int64).

```

# Split the date to (day/month/year) & Drop the date in wrong format
df = df[~df['date of reservation'].str.contains("-")]
df['date of reservation'] = pd.to_datetime(df['date of reservation'])

# Keep the month attribute
df["month"] = df['date of reservation'].dt.month

# Remove the 'date of reservation' column
df = df.drop(columns=['date of reservation'])
df.info()

```

Figure 23: feature engineering to create month of reservation attribute

### 3.3.3 Impute missing values

the train set was checked to see whether it contains any records with missing values. Figure 24 shows that there are 73 rows having null values for attribute “number of adults” and 70 rows having null values for attribute “number of children”.

```

missing_values = X_train.isnull().sum()
missing_values

```

number of adults	73
number of children	70
number of weekend nights	0
number of week nights	0
type of meal	0
car parking space	0
room type	0
lead time	0
market segment type	0
repeated	0
average price	0
special requests	0
month	0
dtype: int64	

Figure 24: missing values in train set

Géron (2022, p.123) suggested three options to deal with missing values: remove the instances that have missing values, remove the columns that have missing values, or insert some value into the missing values. Géron also recommend the least destructive option – filling in missing values. The possible value to be filled is a constant number, the mean, the median, the mode, and so on. In this case, the rounded mean will be calculated. Unlike the median or the mode, the mean takes into account all values of an attribute, making it a representative of the data. Furthermore, an integer (rounded mean) is more appropriate for the attributes “number of adults” and “number of children”.

```

# calculate mean on training set
median_num_of_adults = round(X_train['number of adults'].mean())
median_num_of_children = round(X_train['number of children'].mean())

# fill these values in null places
X_train['number of adults'].fillna(median_num_of_adults, inplace = True)
X_train['number of children'].fillna(median_num_of_children, inplace = True)

```

Figure 25: impute missing values with rounded mean on train set

```

# Impute missing values for X_test
X_test['number of adults'].fillna(median_num_of_adults, inplace = True)
X_test['number of children'].fillna(median_num_of_children, inplace = True)

```

Figure 26: impute missing values with rounded mean on test set



### 3.3.4 Handling imbalanced dataset

As defined in section 2.3.2, the hotel booking cancelation prediction dataset is skewed with the number of “Not\_Cancelled” instances more than double its counterpart. Kotsiantis et al (2006) suggested some data level methods to handle imbalance:

- Undersampling: eliminate majority class (class with more instances). This method may remove potentially useful data.
- Oversampling: duplicate minority class (class with less instances). The drawbacks of this method is it may introduce over-fitting and additional computational cost. To avoid over-fitting issue, generating synthetic minority class instances with SMOTE (Synthetic Minority Over-sampling Technique) is recommended. SMOTE works by creating new examples by interpolating between existing minority class examples that lie together (k nearest neighbours).

Creating synthetic minority examples is chosen for the hotel booking cancelation prediction dataset. However, SMOTE only works with numeric values. SMOTE-NC is used because it can deal with both numeric and categorical data. It is important to perform synthetic data generation on training set only to avoid contaminating and introducing biases into the models (Aguilar, 2019)

```
from imblearn.over_sampling import SMOTENC

# Specify the categorical feature indices
cat_feature_indices = [4, 6, 8]

smotenc = SMOTENC(categorical_features=cat_feature_indices, random_state = 11)

features = booking.drop(["booking status"], axis=1)
target = booking["booking status"]
features, target = smotenc.fit_resample(features, target)
```

Figure 27: apply oversampling with SMOTENC on training set

```
# Check again after oversampling
from collections import Counter
counter = Counter(y_train)
y_train.value_counts()

1    18316
0    18316
Name: booking status, dtype: int64
```

Figure 28: balance data after oversampling

### 3.3.5 Handling categorical attributes

Most machine learning algorithms only work with numeric data so it is required to transform data of other formats to numeric type. However, if encoding is done incorrectly, it can introduce noise in the data which decreases model performance (Filho, 2023).

The selected algorithm for this project (xgbclassifier) can handle categorical data but this feature is only experimental at the moment and is not guaranteed to provide optimal results (XGBoost, n.d).

To handle categorical values in Hotel booking cancellation dataset , one-hot-encoding method was chosen. One-hot-encoding works by creating “a binary column for each category level and returns a matrix with these binary representation” (Filho, 2023). It is a straight forward and effective approach but the dataset can grow quickly if there are many unique categories. Consider the moderate size of hotel booking cancellation dataset and the small number of unique categories (figure 29), one-hot-encoding is an appropriate method. Furthermore, categorical data in the dataset is nominal which means they do not have natural order or ranking. Therefore, other encoding methods such as Label encoding or Ordinal encoding, which use an integer or alphabetical ordering to represent categories, may imply an unintended order and introduce bias to the model.

```
# check number of unique categories
features[['type of meal', 'market segment type', 'room type']].nunique()

type of meal      3
market segment type  5
room type         7
dtype: int64
```

Figure 29: number of unique categories

```
# one hot encoding
object_columns = features.select_dtypes(include=["object"]).columns
features = pd.get_dummies(features, columns=object_columns)
features = features.replace({True: 1, False: 0})
features.info()
```

Figure 30: perform one-hot-encoding

#	Column	Non-Null Count	Dtype
0	number of adults	45790 non-null	float64
1	number of children	45790 non-null	float64
2	number of weekend nights	45790 non-null	int64
3	number of week nights	45790 non-null	int64
4	car parking space	45790 non-null	int64
5	lead time	45790 non-null	int64
6	repeated	45790 non-null	int64
7	P-C	45790 non-null	int64
8	P-not-C	45790 non-null	int64
9	average price	45790 non-null	int64
10	special requests	45790 non-null	int64
11	month	45790 non-null	int64
12	type of meal_Meal Plan 1	45790 non-null	uint8
13	type of meal_Meal Plan 2	45790 non-null	uint8
14	type of meal_Not Selected	45790 non-null	uint8
15	room type_Room_Type 1	45790 non-null	uint8
16	room type_Room_Type 2	45790 non-null	uint8
17	room type_Room_Type 3	45790 non-null	uint8
18	room type_Room_Type 4	45790 non-null	uint8
19	room type_Room_Type 5	45790 non-null	uint8
20	room type_Room_Type 6	45790 non-null	uint8
21	room type_Room_Type 7	45790 non-null	uint8
22	market segment type_Aviation	45790 non-null	uint8
23	market segment type_Complementary	45790 non-null	uint8
24	market segment type_Corporate	45790 non-null	uint8
25	market segment type_Offline	45790 non-null	uint8
26	market segment type_Online	45790 non-null	uint8

Figure 31: new columns in dataset after one-hot-encoding

### 3.3.6 Feature Scaling

Numerical values that have a considerably diverse scales can cause poor performance for many machine learning algorithms. The two common approaches

for this issue are normalization (or min-max scaling) and standardization. The former works by shifting and rescaling values to a range from 0 to 1. However, the restricted range can be easily affected by outliers that are mapped to 1 (the max value in the range 0 to 1) while other values are crushed to a much smaller range (Géron, 2022, p.133). On the other hand, standardization works subtracting the mean value, and it divides the results by the standard deviation. Hence, values rescaled by standardization are not restricted to a particular range so they are much less affected by outliers.

However, the selected model (xgbclassifier) is a tree-based model and it is not sensitive to the scale of features so scaling is an optional task for this project.

The author has performed model training with and without scaling (standardization); the results showed insignificant differences.

```
# scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Figure 32: python code to perform standardization

## 4 Predictive Modelling / Model Development

### 4.1 The predictive modelling process

The modelling process was completed by training the selected algorithm (xgbclassifier) on transformed training set.

The xgbclassifier was imported from xgboost library and an instance of the model was created. The author set use\_label\_encoder parameter to false because the data has been encoded (section 3.3.5). The eval\_metric is set to 'logloss' because it is a more appropriate metric for binary classification tasks.

```
from xgboost import XGBClassifier

model = XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=11)

model.fit(X_train, y_train)
```

Figure 33: Create an xgbclassifier

### 4.2 Evaluation results on “seen” data

After training model with the training set, cross validation with StratifiedKFold (explained in section 3.2) was performed on “seen” data.

```
from sklearn.model_selection import StratifiedKFold
skf = StratifiedKFold(n_splits=5, random_state=11, shuffle=True)

from sklearn.model_selection import cross_validate
cv_results = cross_validate(model, X_train, y_train, cv=skf,
                           scoring=["accuracy", "f1", "roc_auc"])
print(f"""Accuracy : {cv_results['test_accuracy'].mean()}
F1 Score : {cv_results['test_f1'].mean()}
Roc Auc Score : {cv_results['test_roc_auc'].mean()}""")
```

Figure 34: Python code to cross-validate xgbclassifier on training set

```
Accuracy : 0.9012340640701497
F1 Score : 0.9006668297128808
Roc Auc Score : 0.9665059935458065
```

Figure 35: Cross-validation results

Figure 35 presents evaluation metrics used to assess the performance of the xgbclassifier. The accuracy score of 0.9012 implies that the model predicted the correct class for approximately 90.12% of the instances (the accuracy score here is valid because the dataset has been balanced with SMOTENC technique – section 3.3.4). The F1 score of 0.9007 indicates that the model achieved a good balance between precision and recall. ROC AUC of 0.9665 shows that the classifier has a high ability to distinguish between the classes.

## 5 Evaluation and further modelling improvements

### 5.1 Initial evaluation comparison

After cross-validation, the model was evaluated on “unseen” data – test set.

```
from sklearn.metrics import accuracy_score, f1_score, roc_auc_score

# Make predictions on the test data
y_pred = model.predict(X_test)

# Calculate F1 score
f1 = f1_score(y_test, y_pred)

# Calculate ROC AUC score
roc_auc = roc_auc_score(y_test, y_pred)
```

Figure 36: Python code to evaluate model on test set

```
F1 Score: 0.584788219219675
ROC AUC: 0.6806988132000424
```

Figure 37: Evaluation on test set results

The results obtained from evaluation on test set show significantly lower scores in all metrics. The model has low scores for both precision and recall (0.585 F1 score), and cannot differentiate well between two classes (ROC AUC of 0.68).

The above scores imply an overfitting problem. Overfitting happens when “the model does not generalize well observed data to unseen data” – Ying (2019). In other words, the model demonstrates good performance when evaluated on the training (seen) data, but its performance is poor when tested on the test (unseen) data. Overfitted model tends to learn by heart all the data containing inevitable noise in the training set and ignore the general pattern of the data. This issue will be addressed in section 5.3.

### 5.2 Research team comparison

The evaluation results on test set after training and cross validating on training set are presented in table below.

Algorithm	F1 Score	ROC AUC
-----------	----------	---------

K Nearest Neighbor	85.77%	92.63%
Random Forest	91.14%	97.02%
CatBoost	76.61%	91.80%
XGBClassifier	58.48%	68.07%

Table 5: evaluation results on test set

### 5.3 Further modelling improvements attempted

As defined in section 5.1, the current model is overfitted.

Ying (2019) suggested some methods to reduce the effect of overfitting as follows:

- Early stopping: compute the accuracy after each epoch and stop training when there is no further improvement in accuracy on the test data
- Network-reduction : noise reduction based on pruning in relational (especially decision tree) learning. Pruning works by removing less meaningful or irrelevant data.
- Expansion of training data: collect more data, add some random noise, produce new data based on existing data.
- Regularization: select useful features, remove less helpful features from the model, reduce the weights of features with minimal impact on the final classification.

Reviewing the model development process, the author realised that overfitting may be caused by the large number of features resulting in model complexity. As Ying (2019) explained an overfitted model is more likely to consider all the features regardless of their importance and influences on the final output. In worse cases, some of the features are irrelevant and add noise to the output. Therefore, the regularization method is used to handle overfitting. Specifically, SelectKBest was chosen for this task. SelectKBest is a filter-based feature selection method that uses statistical tests to score and rank features based on their relationship with the output variable. It selects the K features with the highest scores to form the final feature subset, independent of any specific machine learning algorithm (Kavya, 2023). There are two parameters in SelectKBest: score function and k. For score function, the most three common options for classification problem are: chi\_2, mutual\_info\_classif, and f\_classif. The first two options give high scores for features that have high correlation with target variable and features that are informative with respect to target variable. Consider the case of hotel booking cancellation prediction, the author chose the third option (f\_classif) to select features that are highly dependent on the target variable. K determine the number of top features to be pick (10 in this case).

```

# Feature selection
k_best = SelectKBest(score_func=f_classif, k=10)

X = k_best.fit_transform(features, target)
y = target

# Get the indices of the selected features
selected_features_indices = k_best.get_support(indices=True)

# Get the scores associated with each feature
feature_scores = k_best.scores_

# Create a list of tuples containing feature names and scores
feature_info = list(zip(features.columns, feature_scores))

# Sort the feature info in descending order based on scores
sorted_feature_info = sorted(feature_info, key=lambda x: x[1], reverse=True)

for feature_name, feature_score in sorted_feature_info[:10]:
    print(f"{feature_name}: {feature_score:.2f}")

```

Figure 38: python code to perform feature selection with SelectKBest technique.

```

lead time: 9505.91
special requests: 6109.90
market segment type_Online: 1194.92
average price: 966.57
market segment type_Corporate: 925.35
repeated: 774.46
car parking space: 629.76
market segment type_Offline: 457.79
number of adults: 290.59
P-not-C: 212.42

```

Figure 39: top 10 most useful features for model development.

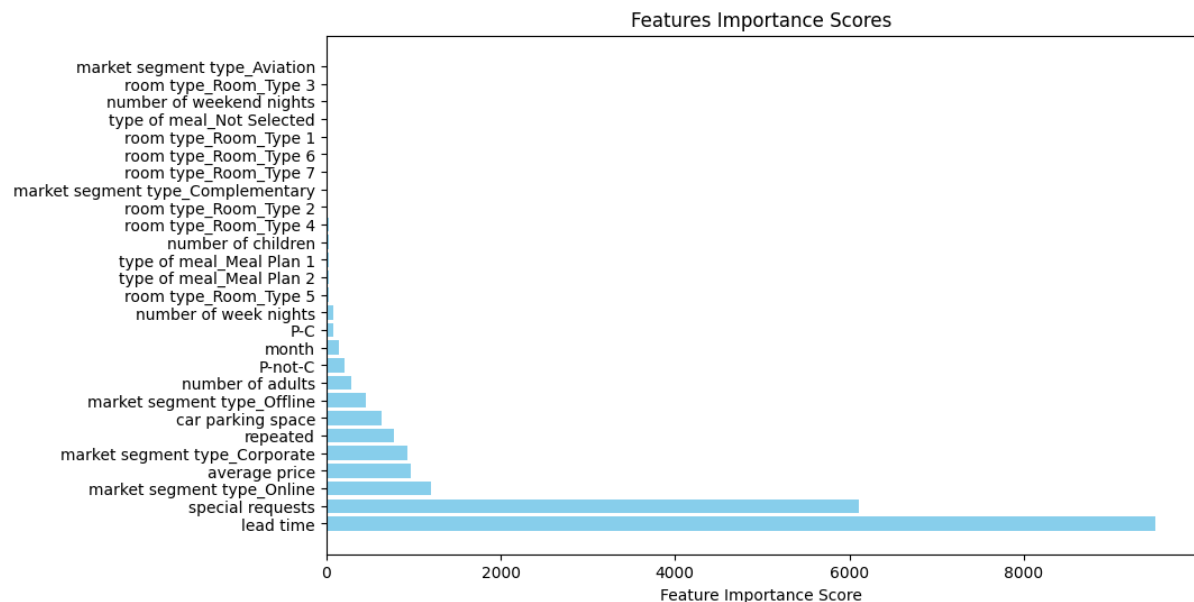


Figure 40: visualization of the relation between features with their score and the target variable

After that, the modelling process was performed in the same manner (same parameters, techniques and protocol) as section 4.1 and cross-validation as section 4.2 but on the newly selected features.

The results obtained from evaluation on training set are slightly lower than results acquired from the overfitted model in section 4.2 - figure 35.

```
F1 Score : 0.8802753515433913
Roc Auc Score : 0.9498872043352871
```

Figure 41: Cross-validation results on training set after feature selection

The model was also evaluated on test set. The results (figure 42) was significantly higher compared to the results produced by the overfitted model.

```
F1 Score: 0.8816109758796193
ROC AUC: 0.8831231719393042
```

Figure 42: Evaluation results on test set after feature selection

Overall, the model trained with select features perform well on both “seen” and “unseen” data. It can be concluded that the problem of overfitting has been solved thanks to feature selection.

## 5.4 Fine-tune the model

The overall score of 0.88 for F1 score and ROC AUC metrics (figure 42) are fairly satisfactory but potential improvements can be achieved with further tuning. Hyperparameters, set prior to the learning process, have a substantial impact on machine learning models. Hence, optimizing the values of hyperparameters is an essential step in model development, significantly influencing the accuracy of predictions. However, manually choosing the right set of hyperparameters to achieve best model performance is a challenging task that requires expertise and understanding of underlying problem. Fortunately, there are more automated approaches such as grid search, random search, Latin hypercube sampling, and optimization (López et al, 2022, pp.109-139). Among these recommended methods, the author decided to use grid search to fine-tune model in this project due to its comprehensive exploration. The grid search technique involves dividing each hyperparameter into a predefined set of values to explore. This approach trains and evaluates models using all possible combinations of the selected values for all hyperparameters. One drawback of grid search is it consumes a great amount of time and computing power. However, the dataset used in this project is fairly moderate and the number of values for hyperparameters are small. The running time of grid search in this case was 6 minutes – fairly acceptable.

The process to fine-tune model with grid search:

- Import GridSearchCV from sklearn.model\_selection
- Define hyperparameter grid of possible values

```
xgboost_params = {"learning_rate": [0.01, 0.5, 0.1, None],
                  "n_estimators": [100, 300, 500, 600, None],
                  "colsample_bytree": [0.5, 1, None]}
```

Figure 43: parameter grid for fine-tuning with grid search

- Create an instance of GridSearchCV with StratifiedKFold cross validation
- Perform grid search



```
grid_search_xgb = GridSearchCV(model, param_grid=xgboost_params, cv=skf)

grid_search_xgb.fit(X_train, y_train)
```

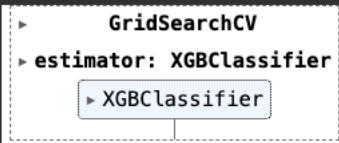


Figure 44: create GridSearchCV instance and perform grid search

- Select best model and feeding it the training data

```
best_model_xgb = grid_search_xgb.best_estimator_

best_model_xgb.fit(X_train, y_train)
```

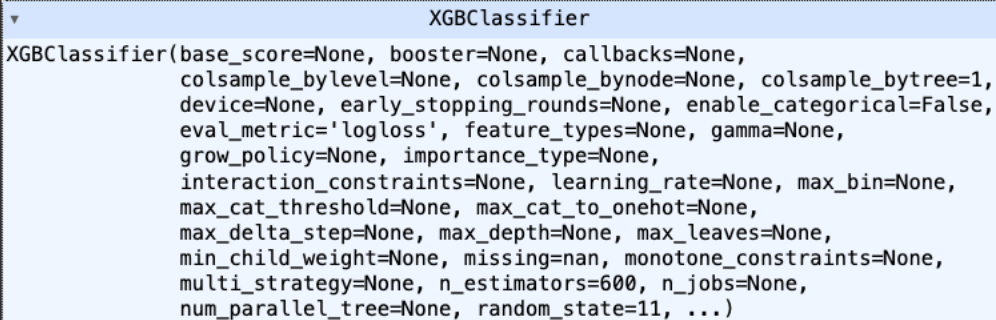


Figure 45: Select best model and feeding it the training data

- Evaluate on test data

The results after fine-tuning are improved by an insignificant amount (0.01 for ROC AUC and even less for F1 score).

```
F1 Score: 0.8899647887323943
ROC AUC: 0.8908260687894589
```

Figure 46: result metrics on test data after fine-tuning with grid search.

## 5.5 Final Evaluation results

The results of final evaluation indicate that the model achieved a high level of performance in terms of classification accuracy and effective discrimination capability. The F1 Score of 0.8899 suggests a balanced combination of precision and recall, indicating that the model effectively identifies both positive and negative instances. Similarly, the ROC AUC value of 0.8908 demonstrates the model's ability to distinguish between the positive and negative classes.

## 5.6 Final team result comparison

The following table displays the evaluation on test set after model improvement

Algorithm	F1 Score	ROC AUC
K Nearest Neighbor	78.85%	84.72%
Random Forest	83.22%	87.60%
CatBoost	81.41%	85.66%



XGBClassifier	89.00%	89.08%
---------------	--------	--------

## 6 Conclusion

### 6.1 Summary of results

The project is about developing a machine learning model to predict hotel booking cancellation. The project was initiated about week 3 of this AI ML module. The team defined the scope for this project and found an appropriate dataset on Kaggle. The team then formed some questions about the dataset as well as the hotel booking domain. These questions were answered with exploratory data analysis (EDA) that was completed individually. Comparing the results from the EDA step, the team discussed to find appropriate methodology and techniques to further process data. Along that, the team also agreed on a common evaluation protocol to ensure model performance results are compared fairly. Following the pre-defined protocol, each team member continued with pre-processing, modelling, cross-validation, and tuning. The team results were compared once before tuning and again after tuning, both on test set ("unseen" data) only.

### 6.2 Suggested further improvements to the model development process

One challenge found during model development process was the potential of data leakage. It is recommended to split data before the EDA. However, there are some steps in pre-processing that need to be done on the whole dataset such as feature engineering of the "date of reservation" attribute or dropping outliers. These tasks were not carried out with sklearn estimators and transformers that facilitate the fit() and transform() methods to work on train and test sets separately. Therefore, if data was splitted before these steps, the author would have to perform these steps on training and test set separately with repeated code.

The lesson learnt is to perform EDA on training set only and then load the dataset again to perform common pre-processing steps before spitting it again for further development. Always keep the concern about data leakage in mind because it leads to inaccurate evaluation results.

### 6.3 Reflection on Research Team

The team was on the same page and put equal contribution to the project. Working within a team helped draw more perspectives to solve a problem. Especially a data related project, each team member can see the data differently and bring insights to make the project more comprehensive.

The data leakage issue may arise and lead to inaccurate result comparison. However, the team has discussed this problem in the early stage and defined a specific protocol to mitigate that.

The team also recognised the problem of imbalanced dataset and asked for advice from the lecturer. After reading the recommended references, the team was able to choose an appropriate method to handle that issue.

## 6.4 Reflection on Individual Learning

The coursework enabled me to extend my knowledge in the field of machine learning and dive deeper in supervised learning (in this case, classification). After weeks of reading papers and performing experiments on multiple datasets, I finally had a chance to putting them all together in this report.

I enjoyed how the coursework was structured which enabled both team discussion and individual work. I loved to discuss with the team to learn from different points of view. We not only worked together to solve the common problem but also helped each other if anyone get stuck. I also love the individual work where I can spending more time reading alone and truly take responsibility for my project.

In the future, I would broaden my machine learning knowledge to understand advanced topics such as neural network and deep learning.

## 7 References

1. Dong, Y. and Ling, L. (2015) 'Hotel overbooking and cooperation with third-party websites', *Sustainability*, 7(9), pp. 11696-11712.
2. Sim, J., Mak, B. and Jones, D. (2006) 'A model of customer satisfaction and retention for hotels', *Journal of Quality Assurance in Hospitality & Tourism*, 7(3), pp. 1-23.
3. Youssef Aboelwafa (2023) 'Hotel Booking Cancellation Prediction'. Available at: <https://www.kaggle.com/datasets/youssefaboelwafa/hotel-booking-cancellation-prediction/data> (Accessed: 11 January 2024).
4. Kaggle (2023) 'Wikipedia'. Available at: <https://en.wikipedia.org/wiki/KaggleBibliography> (Accessed: 11 January 2024).
5. Brownlee, J. (2019) 'Difference Between Classification and Regression in Machine Learning'. Available at: <https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/> (Accessed: 11 January 2024).
6. Sarker, I.H. (2021) 'Machine Learning: Algorithms, Real-World Applications and Research Directions', *SN Computer Science*, 2(3), p. 160.
7. Towards data science (2018) 'Ground Truth Versus Bias'. Available at: <https://towardsdatascience.com/ground-truth-versus-bias-99f68e5e16b> (Accessed: 11 January 2024).
8. Reitermanova, Z. (2010) 'Data splitting', *WDS*, 10, pp. 31-36.
9. Géron, A. (2022) 'Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems', O'Reilly Media.
10. Joseph, V.R. (2022) 'Optimal ratio for data splitting', *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 15(4), pp. 531-538.
11. Seaborn (n.d) 'seaborn: statistical data visualization'. Available at: <https://seaborn.pydata.org/> (Accessed: 12 January 2024).
12. Kotsiantis, S., Kanellopoulos, D. and Pintelas, P. (2006) 'Handling imbalanced datasets: A review', *GESTS international transactions on computer science and engineering*, 30(1), pp. 25-36.
13. Filho (2023) 'How To Handle Imbalanced Data In XGBoost Using scale\_pos\_weight In Python'. Available at: [https://forecastegy.com/posts/xgboost-imbalanced-data-scale\\_pos\\_weight-python](https://forecastegy.com/posts/xgboost-imbalanced-data-scale_pos_weight-python) (Accessed: 13 January 2024).

14. Brownlee (2021) 'How to Develop Your First XGBoost Model in Python'. Available at: <https://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/> (Accessed: 15 January 2024).
15. Brownlee (2020) 'How to Evaluate Gradient Boosting Models with XGBoost in Python'. Available at: <https://machinelearningmastery.com/evaluate-gradient-boosting-models-xgboost-python/> (Accessed: 16 January 2024).
16. Nvidia (n.d) 'XGBoost'. Available at: <https://www.nvidia.com/en-us/glossary/xgboost/> (Accessed: 15 January 2024).
17. Singh, P., Singh, N., Singh, K.K. and Singh, A. (2021) 'Diagnosing of disease using machine learning', Machine learning and the internet of medical things in healthcare, pp. 89-111.
18. Patel (2024) 'Feature Engineering Explained'. Available at: <https://builtin.com/articles/feature-engineering>.
19. Kotsiantis, S., Kanellopoulos, D. and Pintelas, P. (2006) 'Handling imbalanced datasets: A review', GESTS international transactions on computer science and engineering, 30(1), pp. 25-36.
20. Aguilar (2019) 'SMOTE-NC in ML Categorization Models for Imbalanced Datasets'. Available at: <https://medium.com/analytics-vidhya/smote-nc-in-ml-categorization-models-fo-imbalanced-datasets-8adbdcf08c25> (Accessed: 15 January 2024).
21. Filho (2023) 'How To Deal With Categorical Variables in XGBoost'. Available at: <https://forecastegy.com/posts/xgboost-categorical-variables22>.
22. XGBoost (n.d) 'Categorical data'. Available at: <https://xgboost.readthedocs.io/en/stable/tutorials/categorical.html> (Accessed: 16 January 2024).
23. Ying, X. (2019) 'An overview of overfitting and its solutions', Journal of physics: Conference series, 1168, p. 022022.
24. Kavya (2023) 'Optimizing Performance: SelectKBest for Efficient Feature Selection in Machine Learning'. Available at: <https://medium.com/@Kavya2099/optimizing-performance-selectkbest-for-efficient-feature-selection-in-machine-learning-3b635905ed48> (Accessed: 16 January 2024).
25. Montesinos López, O.A., Montesinos López, A. and Crossa, J. (2022) 'Overfitting, model tuning, and evaluation of prediction performance', in Multivariate statistical machine learning methods for genomic prediction, pp. 109-139.
26. Narkhede (2018) 'Understanding AUC - ROC Curve'. Available at: <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5> (Accessed: 16 January 2024).