UNIVERSITY OF HERTFORDSHIRE

School of Physics, Engineering, and Computer Science

MSc Artificial Intelligence and Robotics with Advanced Research

7COM1039- Advanced Computer Science Masters Project

Final Project Report

Prediction of hotel reservation cancellation of a customer with reservation details using Machine Learning.

Name: NIKHIL REDDY MARELLA

Student ID: 20067093

Supervisor: John Lones

Abstract

The hotels have been experiencing drastic changes in their online reservation channels' booking possibilities due to customers' behavior in canceling the hotel reservations in vast numbers as it is made easier to do so free of charge or preferably at a low cost which benefits the hotel guests, but it is a revenue reducing factor for the hotel to deal with. So, this project aims to identify the main reasons behind hotel reservation cancellations and also develop a machine learning model that can predict the customer who might cancel the reservation based on the details provided. This project proposes to explore to answer the 4 research questions in order to understand the importance of dropping unwanted columns, one-hot encoding and its significance to the dataset, and algorithms performance on over-sampled and under-sampled data to deal with imbalanced data and to know what factors are crucial for customer canceling the hotel reservation. The project was conducted in four steps: data cleaning and preprocessing, exploratory data analysis, feature engineering, and model training and evaluation. The project found that the machine learning model trained on over-sampled data using the SMOTE technique performed better than algorithms trained on under-sampled data. The results were discussed in chapters 4 and 5. This Project finds the most important factors that are responsible in accurately predicting the hotel reservation cancellations are discussed in the results section.

Acknowledgment

I would first like to thank my supervisor John Lones for the continuous support, and guidance, and for the valuable feedback throughout the project journey.

Finally, I should express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of education and through the process of researching and writing this thesis.

This accomplishment would not have been possible without my supervisor as he monitored me regularly and helped me to progress in this project support. Thank you.

Declaration

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Artificial Intelligence and Robotics with Advanced Research at the University of Hertfordshire (UH).

It is my own work except where indicated in the report.

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on the university website provided the source is acknowledged (delete as necessary).

Table of Contents

List of Tables

LIST OF FIGURES

CHAPTER I – INTRODUCTION

1.1 Background

Hotel reservation cancellations are on the rise in the hospitality industry due to changes in customer behavior. There can be various reasons for the cancellations like better prices in other hotels, change of plans, scheduling conflicts, etc as mentioned in (www.kaggle.com, 2023). (Antonio, de Almeida and Nunes, 2017) mentioned that to reduce the problems due to cancellations, hotels had implemented strict policies over booking cancellations which in turn has impacted the performance badly on revenue and reputation in the markets. Because of these strict policies, customers might not book a room in the same hotel in the future. Leveraging the capability of machine learning to predict whether the reservation will be canceled or not. There is one statement given by (Romero Morales and Wang, 2010), "It is hard to imagine that someone can predict whether a booking will be canceled or not with high accuracy", but due to the advancements in data science and machine learning, its possible to predict the reservation cancellation with the help of past data obtained by hotels.

In this project, the Hotel Reservations dataset from (www.kaggle.com, 2023) is used for data analysis and experimentation with various machine learning algorithms on sampling techniques to develop a prediction model.

1.2 Research Aim

This project aims to identify the reasons behind the cancellations of hotel reservations and develop machine-learning algorithms on oversampled and under-sampled data to predict the customer, who might cancel the reservation based on the customer details provided using the machine-learning algorithm with the best performance.

1.3 Research Questions

Below are the research questions to investigate in this research work:

1) What are the columns not important for analysis and why were they dropped from the dataset?

2) What is one hot encoding and why it is crucial for analyzing this dataset?

3) What are the classification algorithms considered and which algorithm has given the best performance on comparing Under-Sampled and Over-Sampled Data?

4) Is the average price change and lead time responsible for predicting hotel reservation cancelations?

1.4 Hypothesis

This research would like to identify the below hypothesis:

Hypothesis: Will over-sampled data help achieve better performance than under-sampled data on classification algorithms?

Null Hypothesis: There is no significant difference in the performance of classification algorithms trained on over-sampled data and under-sampled data.

Alternative Hypothesis: Classification algorithms trained on over-sampled data will have better performance than classification algorithms trained on under-sampled data.

1.5 Research Objectives

Below are the objectives of this research work:

- Understand the description of the hotel reservations dataset.
- Data Pre-processing and analyzing the data with respect to the target variable.
- Data Separation into training, validation, and testing sets.
- Understanding Logistic Regression, KNN, Decision Tree, and Random Forest.
- Understanding and implementing the sampling techniques on machine learning classification algorithms considered for this research.
- Performance comparison analysis of the algorithms on over-sampled and under-sampled data.

1.6 Project Planning

Prediction of hotel reservation cancellation of a customer with reservation details using Machine Learning Project Planning

	Milestone description	Progress	Start	End	Days
Project Idea	Data Source and Project Idea	**100%**	23/02/2023	24/02/2023	2
	Data Understanding	**100%**	23/02/2023	24/02/2023	2
	RQ's and Hypothesis Identification	**100%**	25/02/2023	25/02/2023	1
Literature Background	Hotel reservation cancellation papers	**100%**	10/03/2023	12/03/2023	3
	Data Imbalance and Solutions	**100%**	13/03/2023	14/03/2023	2
	RF, DCS, KNN, Logistic Regression	**100%**	15/03/2023	30/03/2023	16
	Cross Validation Techniques	**100%**	01/04/2023	02/04/2023	2

Data Cleaning	Deal with missed values in Columns	**100%**	03/04/2023	03/04/2023	1
	Drop any null value rows	**100%**	03/04/2023	03/04/2023	1
Data Preparation and Visualization	Understanding visualization Plots	**100%**	15/04/2023	20/04/2023	6
	Plotting Visualizations	**100%**	15/04/2023	20/04/2023	6
	learn code for one hot encoding	**100%**	21/04/2023	22/04/2023	2
	One hot encode on categorical coumns	**100%**	21/04/2023	22/04/2023	2
Model Building	Split train and test	**100%**	25/04/2023	25/04/2023	1
	Standardize the data	**100%**	25/04/2023	25/04/2023	1
	ML algorithms on Over-Sampled data with GridSearch	**100%**	26/04/2023	15/05/2023	19
	ML algorithms on Under-Sampled data with GridSearch	**100%**	16/05/2023	30/05/2023	15
	Test on unseen data	**100%**	01/06/2023	02/06/2023	2
	Performance Comparision	**100%**	03/06/2023	04/06/2023	2
FPR Report	Abstract	**100%**	10/07/2023	10/07/2023	1
	Introduction	**100%**	11/07/2023	13/07/2023	3
	Literature Review Search	**100%**	14/07/2023	25/07/2023	12
	Research Methodology	**100%**	21/07/2023	30/07/2023	10
	Results	**100%**	01/08/2023	10/08/2023	10
	Conclusion and Future Work	**100%**	12/08/2023	15/08/2023	4
	References	**100%**	16/08/2023	16/08/2023	1
	Appendices	**100%**	17/08/2023	18/08/2023	2
	Plagarism Check	**100%**	19/08/2023	19/08/2023	1
	Final Edits and Review	**100%**	20/08/2023	25/08/2023	6
	Final Plagarism Check	**100%**	25/08/2023	25/08/2023	1
	Meet with Supervisor	**100%**	26/08/2023	26/08/2023	1
	FPR Submission	**100%**	28/08/2023	28/08/2023	1

Table 1: Project Planning

1.7 Chapter Overview

Below are the chapters discussed in this research project.

Chapter I: Introduction

This chapter discusses the background of hotel reservation cancellations and their impact and why machine learning is needed to deal with those cancellations by customers. The research aim, and research questions to be answered are mentioned, and also the hypothesis and research objectives of this project are discussed. The overall idea of this research work will be discussed in this section.

Chapter II: Literature Review

In this chapter, a detailed introduction to Hotel Reservation cancellation and challenges will be discussed and in the later sections, the importance of one-hot encoding, data imbalance, and its solutions will be reviewed. And later, Standardization, grid-search, and their importance for this project will be discussed followed by an overview of machine learning classification algorithms will be discussed followed by the research gap identified.

Chapter III: Research Methodology

The research process for this project will be discussed In this chapter starting with a detailed review of the Hotel reservations dataset and then the pre-processing techniques used and how the one-hot encoding is applied in this dataset will be discussed. Data visualizations will be interpreted and In the next sections, data separation and feature standardization will be discussed. Later, how data is over-sampled and under-sampled and applied to machine learning algorithms with the best parameters from grid-search will be discussed here. In the last, Performance metrics will be discussed followed by a summary.

Chapter IV: Results and Discussion

In this section, the experimentation setup used for this project and the pre-processing and data visualization results will be discussed here. Later machine learning algorithms' performance on the over-sampled data and under-sampled data will be analyzed and the results be compared to answer the research questions and hypotheses.

Chapter V: Conclusion and Future Work

The answers to research questions will be concluded in this chapter, which also provides a summary of the report's conclusions. It also examines if the objectives were achieved and offers some suggestions for further study.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction to Hotel Reservation Cancellation

Providing a warm and welcoming environment for visitors, guests, or strangers is known as hospitality. The hospitality sector is wide within the service sector and includes hotels, restaurants, bars, and other accommodations, eating, events, entertainment, travel, and tourism-related businesses as mentioned in (Shirisha et al., 2023). The cancellation of a reservation plays a significant role in the decision-making process for associated demands in the hotel management system. It influences the hotel's services and financial status and makes appropriate output predictions. As mentioned in (Shirisha et al., 2023), hotel reservation cancellations have been increasing since 2013 due to unusual reasons of the customers like illness, change of plans, better prices at other hotels, lack of facilities, etc., which are the increasing the losses to the hotel. To minimize this kind of loss, hotels have made strict policies and rules about cancellations. The fact mentioned in (Satu, Ahammed, and Abedin, 2020) say that these regulations can lower service quality, and fewer refunds/non-refund policies decrease income and reservations as well, so they aren't always effective. By providing perks, discounts, or admission to an amusement park for such reservations, hotels may lessen the effect of potential cancellations and cut down on the need for aggressive overbooking as discussed in (Shirisha et al., 2023).

Also due to the online reservation systems (ORS), many people are traveling for vacations as tourism is growing at around 4% per year as mentioned in (He et al., 2018). Due to advancements in tech and the rise in tourism across the world, ORS has been widely used in the hospitality sector. While using an ORS increases bookings for hotels, it also makes it simpler for consumers to cancel due to changes in plans. Therefore, hotels must establish explicit cancellation procedures, especially given the prevalence of ORS. Customers worry about two things when they book a hotel: the cancellation deadline and the amount they will receive in refunds if they prepay and decide to cancel. But as discussed, cancellation and refund policies will incur some sort of loss to the hotel. To get rid of this hotel reservation cancellation problem, predicting whether a hotel reservation will be cancelled or not using machine learning is of necessary requirement for many hotels in order to reduce the loss. This project is about predicting the hotel reservation cancellation provided the customer details.

2.2 What is One-Hot Encoding

One-hot encoding is a technique used to represent categorical data as numerical values in a machine-learning model as mentioned in (Brownlee, 2017). This is done by creating a new binary feature for each possible category and assigning a value of 1 to the feature of each sample that corresponds to its original category. For example, in this project for the “market_segment_type” categorical feature has five categories, then one-hot encoding would create 5 new binary features, one for each category. The feature for the category that the sample belongs to would be assigned a value of 1, and the features for the other two categories would be assigned a value of 0.

market_segment_type		market_segment_type_Aviation	market_segment_type_Complementary	market_segment_type_Corporate	market_segment_type_Offline	market_segment_type_Online
Aviation	one-hot encoding	1	0	0	0	0
Complementary		0	1	0	0	0
Corporate		0	0	1	0	0
Offline		0	0	0	1	0
Online		0	0	0	0	1

Fig 1: One-hot encoding example for market type segment category

Also, column names will be created for each column with respective categories like "**Column Name + Category Name**". For Example: the "market_segment_type" column has 5 categories namely Aviation, Complementary, Corporate, Offline, and Online. Below is how it will create the column names.

"market_segment_type" + "Aviation" = market_segment_type_Aviation

"market_segment_type" + "Complementary" = market_segment_type_Complementary

"market_segment_type" + "Corporate" = market_segment_type_Corporate

"market_segment_type" + "Offline" = market_segment_type_Offline

"market_segment_type" + "Online" = market_segment_type_Online

2.3 Data Imbalance

The class imbalance problem arises when certain classes have significantly more instances than others. The accuracy of categorization can be greatly impacted by this imbalance, especially when forecasting instances of minority classes. Addressing class imbalance is a novel problem inside the machine learning framework. Class imbalance is a widespread problem across many domains. In response to this issue, several research studies have focused on using sampling strategies to improve classification performance.

2.3.1 How Data imbalance will impact the performance of a model

It might be difficult in computing to deal with unbalanced data. When one class dominates another in a dataset, that class is said to be the majority class. In contrast, a class is said to be in the minority when it is represented in the dataset by fewer instances than the other class. This kind of dataset is called an imbalanced dataset. Unbalanced data is a serious problem. (Spelmen and Porkodi, 2018) said that consider a database where 90% of the data falls under the majority category and just 10% falls under the minority category. 90% accuracy is attained using a classification method that simply identifies everything as belonging to the majority class as mentioned by (Spelmen and Porkodi, 2018). We must balance unbalanced data in order to improve classification accuracy and provide accurate predictions.

2.3.2 Solutions for Data Imbalance

To deal with the data imbalance, there are data-level methods like over-sampling and under-sampling which are discussed below.

2.3.2.1 Over Sampling

As mentioned in (Spelmen and Porkodi, 2018) by replicating more copies of the minority class, oversampling helps balance classes, although it doesn't bring any new information and can lead to overfitting.

Traditional oversampling has the important drawback of increasing minority class instances, but frequently in a relatively similar region of the feature space. (Chawla et al., 2002) developed SMOTE, a method that generates "synthetic" instances rather than just replicating the minority class, to overcome this issue. SMOTE is popular and frequently performs better than random or straightforward oversampling.

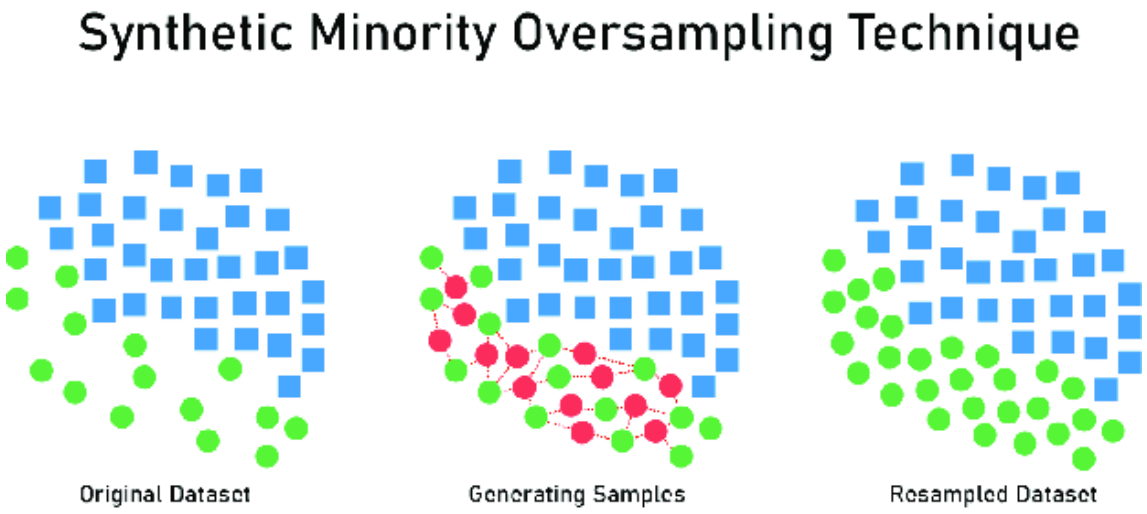


Fig 2: Data generation using SMOTE for the minority class.

[Source: (Sharma, Singh, and Chandra, 2022)]

2.3.2.2 Under Sampling

According to (Hasanin and Khoshgoftaar, 2018), Under-sampling the majority class removes the instances from the majority and the latter adds instances to the minority class. Random under-sampling (RUS) is based on randomly removing the majority class, but other methods selectively undersample the majority class while keeping the original population of the minority class. Tough undersampling can lead to loss of the original data from the majority class but it is preferred if people want to experiment only on the original data instead of the synthetic data generated. In this project random undersampling technique is also used to experiment on the algorithms.

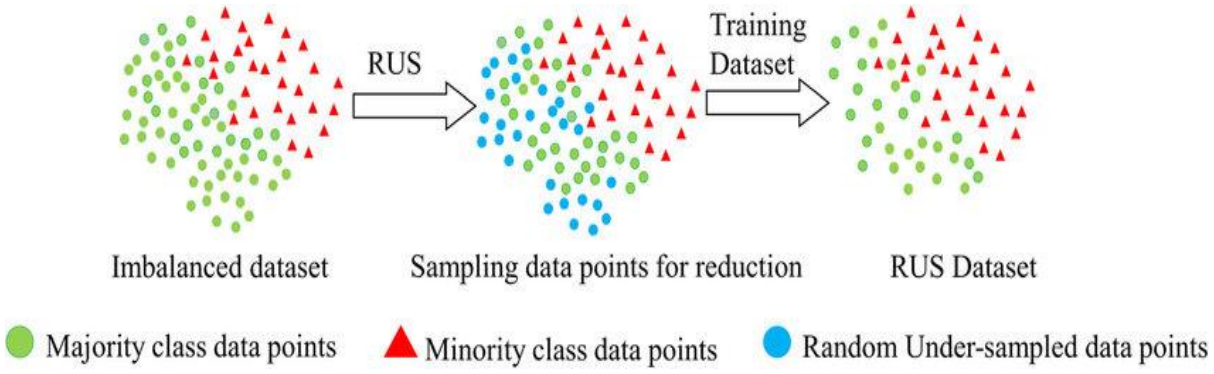


Fig 3: Data reduction using RUS from the majority class.

[Source: (Aldraimli et al., 2020)]

2.4 Feature Scaling

Feature Scaling is a method for ensuring that all the various characteristics in a dataset work well together. (Bhandari, 2020) mentioned that features might occasionally have inconsistent sizes or units, which can screw up a model. This is resolved via scaling using the standardization technique as it makes data simpler for machines to understand.

2.4.1 What is Standardization

Data standardization is a preprocessing technique that modifies data such that its mean (average) is 0 and its standard deviation is 1 as mentioned in (Bhandari, 2020). All aspects may be immediately compared by making sure that each characteristic has a comparable scale. It is especially useful when working with variables that have a wide range of units. By first removing the mean and then dividing it by the standard deviation, each data point is subjected to standardization. Below is the formula for standardization:

$$X' = \frac{X - \mu}{\sigma}$$

Where μ is the mean of the column values and σ represents the standard deviation of the column values.

2.4.2 Benefits of Standardization

Below are the benefits of the standardizing the data:

- It ensures the equal contribution of all features for better performance of the models.

- During the training of machine learning algorithms, standardization helps the model to converge faster in the process of finding solutions.

- Standardisation strengthens models against outliers and lessens the distortion of the data distribution by reducing the impact of extreme values on the mean and standard deviation.

2.5 Hyperparameter tuning with Grid Search Cross Validation

Machine learning algorithms are essential in the decision-making process that is data-driven. To improve the model's internal settings, or hyperparameters, to improve model performance. GridSearchCV is one of the Python tools that can help in the process of determining the ideal hyperparameter values.

2.5.1 What is Grid Search Cross Validation

The ultimate goal of fine-tuning is to find the best hyperparameter values for accurate model predictions. It takes a lot of time to manually search via trial and error. Techniques like Random Search and Grid Search were devised to overcome this. Different hyperparameter combinations are thoroughly evaluated through Grid Search. Grid Search evaluates each hyperparameter combination's performance and chooses the optimal values. However, especially when there are several hyperparameters, it might be computationally costly.

As mentioned in (Shah, 2021), In GridSearchCV, cross-validation aids in model training, improving Grid Search. Instead of just splitting the data into train and test sets, cross-validation further splits the training data into "k" divisions, with one for validation and the remaining for training in each iteration. In general, model performance is determined for each iteration using K-fold Cross-Validation, and the results are averaged. This process takes a lot of time when using grid search and it requires a lot of computing, but it is necessary to find the best hyperparameters.

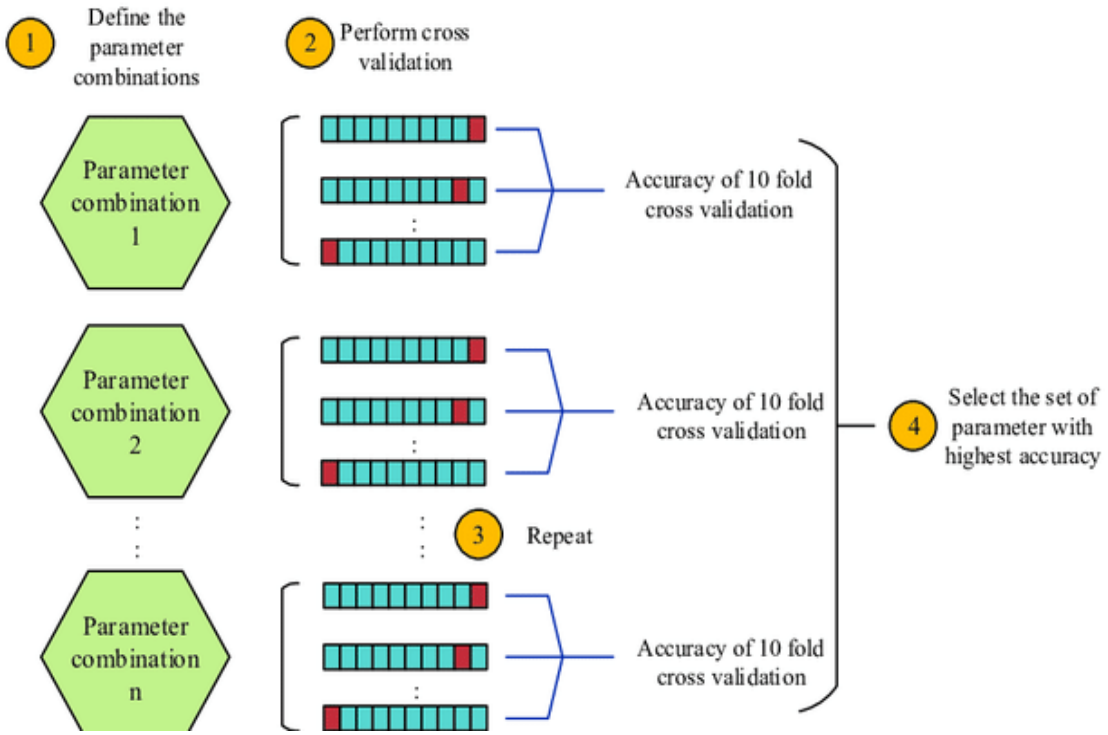


Fig 4: Grid Search Cross Validation Operation

[Source: (Nurul Liyana Hairuddin, Lizawati Mi Yusuf and Mohd Shahizan Othman, 2020)]

2.5.2 Advantages and Disadvantages of Grid Search CV

Advantages:

- A lot of time will be saved using GridSearchCV, from doing manual trial and error to find the best hyper-parameters (s, 2021).
- GridSearchCV is flexible and it has its own parameters that can be changed to meet our goals.

Disadvantages:

- If the computational resources are poor and there are a lot of hyper-parameters, it will cost time as mentioned in (s, 2021).
- The primary goal of GridSearchCV is to decrease bias error, however, it may neglect variance error, thus creating high variance problems (s, 2021).

2.6 Overview of Machine Learning Classification

Machine learning is defined as the subset of artificial intelligence that gives the ability to the computer to learn and improve from the experience without being explicitly programmed. Machine learning is divided into 3 categories namely, supervised learning which deals with classification and regression problems with labels. Meanwhile, unsupervised learning deals with the data without labels and the semi-supervised learning idea is to generalize better on large unlabelled data using a small amount of labeled data. In this project, we are dealing with a supervised classification task.

2.6.1 Logistic Regression

Modeling the probability that a certain class or event will occur using the supervised machine learning approach is known as logistic regression. Essentially, binary classification problems, where the objective is to predict an output variable that belongs to one of two discrete groups, are the main applications of logistic regression as mentioned in (Bonthu, 2021). Logistic regression is mostly employed to answer Yes or No, Cancelled or Not Cancelled, Win or Lose, etc. The output is a probability value between 0 and 1.

Multinomial logistic regression is also available which comes into the picture if the output variable consists of three or more classes without natural ordering as mentioned in (Bonthu, 2021).

The equation for the logistic regression is

$$p(X; b, w) = \hat{y} = \frac{1}{1+ e^{-w.X+b}} = \frac{1}{1+ e^{-z}}$$

Similarly, with 'n' predictors the logistic regression equation is as below.

$$\hat{y} = \frac{1}{1 + e^{-(w1.X1+w2X2+w3X3+ --- +wnXn)+b0}}$$

Where *Xn* is the nth observation of X and [w1, w2, w3, ….. , wn] are the weights and b is the bias term.

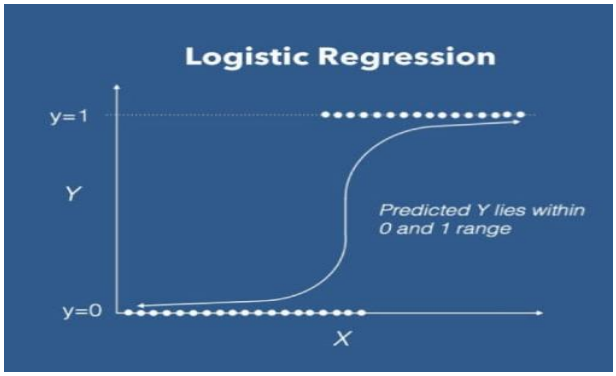


Fig 5: Logistic Regression [Source: (Bonthu, 2021)]

2.6.2 K-Nearest Neighbours

K-Nearest Neighbours (KNN) is a supervised learning non-parametric classification algorithm that is famous for its simplicity and efficiency as mentioned in (Taunk et al., 2019). Data is classified by KNN by looking at its near neighbors from the training dataset within a predefined area. This approach is favored because of its simple implementation and low processing demands. It uses Euclidean distance to calculate closeness for continuous data. As mentioned by (Taunk et al., 2019) KNN determines the K closest neighbors for a new input and classifies it according to the dominant class among these neighbors. Although this classifier is simple, the choice of the 'K' value has a big impact on how unlabeled data is classified.

Below are the steps for KNN working according to the points mentioned in (Taunk et al., 2019):

$$\{ (x(1), y(1)) , (x(2), y(2)), \ldots\ldots , (x(m), y(m)) \}$$

Step 1: Initially, training set given has to be stored.

Step 2: Now for each new unlabeled data, Euclidean distance has to be calculated for all the training data using the formula:

$$\sqrt{\sum_{1=1}^{n} (x_i - y_i)^2}$$

"Find the k-nearest neighbours with the specified K-value for the new datapoint and assign the class containing the maximum number of nearest neighbours" as mentioned in (Tanuk et al., 2019).

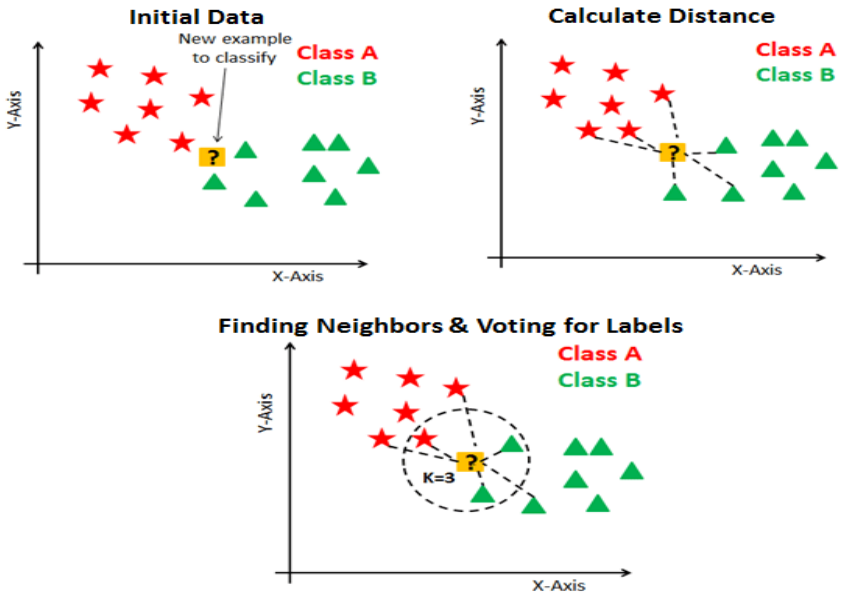


Fig 6: In-depth intuition of KNN Algorithm Working

[Source: (S, 2020)]

2.6.3 Decision Trees

Similar to flowcharts, decision trees are graphic representations that may be used to categorize and sort data. A decision tree is a very important algorithm in supervised learning. There are two types: classification trees for categories and regression trees for continuous data. These trees operate by recursively separating data in a "top-down" manner, breaking data down incrementally. They continue separating until the data is comparable enough to satisfy user needs.

The thing to note here is excessive splitting might result in overfitting, where the data is too much categorized. This can be solved by pruning the tree. Though there are various ways to find the best feature at each node, the two methods namely information gain and entropy are the popular splitting techniques for decision tree models as mentioned in (IBM, 2022).

Knowing entropy is useful for understanding information gain. Entropy is a metric for assessing how disorganized a dataset is, with values ranging from well-sorted (0) to complete confusion (1). In order to create a successful decision tree, we try to select features that reduce entropy, or how chaotic things get. We select the traits with the lowest entropy in order to construct the best decision tree.

$$\text{Entropy}(S) = -\sum_{c \in C} p(c)\log_2 p(c)$$

- S stands for the collection of data from which entropy is calculated.
- The classes in set S are represented by c.
- The percentage of data points in a collection that are members of class C is shown by the symbol p(c).

Entropy before and after a split on a particular property are different, and this difference is represented by information gain. Because it performs the best at categorising the training data in accordance with its goal classification, the attribute with the maximum information gain will result in the best split.

$$\text{Information Gain}(S,a) = \text{Entropy}(S) - \sum_{\text{vevcalues}(a)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

- ‘a’ stands for a specific class label
- Entropy(S) represents the entropy of dataset S.
- “|Sv|/ |S| represents the proportion of the values in Sv to the number of values in dataset, S” (IBM, 2022).
- “Entropy(Sv) is the entropy of dataset, Sv” (IBM, 2022).

Gini impurity is the likelihood that a randomly chosen data point in the dataset would be wrongly classified if it were labeled using the dataset's class distribution. Similar to entropy, the impurity of a set, S, if it is pure (belonging to one class), is zero. This is shown by the formula:

$$\text{Gini Impurity} = 1 - \sum_i (p_i)^2$$

2.6.4 Random Forest

Random Forest is a very powerful supervised machine learning algorithm and is most frequently used for both classification and regression tasks. Random Forest operation is very simple since we know the operation of the decision tree in the above section.

As mentioned in (E R, 2021), "Random Forest builds decision trees on various data points and takes the majority vote in case of classification and in case of regression, it will take average. The random forest method produces a consolidated and more accurate result by integrating the outputs of various trees.

The Process of the Random Forest Algorithm

Step 1: Each decision tree in the Random forest model is built using a subset of attributes and a subset of data points. In simple terms, m features and n randomly chosen records are obtained from a data collection of k records.

Step 2: For each sample, a unique decision tree is built.

Step 3: An output will be produced by each decision tree.

Step 4: For classification and regression, the final result is evaluated using the majority vote or averaging.

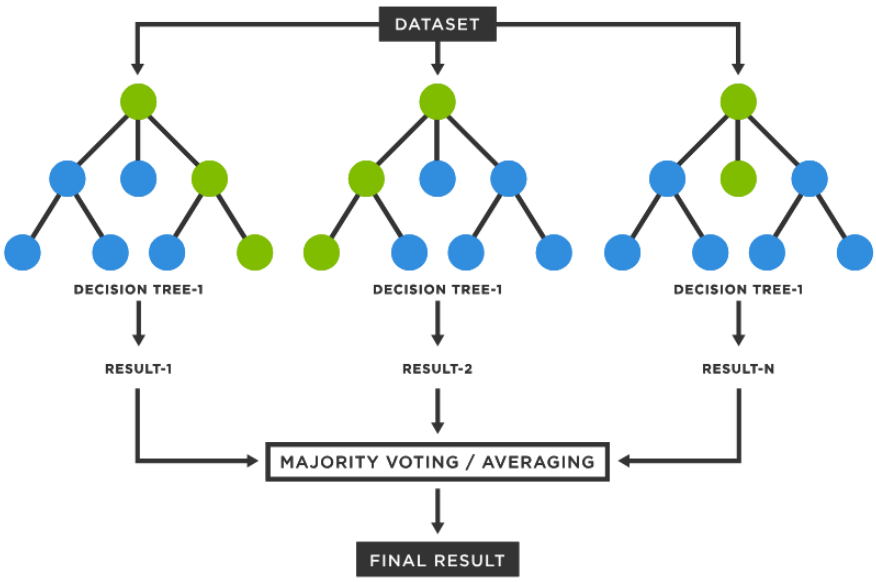


Fig 7: Random Forest Representation

[Source: (TIBCO, 2023)]

2.6.4.1 Random Forest Feature Importance

Feature importance is the key concept in machine learning as it tells, which features are most predictive of the target variable (most important) and tells the least important features to safely ignore them. “The feature importance is measured using **average impurity decrease** which is computed from all the decision trees” as mentioned in (Data Analytics, 2020).

2.7 Research Gap

The dataset for this research is imbalanced. As everyone knows, oversampling and undersampling techniques will help to get rid of imbalance issues. There are many papers dealing with sampling techniques individually on the machine learning algorithms. Hence the research gap identified is to know whether over-sampled data or under-sampled data gives the best performance on machine learning algorithms on predicting the hotel reservation cancellation. So, the machine learning algorithms will be applied on over-sampled and under-sampled data and the results will be compared to know which algorithm has given the best performance in predicting whether the customer will cancel the hotel reservation or not based on the details provided.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

As discussed in the previous chapter, finding the customer who's going to cancel the reservation made in the hotel at the time of booking will give the hotel management a clear idea of what kind of services should be provided to the customer. In this project, developing and choosing the best reservation cancellation prediction system for a hotel with the experimentation of machine learning algorithms on over-sampled and under-sampled data in the process of dealing with imbalanced data, is the research gap identified in the previous chapter. This chapter will discuss the process involved in this research and what are the methods used to develop this project and the performance metrics will also be discussed.

3.2 Hotel Reservations Dataset

For this project, the Hotel Reservations dataset from Kaggle is considered. This dataset consists of 36275 rows and 19 columns. The description for each column is given below to understand the data along with separating the independent and dependent (target) variables as provided in (www.kaggle.com, 2023).

Independant Variables

"***Booking_ID**:* unique identifier of each booking"
"***no_of_adults**:* Number of adults"
"***no_of_children**:* Number of Children"
"***no_of_weekend_nights**:* Number of weekend nights (Saturday or Sunday) the guest stayed or booked to stay at the hotel"
"***no_of_week_nights:*** Number of week nights (Monday to Friday) the guest stayed or booked to stay at the hotel"
"***type_of_meal_plan:*** Type of meal plan booked by the customer"
"***required_car_parking_space:*** Does the customer require a car parking space? (0 - No, 1- Yes)"
"***room_type_reserved:*** Type of room reserved by the customer. The values are ciphered (encoded) by INN Hotels."
"***lead_time:*** Number of days between the date of booking and the arrival date"
"***arrival_year:*** Year of arrival date"
"***arrival_month:*** Month of arrival date"
"***arrival_date:*** Date of the month"
"***market_segment_type:*** Market segment designation."
"***repeated_guest:*** Is the customer a repeated guest? (0 - No, 1- Yes)"
"***no_of_previous_cancellations:*** Number of previous bookings that were canceled by the customer prior to the current booking"
"***no_of_previous_bookings_not_canceled:*** Number of previous bookings not canceled by the customer prior to the current booking"
"***avg_price_per_room:*** Average price per day of the reservation; prices of the rooms are dynamic. (in euros)"
"***no_of_special_requests:*** Total number of special requests made by the customer (e.g., high floor, view from the room, etc.)"

Dependant Variables

"***booking_status*:** Flag indicating if the booking was canceled or not."

There are 5 object data type columns, 1 float64 data type column, and the remaining 13 columns are int64 type columns. With this data, further research will be carried on.

3.3 Data Pre-processing

3.3.1 Missing Values

Sometimes there will be scenarios where there will be no data for a respective feature, in that case, the value will be missed, This is just one scenario there will be other cases like corrupted files, lost files, and many other reasons as mentioned in (Bhandari, 2022). Missing data can create trouble because they can introduce bias into the model, affecting the performance of real-time data. This research identifies missing values using simple functions like "**.info()"** and "**.isnull()".**

dataframe.info() will give a clear picture of the data available like it gives how many no. of rows and for each column, it will show the Non-Null count and also show as "null" if there is any missing data.

dataframe.columns.isnull() will give you the Boolean values as a result for each column i.e., if there is a missing value, it will show as True otherwise it shows False. For the columns in the data frame available find if there are any missing values, and what is the meaning of dataframe.columns.isnull()

3.3.2 Removing unwanted columns.

As clearly mentioned in (Testprep Training Tutorials, n.d.), It is crucial to reduce as much data as possible for loading into the machine learning models. Data should be cleaned before sending training machine learning algorithms in order to reduce the bias. Removing the unused columns will also increase the performance of the data. In this project, initially, the columns that are not important for the research are analyzed and removed from the dataset using the below function:

dataframe.drop(labels, axis=1)

axis = 0 will drop the rows and axis=1 will drop the columns. The columns' names should be given in place of labels.

3.4 One-Hot Encoding

There are 2 categorical columns considered important in independent variables i.e., the column room_type_reserved has 7 categories and market_segment_type has 5 categories. One hot encoding has been applied to these two columns which resulted in 7 columns for 7 categories in room_type_reserved and 5 columns for 5 categories in market_type_segment. After applying one-hot encoding, a total 27 columns are available in the data frame as shown in below figure after dropping two columns as they are not important.

```
: data_encoded.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36275 entries, 0 to 36274
Data columns (total 27 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   no_of_adults                          36275 non-null  int64
 1   no_of_children                        36275 non-null  int64
 2   no_of_weekend_nights                  36275 non-null  int64
 3   no_of_week_nights                     36275 non-null  int64
 4   required_car_parking_space            36275 non-null  int64
 5   lead_time                             36275 non-null  int64
 6   arrival_year                          36275 non-null  int64
 7   arrival_month                         36275 non-null  int64
 8   arrival_date                          36275 non-null  int64
 9   repeated_guest                        36275 non-null  int64
 10  no_of_previous_cancellations          36275 non-null  int64
 11  no_of_previous_bookings_not_canceled  36275 non-null  int64
 12  avg_price_per_room                    36275 non-null  float64
 13  no_of_special_requests                36275 non-null  int64
 14  room_type_reserved_Room_Type 1        36275 non-null  uint8
 15  room_type_reserved_Room_Type 2        36275 non-null  uint8
 16  room_type_reserved_Room_Type 3        36275 non-null  uint8
 17  room_type_reserved_Room_Type 4        36275 non-null  uint8
 18  room_type_reserved_Room_Type 5        36275 non-null  uint8
 19  room_type_reserved_Room_Type 6        36275 non-null  uint8
 20  room_type_reserved_Room_Type 7        36275 non-null  uint8
 21  market_segment_type_Aviation          36275 non-null  uint8
 22  market_segment_type_Complementary     36275 non-null  uint8
 23  market_segment_type_Corporate         36275 non-null  uint8
 24  market_segment_type_Offline           36275 non-null  uint8
 25  market_segment_type_Online            36275 non-null  uint8
 26  booking_status                        36275 non-null  object
dtypes: float64(1), int64(13), object(1), uint8(12)
memory usage: 4.6+ MB
```

Fig 8: Data frame info after applying one-hot encoding.

3.5 Data Visualization

For this dataset, data visualization also plays a crucial role in understanding the characteristics of each column. In this research, knowing which categories in the columns occur more to identify the patterns with respect to the target column. 3 types of visualizations are used in this research for better understanding of the data. These are described below:

Bar Graph is applied to the object-type columns and some other columns i.e.,

['room_type_reserved', 'market_segment_type', 'booking_status', 'no_of_adults', 'no_of_children', 'no_of_weekend_nights', 'no_of_week_nights', 'arrival_year', 'required_car_parking_space', 'arrival_month', 'repeated_guest', 'no_of_special_requests' 'no_of_previous_cancellations']

The **histogram** is plotted for the columns [‘avg_price_per_room’, ‘lead_time’].

The **Grouped bar chart** is plotted for,

1) ‘room_type_reserved’ with respect to ‘booking_status’ column.
2) ‘market_segment_type’ with respect to ‘booking_status’ column.

3.6 Training and Testing Data Separation

The train_test_split() function is a function from the sklearn.model_selection library. It is used to split a dataset into two parts: a training set and a test set. The training set is used to train a model, and the test set is used to test the model's accuracy.

The data_encoded.iloc[:, :-1] part of the code specifies the features of the dataset. The features are the columns of the dataset that are not the target variable. In this case, the target variable is the last column of the dataset.

The data_encoded.iloc[:, -1] part of the code specifies the target variable.

In the initial split, The test_size=0.15 part of the code specifies the size of the test set. In this case, the test set will be 15% of the total dataset. The random_state=42 part of the code specifies the random seed. This is used to ensure that the same split is produced every time the code is run.

In summary, the line of code train_test_split(data_encoded.iloc[:, :-1], data_encoded.iloc[:, -1], test_size=0.1, random_state=42) splits the dataset into a training set and a test set. The training set will be 85% of the total dataset, and the 15% will be the test set of the total dataset which will be used for the second split with 20% of the data separation into test set and 80% into validation data.

3.7 Feature Standardization

In this research, after the data is oversampled using SMOTE or RUS, standardization will be applied to that respective data. Standardization will be applied to only the independent variables on the dataset hence it will be applied to respective train data, validation data, and test data in both the oversampled and under-sampled data.

For example on oversampled data:

```
# Standardization of the values
scaler = StandardScaler()
train_data_osampled = scaler.fit_transform(train_data_osampled)
test_data_osampled = scaler.transform(test_data)
val_data_osampled = scaler.transform(val_data)
```

The first line of code, *scaler = StandardScaler()*, creates a new object of the StandardScaler class. The StandardScaler is a function from the sklearn.preprocessing library. It is used to standardize the values in a dataset.

train_data_osampled = scaler.fit_transform(train_data_osampled), val_data_osampled = scaler.transform(val_data), and test_data = scaler.transform(test_data), use the StandardScaler object to standardize the values in the train_data_osampled, test_data, and unseen_data datasets.

The fit_transform() method is used to fit the StandardScaler object to the train_data_osampled dataset. This means that the StandardScaler object will learn the mean and standard deviation of the values in the train_data_osampled dataset.

The transform() method is then used to standardize the values in the test_data and unseen_data datasets. This is done by subtracting the mean of the train_data_osampled dataset from the values in the test_data and unseen_data datasets and then dividing the values by the standard deviation of the train_data_osampled dataset.

In summary, Standardization of the values standardizes the values in the train_data_osampled, test_data, and unseen_data datasets. This makes the values in the datasets more comparable to each other, which can improve the performance of machine learning models.

Similarly, standardization on under-sampled data is also performed.

3.8 Oversampling

3.8.1 Oversampling using SMOTE

The hotel reservation dataset is imbalanced data with 24390 samples in the Not_Canceled Category and 11885 samples in the Canceled Category. Hence the oversampling technique is applied to the data.

The sampling technique will be applied only to the training data as it should be balanced for training the model and generalize well on the validation and test data. So, oversampling will generate the samples using interpolation between the data points minority class and balance the data. This will be done using the SMOTE function which is discussed in the literature review. After oversampling, the data consists of 41584 rows and 26 columns. This data will be used for the further process.

3.8.2 Grid Search on ML Algorithms on Over-Sampled Data

In this step, oversampled data is used to perform the grid search with the specified user-chosen hyper-parameters for the respective parameters to train the machine learning models. This process will take a lot of time as it will experiment with all the combinations of hyper-parameters to identify the parameters that give the best performance. Hence, this part is crucial. In this project grid search will be applied to the only KNN, Decision tree, and Random Forest. Logistic regression does not have any hyper-parameters hence it will be operated normally.

The first step is to define the parameters grid which consists of user-chosen hyper-parameters. In the next step, the machine learning algorithm has to be defined. In the third step, the grid

search will be created using the GridSearchCV function and stored in the "grid_search" variable and the chosen machine learning algorithm stored in a variable will be passed along with the parameter grid defined previously, and the performance metric for the validation along with the no. of cross-validations (cv).

In the next step, grid_search will be passed with the oversampled data, it will fit the data to the chosen algorithm and will experiment with the chosen hyper-meters combination. After the training, the best parameters can be seen using the function "*grid_search.best_params_".*

3.8.3 Model Building with Best Params of Algorithms on Over-Sampled Data

After the best parameters are identified, the respective machine learning model will be passed with those parameters, and the model will be fitted with the oversampled data and trained. In the next step, the responses for train data and validation data will be predicted and evaluated using the accuracy score and confusion matrix.

In the final step, the model with the best parameters will be tested on unseen test data and accuracy will be calculated.

3.9 Undersampling

3.9.1 Undersampling using RUS

The imbalanced data with 24390 rows and 11885 rows for Not_Canceled and Canceled Categories are passed to a random under-sampling process to reduce the rows of the majority class to match with the minority class. The resulting data size is 20082 rows and 26 columns.

3.9.2 Grid Search on ML Algorithms on Under-Sampled Data

This process is the same as mentioned in the above section 3.9.2 in chapter 3. But the only difference is this time, under-sampled data will be passed to the machine learning models. Here also, only KNN, Decision tree, and Random forest algorithms are performed with grid search and the best parameters will be identified.

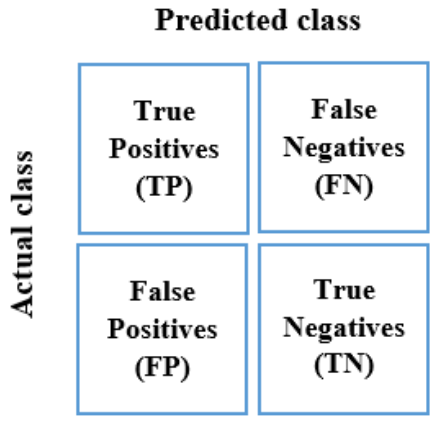
3.9.3 Model Building with Best Params of Algorithms on Under-Sampled Data

The model-building procedure with the best parameters is the same, but the only difference is that under-sampled data will be fed to the respective algorithms and after training and predictions are done, the performance will be measured and using accuracy on training, validation, and test data.

3.10 Performance Metrics

3.10.1 Confusion Matrix

Measuring the effectiveness of model performance is crucial for any machine-learning algorithm or in any other field. Below is the image of the confusion matrix.



Measure	formula
Accuracy	(TP+TN)/(TP+FP+FN+TN)
Precision	TP/ (TP+FP)
Recall	TP/ (TP+FN)
F-Measure	2*Precision*Recall/ (Precision+Recall)

Fig 9: Confusion matrix and performance metrics
[Source: (Sossi Alaoui, Farhaoui, & Aksasse, 2018)]

The confusion matrix is divided into four quadrants:

True positives (TP): These are the instances that were correctly classified as positive.
False positives (FP): These are the instances that were incorrectly classified as positive.
True negatives (TN): These are the instances that were correctly classified as negative.
False negatives (FN): These are the instances that were incorrectly classified as negative.

The confusion matrix can be used to evaluate the performance of a classification model. The model can be improved by reducing the number of false positives and false negatives.

3.10.2 Performance measures

Here are some of the metrics that can be calculated from the confusion matrix:

Accuracy: "From all the classes (positive and negative), how many of them we have predicted correctly" (Narkhede, 2018).

Precision: "From all the classes we have predicted as positive, how many are actually positive." (Narkhede, 2018).

Recall: "From all the positive classes, how many we predicted correctly." (Narkhede, 2018).

F1 score: The F1 score is a weighted average of precision and recall.

In this project, only Accuracy is considered for the comparison of the performances.

3.11 Overall Workflow

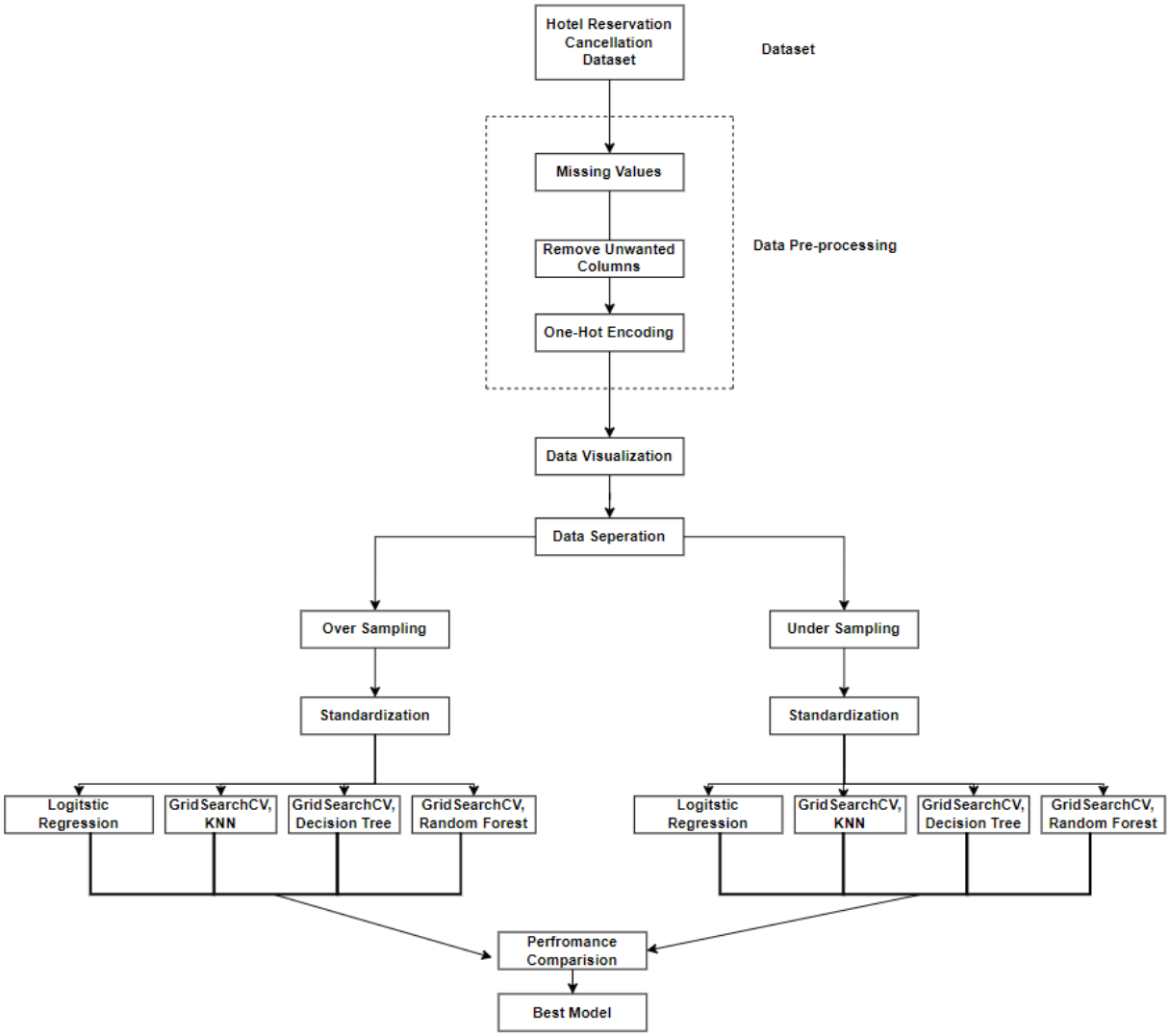


Fig 10: Overall workflow of this project.
[Source: designed in draw.io website]

3.12 Overall Summary

In Chapter 3, the detailed methodology of this research project is demonstrated starting with the data understanding followed by data pre-processing, data visualization, and data separation. In the next step, the over-sampling and under-sampling techniques are discussed individually as machine learning algorithms are fed with over-sampled data, and under-sampled data along with respective best parameters obtained from the grid search separately are discussed. Performance metrics are also discussed along with the overall workflow of the project. In the next chapter, the results will be discussed in detail.

CHAPTER 4: RESULTS AND DISCUSSION

4.1 Experimentation setup

In this project, I have done research on the hotel reservation cancellation dataset and developed machine learning algorithms on oversampled and under-sampled data separately to answer the research questions and hypothesis as well. All this work is done in Python Jupyter Notebook. Below are the details of the experimentation setup for this project.

System Specifications	Configuration Details
OS	64-bit Windows 11 OS
RAM	8 GB
Processor	11th Gen Intel(R) Core(TM) i9-11900H
Clock Speed	up to 3.90 GHz
Python	3.10
Python Libraries	NumPy, Pandas, Matplotlib, Librosa, sklearn, imlearn

Table 2: Experimentation Setup for this Research

4.2 Checking for missing data

As mentioned in the previous chapter, *DataFrame.Columns.isnull()* function and *DataFrame.info()* will give a clear picture of the null values available in the data. In this research, after running the code line ***d.info()*** where ***d*** is the data frame with stored data of the hotel reservation cancellation dataset. All the columns in the data frame consist of 36275 rows without any missing values as shown in Fig 12.

Whereas to verify again for null values in every column, ***d.columns.isnull()*** is executed with the following results shown in Fig 11.

```
[30]: # Check the misssing data from the dataset in all the columns,
      # if there are any null values, isnull() will show that as TRUE.
      # If there are no null values, it will show false.
      d.columns.isnull()

[30]: array([False, False, False, False, False, False, False, False, False,
             False, False, False, False, False, False, False, False, False,
             False])
```

Fig 11: Missing values check using *d.columns.isnull()*

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36275 entries, 0 to 36274
Data columns (total 19 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Booking_ID                            36275 non-null  object
 1   no_of_adults                          36275 non-null  int64
 2   no_of_children                        36275 non-null  int64
 3   no_of_weekend_nights                  36275 non-null  int64
 4   no_of_week_nights                     36275 non-null  int64
 5   type_of_meal_plan                     36275 non-null  object
 6   required_car_parking_space            36275 non-null  int64
 7   room_type_reserved                    36275 non-null  object
 8   lead_time                             36275 non-null  int64
 9   arrival_year                          36275 non-null  int64
 10  arrival_month                         36275 non-null  int64
 11  arrival_date                          36275 non-null  int64
 12  market_segment_type                   36275 non-null  object
 13  repeated_guest                        36275 non-null  int64
 14  no_of_previous_cancellations          36275 non-null  int64
 15  no_of_previous_bookings_not_canceled  36275 non-null  int64
 16  avg_price_per_room                    36275 non-null  float64
 17  no_of_special_requests                36275 non-null  int64
 18  booking_status                        36275 non-null  object
dtypes: float64(1), int64(13), object(5)
memory usage: 5.3+ MB
```

Fig 12: Missing values check using *d.info()*

There are no missing values in the dataset.

4.3 Data Separation Results

After data pre-processing and one-hot encoding, the data frame consists of 36275 rows and 27 columns. This dataset is split two times. Initially, data is separated into 85% train data with (30833, 27) and 15% into validation data with (5442, 27). Secondly, validation is split into two parts i.e., validation (80% of (5442, 27)) and test data (20% of (5442, 27)). Overall, below is the structure of the data before applying sampling techniques.

Train data: (30833, 26)
Train target: (30833,)
Validation data: (4353, 26)
Validation target: (4353,)
Test data: (1089, 26)
Test target: (1089,)

4.4 Data Visualization Results

Data visualization is important for this project to observe the data behavior. Firstly, the bar graphs are plotted for the object-type columns which are categorical as shown in the below figure.

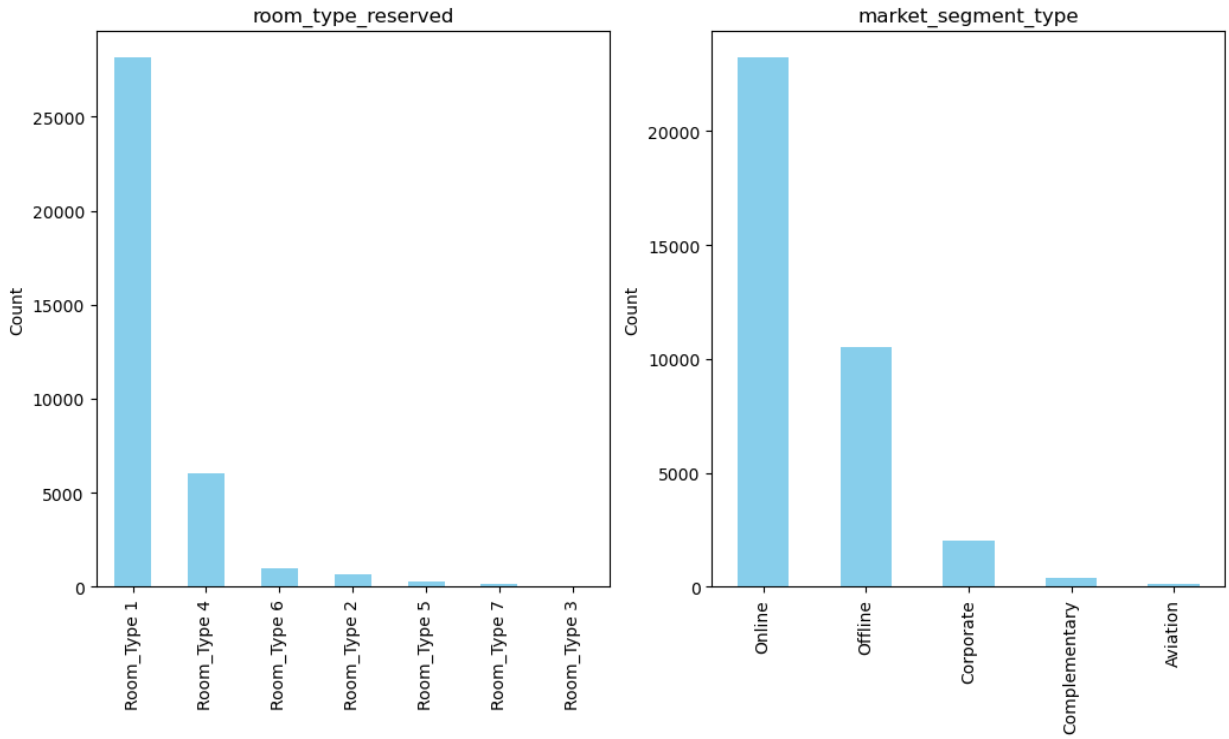


Fig 13: Count of categories in *room_type_reserved, market_segement_type* columns.

In Fig 13, it is observed that room type 1 is the most chosen room 25000 people followed by room type 4 as the second preferred by nearly 6000 people. The remaining room types are the very least chosen.

In the second graph, the market segment type through which the customer has made the booking. It is observed that online and offline booking are more with online bookings over 25000 and 10000 offline bookings followed by bookings through the corporate segment with nearly 2000.

Overall, it is identified that most customers have chosen to book online and offline with room type 1 as most preferred.

It is important to know, how many people have canceled and not canceled the reservation with respect to each room type. Fig 14 has shown that, in room_type_1 total of 9072 bookings were cancelled and 19058 bookings were not cancelled. Whereas 2069 people preferred to cancel the reservation who booked room type 4 and 3988 people decided to stay.

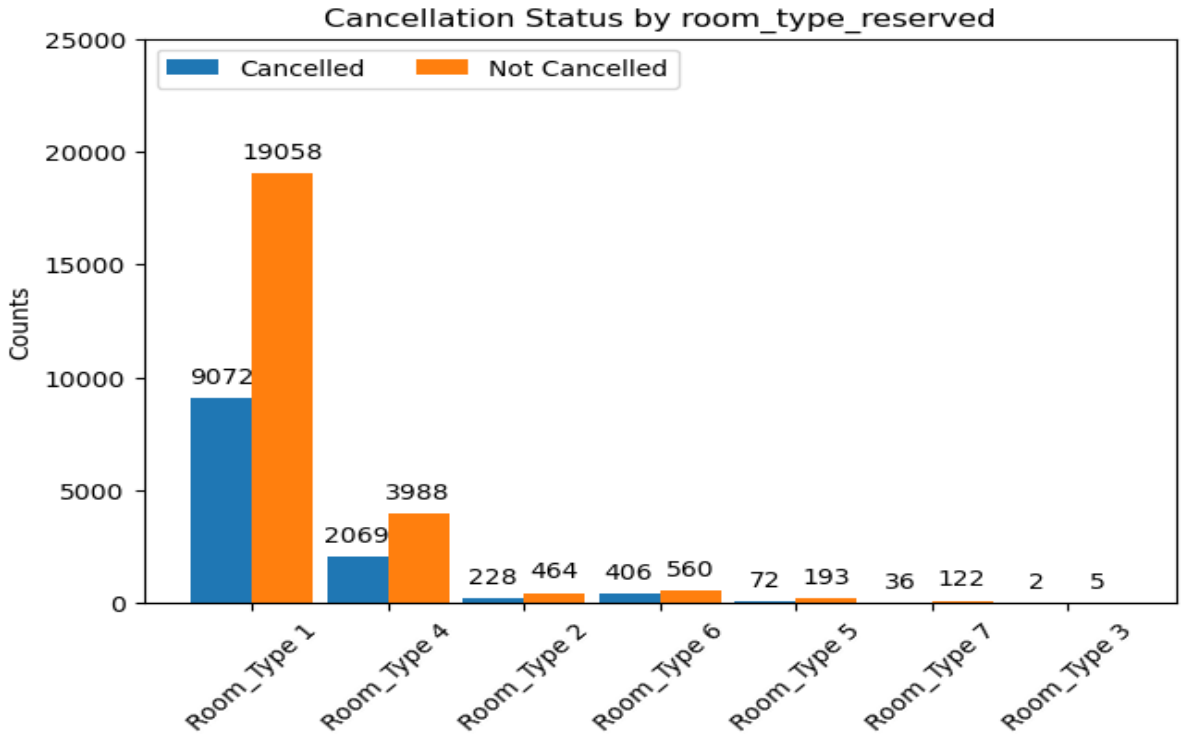


Fig 14: Cancellation Status with respect to room type reserved.

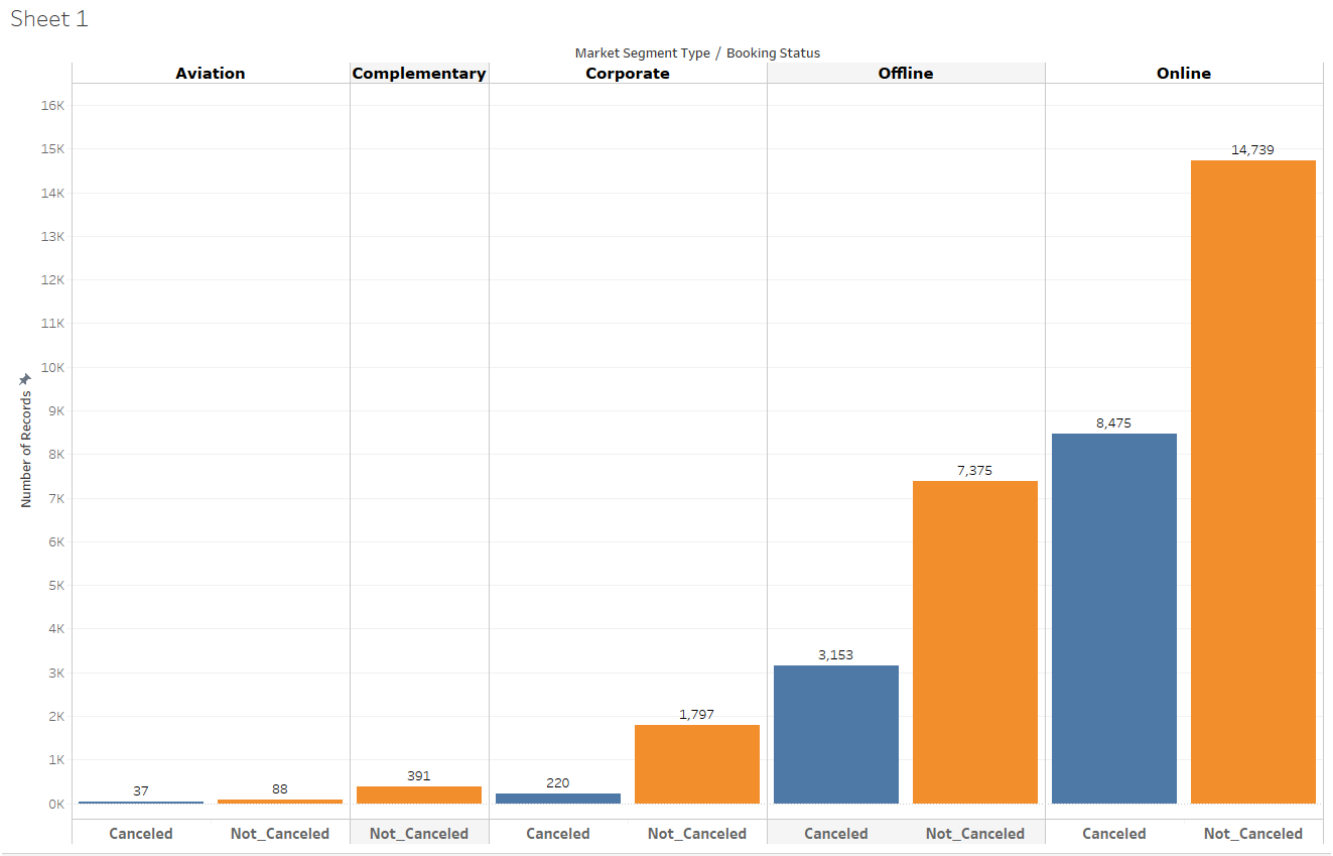


Fig 15: Cancellation Status with respect to market type segment.

Whereas with respect to the market type segment, 8475 people have canceled the online reservation which is nearly 50% compared to 14739 people who didn't cancel the reservation made online. Offline cancellations are nearly 30% with 3153 and 70% of reservations made through the offline segment are not cancelled as shown in Fig 15.

It is observed that nearly 25% of people who booked room_type_1 online have preferred to cancel the reservation made in the hotel.

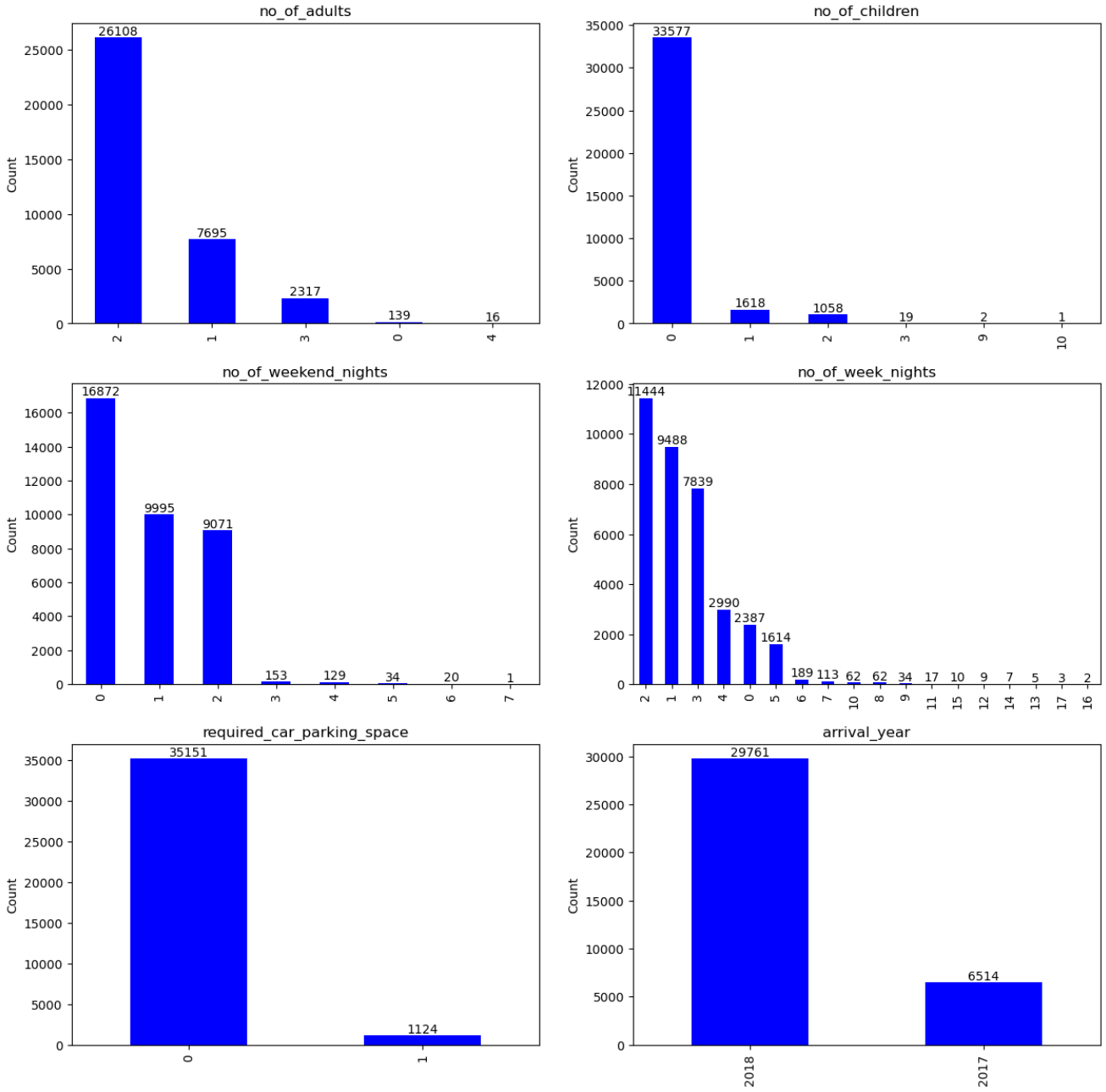


Fig 16: bar plots columns with int datatype – part1

From the above Fig 16, it is clearly shown that most people have opted for no car parking space required and many of them are adults with no children they also prefer to stay 1 to 3 weeknights on average, and some are preferred to stay 1 to 2 weekend nights as well.

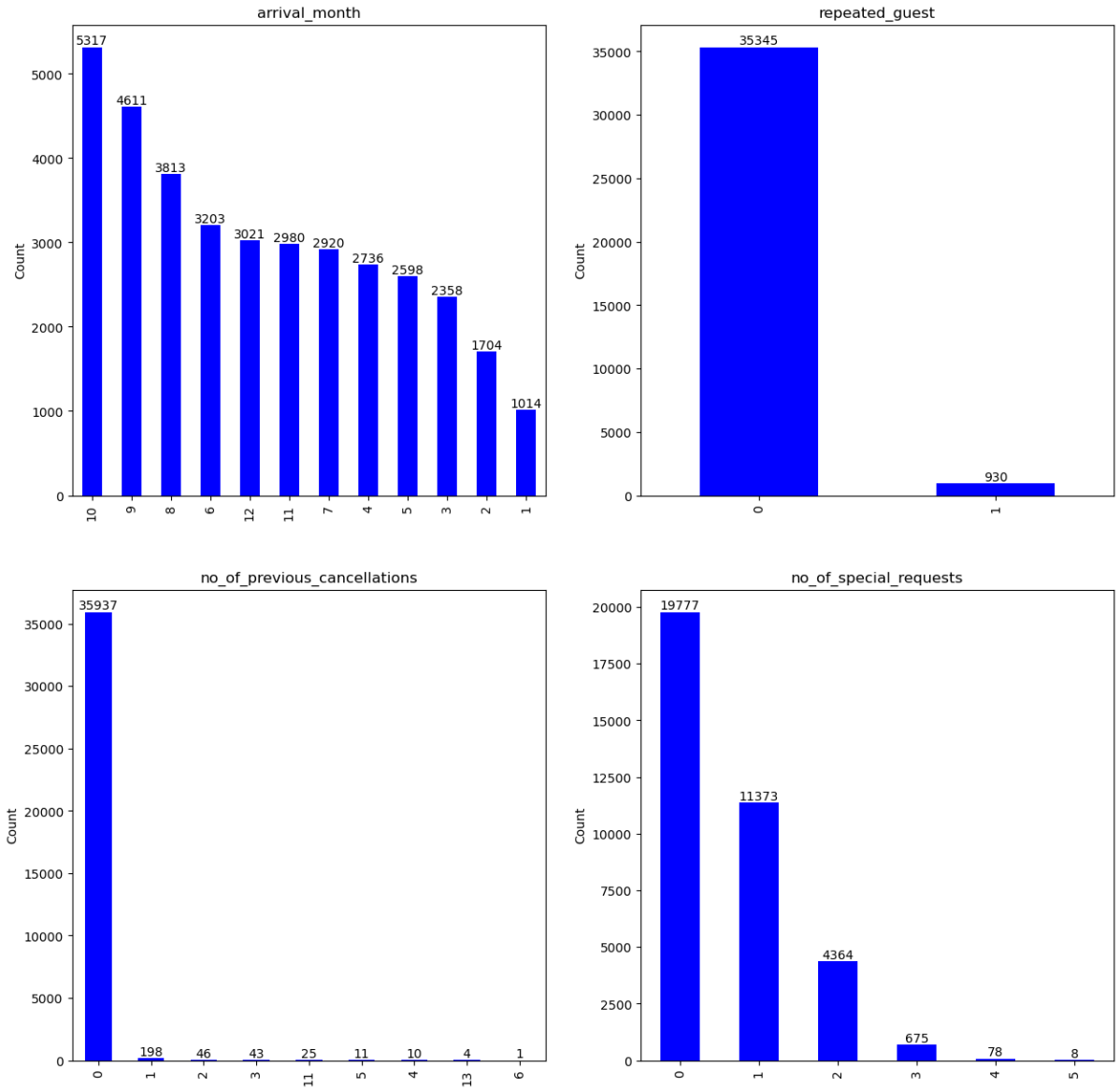


Fig 17: bar plots columns with int datatype – part2

Many people have booked cancellations for almost all the months, but peak-time bookings are between August, September, and October. Almost 97% of the people are not repeat guests and don't have any previous cancellations. And also nearly 15000 have made the special requests at the time of booking.

There are two more important columns that need to be analyzed as I personally feel, they might be the reason to cancel the reservation. i.e. Average price per room and lead time columns.

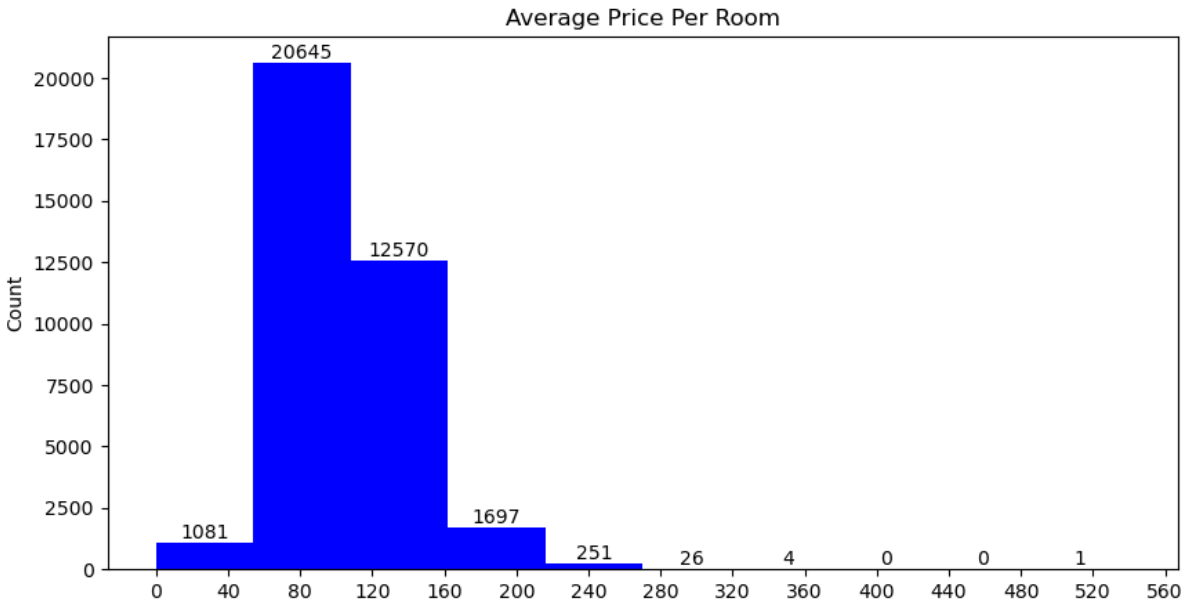


Fig 18: Histogram of Average price per room column

Fig 18 represents the price ranges on the x-axis and the number of records falling into those average price ranges. There are 20645 rows that are falling in the price range between 60 to 100 and 12570 people have booked the room with much higher prices ranging between 101 to 160. The least price range of a room is from 0-59.

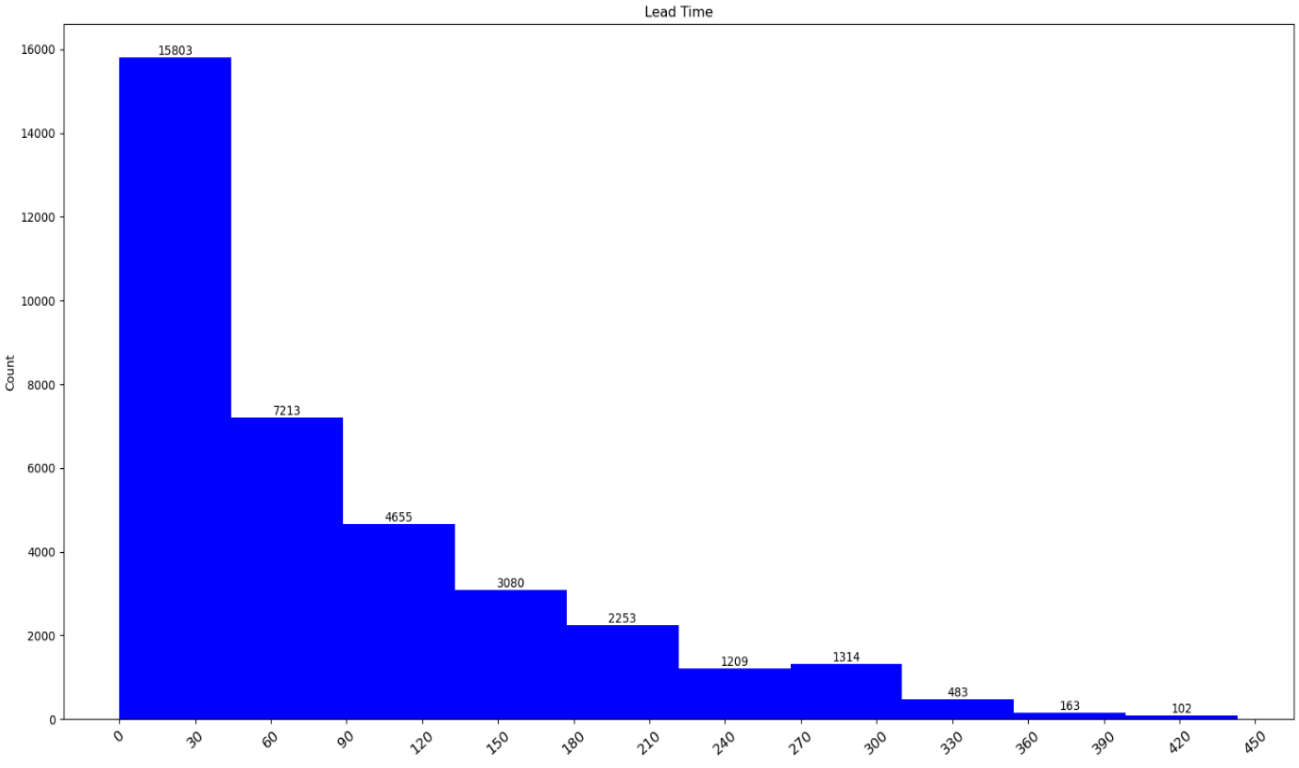


Fig 19: Histogram of Lead Time column

There are only, 15803 rows which equivalent to 43% of the records are falling in the range of 0 to 50 days as lead time. Whereas 19% of the data accompanies to fall in the range of 50 to 90 days, this seems an acceptable range as people try to plan it before 3 months. But the remaining 38% of the data falls between the range 90 to 450 days which is very long and there is a high chance of cancellations due to a longer gap between the reservation time and arrival time.

The target variable is also available with two classes as shown in the figure below with a huge imbalance in the data.

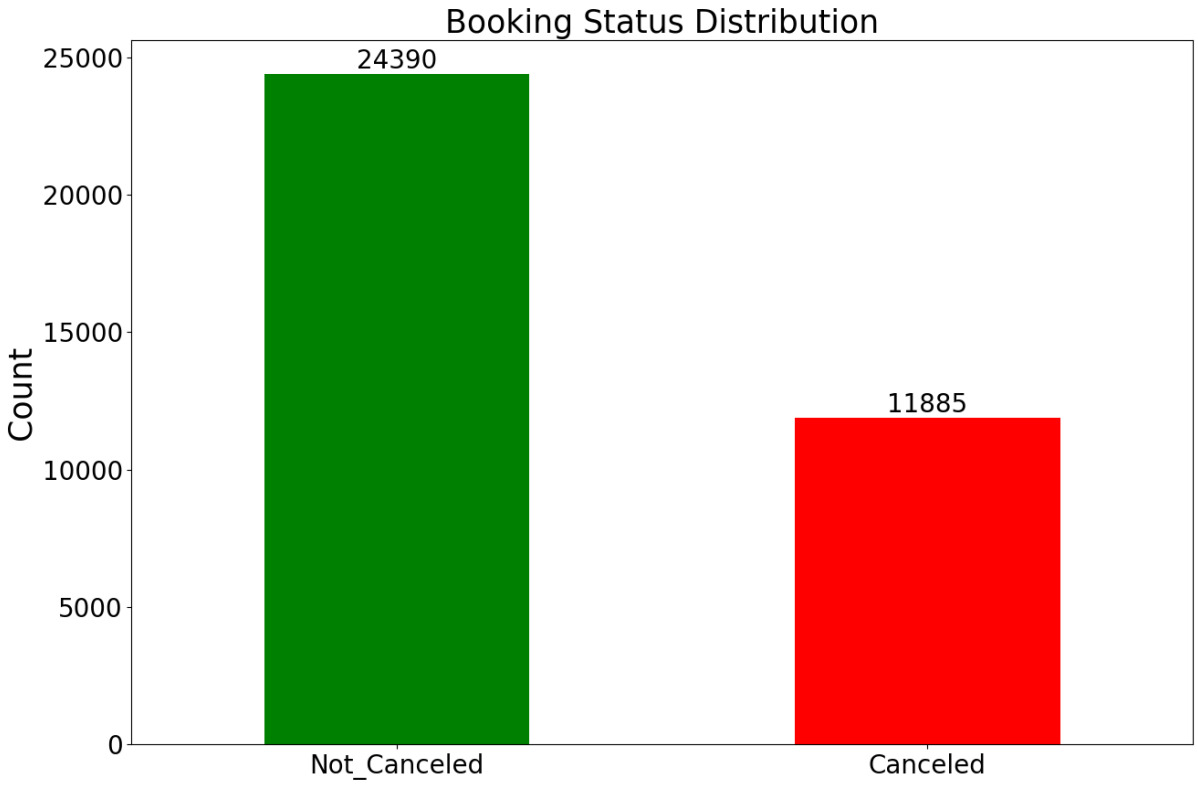


Fig 20: Counts of the Booking Status

In the booking status column, there are 24390 records that are Not Cancelled, and 11885 rows are canceled rows. The dataset is quite imbalanced, the sampled techniques have already been discussed to over this problem.

This is the overall interpretation of the visualizations obtained during the experiment.

4.5 Machine Algorithms Results on Over-Sampled Data

After separating the data, the training data is passed to the Oversampling technique to generate the data in between the records using the SMOTE function. The operation of SMOTE is already explained clearly in the previous chapter. After passing the training data with 30833 rows and 26 columns as independent variables and the target variable with 30833 rows, the SMOTE

generated (41584,26) samples for independent data and (41584) samples for the target variable respectively for the minority class and balanced the data.

This is sent to standardization and the resulting shape is the same but with the values with mean 0 and standard deviation 1 for each column. This data is fed to the grid search and machine learning models.

4.5.1 Logistic Regression

After passing the standardized over-sampled data to fit the defined logistic regression model, 79.3% of the training accuracy was achieved and the validation accuracy was achieved at 78.15% with the confusion matrix below:

```
Confusion Matrix:
 [[1090  374]
 [ 577 2312]]
```

The classification report for the validation data is as follows:

	precision	recall	f1-score	support
Canceled	0.65	0.74	0.70	1464
Not_Canceled	0.86	0.80	0.83	2889
accuracy			0.78	4353
macro avg	0.76	0.77	0.76	4353
weighted avg	0.79	0.78	0.78	4353

Table 3: Classification report for logistic regression with oversampled data

On the test data, this logistic regression model with oversampled data has achieved 77.68%.

4.5.2 K-Nearest Neighbours

The KNN is passed with the below parameters,

param_grid = {

'n_neighbors': [3, 5, 7, 9, 11],

'weights': ['uniform', 'distance'],

'algorithm': ['ball_tree', 'kd_tree', 'brute']

}

These parameters as passed to grid search with the KNN algorithm have identified the below parameters as the best.
{'algorithm': 'brute', 'n_neighbors': 11, 'weights': 'distance'}

After passing the best hyper-parameters obtained from the grid search, the KNN model with over-sampled data has achieved 99.38% training accuracy and 86.72% validation accuracy.

Below is the confusion matrix and classification report for the validation data of KNN.

```
Confusion Matrix:
 [[1202  262]
 [ 300 2589]]
```

	precision	recall	f1-score	support
Canceled	0.80	0.82	0.81	1464
Not_Canceled	0.91	0.90	0.90	2889
accuracy			0.87	4353
macro avg	0.85	0.86	0.86	4353
weighted avg	0.87	0.87	0.87	4353

Table 4: Classification report for KNN with oversampled data

On testing the model on unseen data, it resulted in 87.87% accuracy.

4.5.3 Decision Tree

Below are the hyperparameters passed to the decision tree grid-search model with oversampled data.

dt_params = {

'criterion': ['gini', 'entropy'],

'max_depth': [None, 5, 10, 15],

'min_samples_split': [2, 5, 10],

'min_samples_leaf': [1, 2, 5]

}

The decision tree's best parameters are *{'criterion': 'entropy','max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2}*

After training with the best params, the decision tree model has achieved 99.46% of training accuracy and 87.08% of validation accuracy. Below is the confusion matrix and classification report for the Decision Tree model on over-sampled data.

```
Confusion Matrix:
 [[1202  262]
 [ 300 2589]]
```

	precision	recall	f1-score	support
Canceled	0.80	0.82	0.81	1464
Not_Canceled	0.91	0.90	0.90	2889
accuracy			0.87	4353
macro avg	0.85	0.86	0.86	4353
weighted avg	0.87	0.87	0.87	4353

Table 5: Classification report for Decision Tree with oversampled data

The unseen test data accuracy achieved by this model is 88.33%

4.5.4 Random Forest

Random Forest is the most powerful algorithm in machine learning. The operation is mentioned in previous chapters. The hyper-parameters considered for the grid search is as follows:

rf_params = {

'n_estimators': [100, 300, 500],

'criterion': ['gini', 'entropy'],

'max_depth': [None, 5, 10, 15],

'min_samples_split': [2, 5, 10],

'min_samples_leaf': [1, 2, 5]

}

Below are the results after passing the best parameters obtained to the random forest model using oversampled data are:

Training Accuracy: 99.46%

Validation Accuracy: 90.55%

Below is the confusion matrix and classification report.

```
Confusion Matrix:
 [[1239  225]
 [ 186 2703]]
```

	precision	recall	f1-score	support
Canceled	0.87	0.85	0.86	1464
Not_Canceled	0.92	0.94	0.93	2889
accuracy			0.91	4353
macro avg	0.90	0.89	0.89	4353
Weighted avg	0.91	0.91	0.91	4353

Table 6: Classification report for Random Forest with oversampled data

On unseen test data, the random forest has achieved 89.99% accuracy.

4.5.4.1 Feature Importance Results

As mentioned in Chapter 2 under section 2.6.4.1, the feature importance will be obtained from the random forest. Below is the image generated by Random Forest for the feature importance of the columns based on the most used column for decision-making.

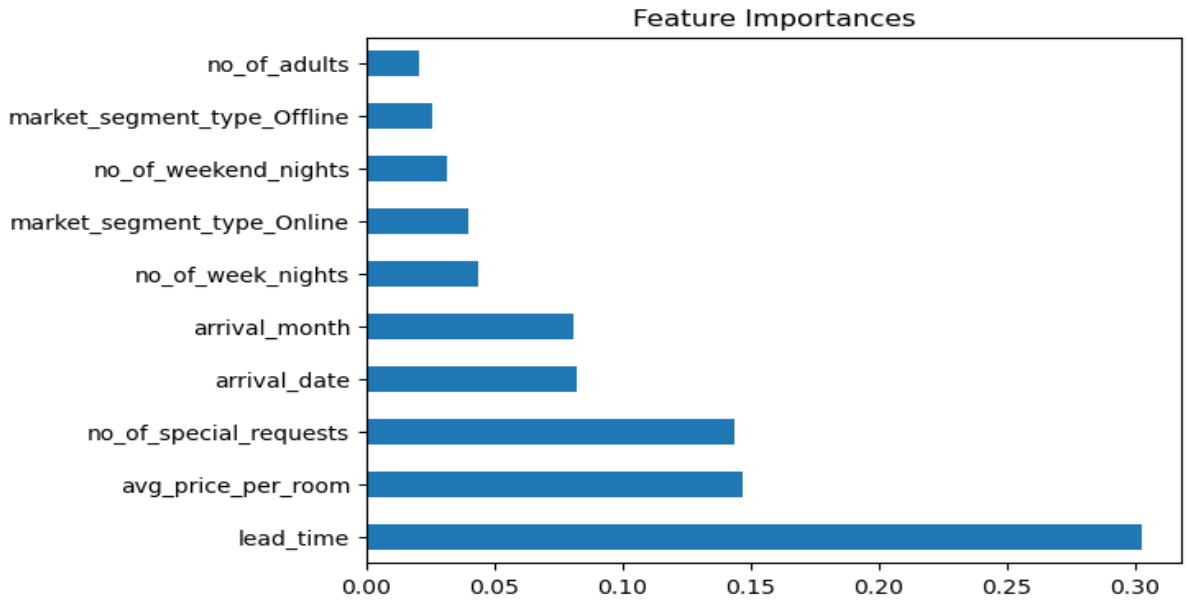


Fig 21: Random Forest Feature Importance on Oversampled Data.

As expected, lead time and average price per room columns are the top two columns that are very important in predicting hotel reservation cancellation. Hence, research question 4 is answered with the feature importance.

4.5.5 Combined Results of All Algorithms on Over-Sampled Data

Below are the combined results of all the algorithms on the oversampled data with the best parameters obtained from the grid search.

	Training Accuracy (OverSampled)	Validation Accuracy (OverSampled)	Testing Accuracy (OverSampled)
Logistic Regression	79.30	78.15	77.68
K-Nearest Neighbours	99.38	86.72	87.87
Decision Tree	99.46	87.08	88.33
Random Forest	99.46	90.55	89.99

Fig 22: Combined Results of algorithms on over-sampled data

4.6 Machine Algorithms Results on Under-Sampled Data

After separating the data, the training data is passed to the Under-sampling technique to reduce the data randomly in between the records of the majority class using the RUS function. After passing the training data with 30833 rows and 26 columns as independent variables and the target variable with 30833 rows, the RUS generated (20082,26) samples for independent data and (20082) samples for the target variable respectively.

This is sent to standardization and the resulting shape is the same This data is fed to the grid search and machine learning models.

4.6.1 Logistic Regression

The operation is the same as performed with oversampled data, but here only difference is that under-sampled data is passed to the logistic regression model and evaluated. Below are the performance results.

Training Accuracy: 77.79 %

Validation Accuracy: 77.92%

```
Confusion Matrix:
 [[1107  357]
 [ 604 2285]]
```

Classification Report:

	precision	recall	f1-score	support
Canceled	0.65	0.76	0.70	1464
Not_Canceled	0.86	0.79	0.83	2889
accuracy			0.78	4353
macro avg	0.76	0.77	0.76	4353
weighted avg	0.79	0.78	0.78	4353

Table 7: Classification report for Logistic Regression with under-sampled data

The testing accuracy achieved on unseen data is 78.23%

4.6.2 K-Nearest Neighbours

The same parameters are passed to KNN as in the oversampled section, the only difference is that under-sampled data is passed this time. Below are the results of the KNN on under-sampled data.

Training Accuracy: 99.26%

Validation Accuracy: 84.58%

Confusion Matrix:
[[1259 205]
[466 2423]]

Below is the classification report for KNN with under-sampled data:

	precision	recall	f1-score	support
Canceled	0.73	0.86	0.79	1464
Not_Canceled	0.92	0.84	0.88	2889
accuracy			0.85	4353
macro avg	0.83	0.85	0.83	4353
weighted avg	0.86	0.85	0.85	4353

Table 8: Classification report for KNN with under-sampled data

The Testing Accuracy is 84.38% on unseen data for the KNN model trained on under-sampled data.

4.6.3 Decision Tree

The decision tree is also passed with under-sampled data in this section, below are the performance results of the Decision Tree model.

Training Accuracy: 91.46 %

Validation Accuracy: 85.38%

Confusion Matrix:
[[1275 189]
[447 2442]]

	precision	recall	f1-score	support
Canceled	0.74	0.87	0.80	1464
Not_Canceled	0.93	0.85	0.88	2889
accuracy			0.85	4353
macro avg	0.83	0.86	0.84	4353
weighted avg	0.87	0.85	0.86	4353

Table 9: Classification report for Decision Tree with under-sampled data

The test accuracy of the decision tree model is 85.21%.

4.6.4 Random Forest

The random forest achieved the below results on training with the under-sampled data.

Training Accuracy: 97.92%

Testing Accuracy: 89.34%

```
Confusion Matrix:
 [[1286  178]
 [ 286 2603]]
```

	precision	recall	f1-score	support
Canceled	0.82	0.88	0.85	1464
Not_Canceled	0.94	0.90	0.92	2889
accuracy			0.89	4353
macro avg	0.88	0.89	0.88	4353
weighted avg	0.90	0.89	0.89	4353

Table 10: Classification report for random forest with under-sampled data

The trained random forest on under-sampled data has achieved 89.25% on test data.

4.6.4.1 Feature Importance Results

Below is the image of the feature importance generated by the random forest.

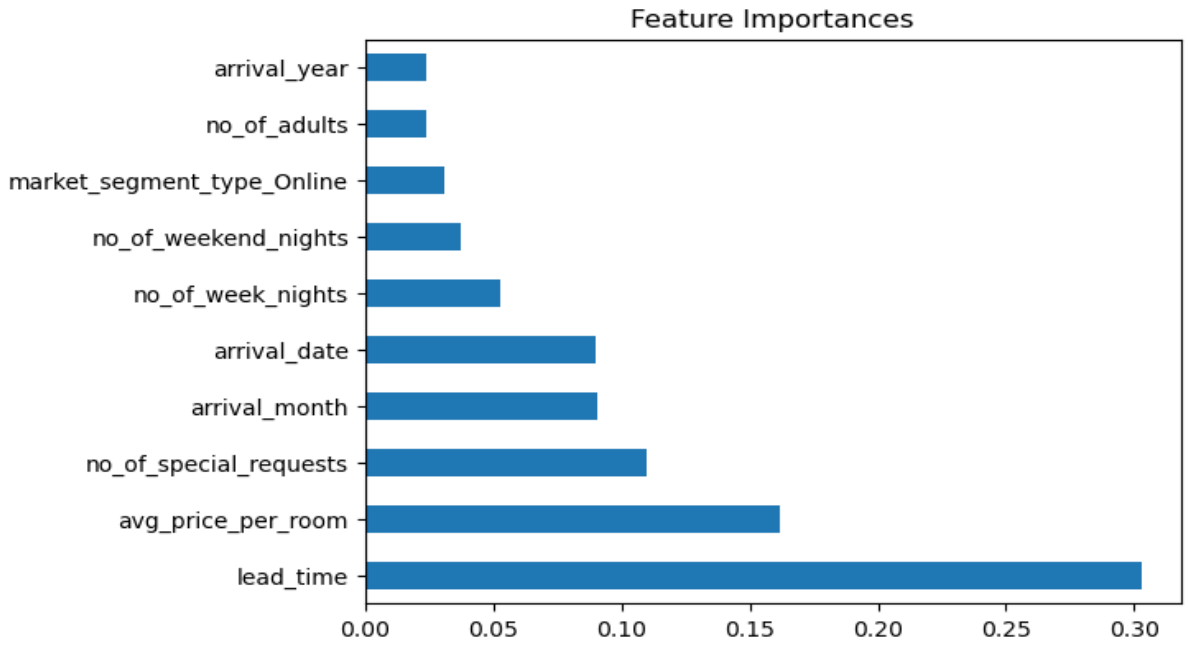


Fig 23: Random Forest Feature Importance on under-sampled Data.

If we observe, the most important features are lead time and the average price per room same as in oversampled data. However, the remaining features have changed their order in under-sampled feature importance. The answer to research question 4 remains the same.

4.6.5 Combined Results of All Algorithms on Under-Sampled Data

Below are the combined results of the algorithms performed using under-sampled data.

	Training Accuracy (UnderSampled)	Validation Accuracy (UnderSampled)	Testing Accuracy (UnderSampled)
Logistic Regression	77.79	77.92	78.23
K-Nearest Neighbours	99.26	84.58	84.38
Decision Tree	91.46	85.38	85.21
Random Forest	97.92	89.34	89.25

Fig 24: Combined Results of algorithms on under-sampled data

4.7 Comparison of Algorithms Results on Over-Sampled and Under-Sampled Data

Below are the results tabulated for training, validation, and testing accuracy on oversampled and under-sampled data for all the machine learning algorithms used.

	Training Accuracy (Over Sampled)	**Training Accuracy (Under Sampled)**
Logistic Regression	***79.30***	***77.79***
K-Nearest Neighbours	***99.38***	***99.26***
Decision Trees	***99.46***	***91.46***
Random Forest	***99.46***	***97.92***

Table 11: Training accuracy comparison of algorithms on oversampled and under-sampled data

From the above table, it is observed that, on oversampled data, the decision tree model and random forest have trained equally with the highest accuracy compared to other algorithms. In under-sampled training, performances are very low comparatively.

	Validation Accuracy (Over Sampled)	**Validation Accuracy (Under Sampled)**
Logistic Regression	***78.15***	***77.92***
K-Nearest Neighbours	***86.72***	***84.58***
Decision Trees	***87.08***	***85.38***
Random Forest	***90.55***	***89.34***

Table 12: Validation accuracy comparison of algorithms on oversampled and under-sampled data

On validation data, clearly, the winner is the random forest with 90.55 % on oversampled and 89.34% on under-sampled data. Random forest algorithm trained on oversampled data has given the highest accuracy on the validation data.

	Testing Accuracy (Over Sampled)	**Testing Accuracy (Under Sampled)**
Logistic Regression	***77.68***	***78.23***
K-Nearest Neighbours	***87.87***	***84.38***
Decision Trees	***88.33***	***85.21***
Random Forest	***89.99***	***89.25***

Table 13: Testing accuracy comparison of algorithms on oversampled and under-sampled data

On comparing the performances of various machine learning algorithms on oversampled and under-sampled data, it is observed that Random Forest has performed the best in both cases. But Random Forest on Oversampled data has given the training accuracy of 99.46% Validation accuracy of 90.55% and Testing Accuracy is 89.99%.

All the algorithms trained on the oversampled data have achieved better performance on validation and test data compared to the algorithms trained on the under-sampled data. Hence, the "**Null Hypothesis has been rejected".**

CHAPTER 5: CONCLUSION AND FUTURE WORK

CONCLUSION

In this research, I have developed machine-learning algorithms on oversampled and under-sampled data separately to predict the customer who might cancel the reservation based on the customer details provided and the best performance model has been chosen on the performance comparison of models on both the sampled data individually and also, the data has been analyzed through the data visualizations to identify the possible reasons behind the cancellations. The Data understanding is crucial for this project as an initial step to understand the significance of each column. In the pre-processing steps, checked for the missing values and none of them were found. But there are a few columns that might not make any impact on the data, those two columns have been removed namely, the ID and the type of meal plan. There are 3 categorical columns, two of them are independent variables and one is a target variable, the one-hot encoding is applied to the independent variables namely "room_type_reserved" and "market_segment_type" which created 12 columns for the categories in it. In the next step, data visualizations are created to analyze all the columns of the data. Cancellation status by room type reserved, and market segment type are analyzed and the average price per room and lead time are plotted with histogram and some important findings are observed and noted in the results section. Counts of booking status have shown great imbalance in the dataset, hence, to deal with it, data is separated into the train, validation, and test sets in the data separation section. Later the sampling techniques operation is understood in the literature review, and they have been implemented separately for all the algorithms on both over-sampled and under-sampled data. Applying the SMOTE technique for oversampling the actual training data for the minority class, resulted in 41584 rows and 26 columns for independent data and 41584 rows for target variables, later This data was standardized and used to train the machine learning models with the best parameters obtained from the grid search on every algorithm namely, KNN, Decision Tree and Random Forest and the results are tabulated. Similarly, the RUS technique for under-sampling the actual training data to reduce the majority class, which resulted in the independent data with shape (20082,26) and (20082,) for the target variable. Under-sampled data is standardized and fed to the machine-learning algorithms and results are tabulated and analyzed, below are the observations made from the data visualizations and from the model performances on sampled data.

From the data visualizations, it is observed that 38% of the lead time column data is falling between the gap range of 90 to 450 days which can be the important reason for cancellations people might get other plans to do or change their mind and also average price per room shows the people mostly preferred to book the rooms with price between 60 to 100 euros. Also, peak times for most bookings are August, September, and October with most of them adults with no children preferring to stay 1 or 2 weekend nights or 1 to 3 weeknights. Room type 1 is the most chosen with 25% of cancellations, and most people preferred to book online, people who booked online have taken advantage of canceling them, and it accounts for 25% as well.

The machine algorithms trained and tested on oversampled data have achieved the maximum performance on the random forest algorithm with 99.46% training accuracy, 90.55% validation accuracy, and 89.99% test accuracy on unseen data. The decision tree stands in the second

position with 99.46%, 87.08%, and 88.33% for train, validation, and test accuracy. KNN stands 3rd and logistic regression is the base model with 77.68% accuracy on test data.

On under-sampling data, Random Forest had given the highest training, validation, and testing accuracy with 97.2%, 89.34%, and 89.25%. Similar to oversampled results, the decision tree stands in second position with 85.21% on test data, KNN stands 3rd with 84.38% on test data and the logistic regression model is the least performant model with only 78.23%.

In both the cases of under-sampled and over-sampled, Random feature importance has resulted in 'lead time" and "average price per room" as are most important columns in predicting the hotel reservation cancellation of the customer provided the details. All the research questions have been answered with the help of a literature review and experimentation results.

On a conclusion note, "the null hypothesis has been rejected" as machine learning algorithms trained on oversampled data have achieved the highest performance results compared to the model's performance on under-sampled data. The best machine learning model for predicting hotel reservation cancellation is the random forest trained on over-sampled data with a 90% confidence rate on unseen data.

FUTURE WORK

There are many algorithms, like neural networks, support vector machines, K-means clustering, etc., that can be used to experiment on this hotel reservation data. much advanced architectures like artificial neural networks with learning parameters and loss functions can give more generalized performance and better accuracy. Also, alternative approaches to deal with imbalanced data have to be explored. These are things I have observed to consider for future work.

REFERENCES

Aldraimli, M., Soria, D., Parkinson, J., Thomas, E.L., Bell, J.D., Dwek, M.V. and Chaussalet, T.J. (2020). Machine learning prediction of susceptibility to visceral fat associated diseases. Health and Technology, 10(4), pp.925–944. doi:https://doi.org/10.1007/s12553-020-00446-1.

Antonio, N., de Almeida, A. and Nunes, L. (2017). Predicting Hotel Bookings Cancellation with a Machine Learning Classification Model. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. doi:https://doi.org/10.1109/icmla.2017.00-11.

Bhandari, A. (2020). Feature Scaling | Standardization Vs Normalization. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/.

Bhandari, P. (2022). Missing Data | Types, Explanation, & Imputation. [online] Scribbr. Available at: https://www.scribbr.co.uk/stats/missing-values/.

Bonthu, H. (2021). An Introduction to Logistic Regression. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/07/an-introduction-to-logistic-regression/.

Brownlee, J. (2017). Why One-Hot Encode Data in Machine Learning? [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/.

Chawla, N.V., Bowyer, K.W., Hall, L.O. and Kegelmeyer, W.P. (2002). SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research, 16(16), pp.321–357. doi:https://doi.org/10.1613/jair.953.

Data Analytics. (2020). Feature Importance using Random Forest Classifier - Python. [online] Available at: https://vitalflux.com/feature-importance-random-forest-classifier-python/.

E R, S. (2021). Random Forest | Introduction to Random Forest Algorithm. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/.

Hasanin, T. and Khoshgoftaar, T. (2018). The Effects of Random Undersampling with Simulated Class Imbalance for Big Data. [online] IEEE Xplore. doi:https://doi.org/10.1109/IRI.2018.00018.

He, Y.F., Wen, P.P., Lan, Y.Q. and Miao, Z.W. (2018). Hotel Cancellation Strategies Under Online Advanced Booking. [online] IEEE Xplore. doi:https://doi.org/10.1109/IEEM.2018.8607679.

IBM (2022). What is a Decision Tree | IBM. [online] www.ibm.com. Available at: https://www.ibm.com/topics/decision-trees.

Narkhede, S. (2018). Understanding Confusion Matrix. [online] Medium. Available at: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62.

Nurul Liyana Hairuddin, Lizawati Mi Yusuf and Mohd Shahizan Othman (2020). GENDER CLASSIFICATION ON SKELETAL REMAINS: EFFICIENCY OF METAHEURISTIC ALGORITHM METHOD AND OPTIMIZED BACK PROPAGATION NEURAL NETWORK. Journal of ICT, 19. doi:https://doi.org/10.32890/jict2020.19.2.5.

Romero Morales, D. and Wang, J. (2010). Forecasting cancellation rates for services booking revenue management using data mining. European Journal of Operational Research, 202(2), pp.554–562. doi:https://doi.org/10.1016/j.ejor.2009.06.006.

s, A. (2021). The Magic of Grid Search Cross Validation. [online] Medium. Available at: https://medium.com/@abhivarma362/the-magic-of-grid-search-cross-validation-81fd1d4b73b9.

S, R.V. (2020). ALL ABOUT KNN ALGORITHM. [online] Medium. Available at: https://rekhavsrh.medium.com/all-about-knn-algorithm-6b35a18c2b15.

Satu, Md.S., Ahammed, K. and Abedin, M.Z. (2020). Performance Analysis of Machine Learning Techniques to Predict Hotel booking Cancellations in Hospitality Industry. 2020 23rd International Conference on Computer and Information Technology (ICCIT). doi:https://doi.org/10.1109/iccit51783.2020.9392648.

Shah, R. (2021). GridSearchCV |Tune Hyperparameters with GridSearchCV. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/.

Sharma, A., Singh, P.K. and Chandra, R. (2022). SMOTified-GAN for Class Imbalanced Pattern Classification Problems. IEEE Access, 10, pp.30655–30665. doi:https://doi.org/10.1109/access.2022.3158977.

Shirisha, N., Anusha, K., Kiran, A. and Buavani, Y.T.S. (2023). Prediction of Hotel Booking & Cancellation using Machine Learning Algorithms. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICCCI56745.2023.10128484.

Sossi Alaoui, Safae, Farhaoui, Yousef, & Aksasse, B. (2018). Classification algorithms in Data Mining. International Journal of Tomography and Simulation, 31, 34-44.

Spelmen, V.S. and Porkodi, R. (2018). A Review on Handling Imbalanced Data. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICCTCT.2018.8551020.

Taunk, K., De, S., Verma, S. and Swetapadma, A. (2019). A Brief Review of Nearest Neighbor Algorithm for Learning and Classification. [online] IEEE Xplore. doi:https://doi.org/10.1109/ICCS45141.2019.9065747.

Testprep Training Tutorials. (n.d.). Remove Unnecessary Rows and Columns. [online] Available at: https://www.testpreptraining.com/tutorial/remove-unnecessary-rows-and-columns/ [Accessed 26 Aug. 2023].

TIBCO (2023). What is a Random Forest? [online] TIBCO Software. Available at: https://www.tibco.com/reference-center/what-is-a-random-forest.

www.kaggle.com. (2023). Hotel Reservations Dataset. [online] Available at: https://www.kaggle.com/datasets/ahsan81/hotel-reservations-classification-dataset.

APPENDICES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix

import warnings
warnings.filterwarnings('ignore')
```

Data Loading

```
d = pd.read_csv("Hotel Reservations.csv")
d.head()
```

Data Preprocessing

Checking for the missing values

```
d.info()

d.columns.isnull()
```

Remove UnWanted Columns

```
d["type_of_meal_plan"].unique()
d["room_type_reserved"].unique()
d["arrival_year"].unique()
d["arrival_year"].unique()
d["market_segment_type"].unique()

d = d.drop(columns = ["Booking_ID","type_of_meal_plan"],axis=1)
d
```

One-Hot Encoding

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

one_hot_encoder = OneHotEncoder(sparse=False, drop=None

one_hot_encoded_features = one_hot_encoder.fit_transform(d[['room_type_reserved',
'market_segment_type']])

encoded_column_names = one_hot_encoder.get_feature_names_out(['room_type_reserved',
'market_segment_type'])
data_encoded = pd.DataFrame(one_hot_encoded_features,
columns=encoded_column_names)

data_encoded = pd.concat([d, data_encoded], axis=1)

data_encoded.drop(['room_type_reserved', 'market_segment_type'], axis=1, inplace=True)

data_encoded.info()

data_encoded.head()
```

Poping target column inbetween the columns and adding it to the last

```
booking_status_col = data_encoded.pop("booking_status")
data_encoded["booking_status"] = booking_status_col
data_encoded

data_encoded.info()

data_encoded["booking_status"].value_counts()
```

Data Visualization

Plotting the object columns counts using bar graph

```
object_cols = [col for col in d.columns if d[col].dtype == "object"]

fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(19, 6))

for col, ax in zip(object_cols, (ax1, ax2, ax3)):
    d[col].value_counts().plot(kind="bar", ax=ax, color="SkyBlue")
    ax.set_title(col)
    ax.set_ylabel("Count")
```

```
print("Object Columns: ",object_cols)
plt.show()
```

Create a dictionary to store the counts of booking_status for each room_type_reserved

```
counts = {}
for room_type, booking_status in zip(d["room_type_reserved"], d["booking_status"]):
    if room_type not in counts:
        counts[room_type] = {}
    if booking_status not in counts[room_type]:
        counts[room_type][booking_status] = 0
    counts[room_type][booking_status] += 1

print(counts)
# Create a NumPy array to store the counts of booking_status for each room_type_reserved
bar_data = np.array([[counts[room_type]["Canceled"],
counts[room_type]["Not_Canceled"]] for room_type in counts])
bar_keys = list(counts.keys())

print(bar_data)
print(bar_keys)
```

Grouped bar plot for cancellation status by room_type_reserved

```
room_type_reserved_col = (tuple(bar_keys))
counts = {
    'Cancelled': tuple([bar_data[i][0] for i in range(bar_data.shape[0])]),
    'Not Cancelled': tuple([bar_data[i][1] for i in range(bar_data.shape[0])]),

}

x = np.arange(len(room_type_reserved_col))  # the label locations
width = 0.45  # the width of the bars
multiplier = 0

fig, ax = plt.subplots(layout='constrained')

for attribute, measurement in counts.items():
    offset = width * multiplier
    rects = ax.bar(x + offset, measurement, width, label=attribute)
    ax.bar_label(rects, padding=5)
    multiplier += 1

    ax.set_xticks(x + width / 2)
    ax.set_xticklabels(room_type_reserved_col, rotation=45)
    ax.set_ylabel('Counts')
    ax.set_title('Cancellation Status by room_type_reserved')
```

```
    ax.legend(loc='upper left', ncols=3)
    ax.set_ylim(0, 25000)

plt.show()


non_float_cols = [col for col in d.columns if (d[col].dtype != "object") and (col not in
("lead_time",
"avg_price_per_room","no_of_previous_bookings_not_canceled","arrival_date"))]
float_cols = [col for col in d[["lead_time",
"avg_price_per_room","no_of_previous_bookings_not_canceled","arrival_date"]]]
```

Histogram for Average Price per room

```
fig, axes = plt.subplots(1, 1, figsize=(10, 5))

d["avg_price_per_room"].plot(kind="hist", ax=axes, color="blue")
axes.set_title("Average Price Per Room")
axes.set_ylabel("Count")

    # Set the x-axis limits to avoid clumbiness
axes.set_xticks(np.arange(0, axes.get_xlim()[1], 40))
axes.bar_label(axes.containers[0])

plt.show()
```

Histogram for Lead Time

```
fig, axes = plt.subplots(1, 1, figsize=(20, 10))

d["lead_time"].plot(kind="hist", ax=axes, color="blue")
axes.set_title("Lead Time")
axes.set_ylabel("Count")


axes.set_xticks(np.arange(0, axes.get_xlim()[1], 30))
plt.xticks(rotation=40,size=12)
axes.bar_label(axes.containers[0])

plt.show()
```

Plotting the int data type columns

```
fig, axes = plt.subplots(3,2 , figsize=(15, 15))
```

```
for col, ax in zip(non_float_cols[0:6], axes.flatten()):
    d[col].value_counts().plot(kind="bar", ax=ax, color="blue")
    ax.set_title(col)
    ax.set_ylabel("Count")
    ax.bar_label(ax.containers[0])

plt.show()


fig, axes = plt.subplots(2,2 , figsize=(15, 15))

for col, ax in zip(non_float_cols[6:10], axes.flatten()):
    d[col].value_counts().plot(kind="bar", ax=ax, color="blue")
    ax.set_title(col)
    ax.set_ylabel("Count")
    ax.bar_label(ax.containers[0])

plt.show()
```

Bar Plot for the target variable

```
fig, ax = plt.subplots(figsize=(15, 10))

d['booking_status'].value_counts().plot(kind="bar", ax=ax, color=["green","red"])
ax.set_title("Booking Status Distribution", size = 25)
plt.xticks(rotation=0,size=20)
ax.set_ylabel("Count", size=25)
plt.yticks(size=20)
ax.bar_label(ax.containers[0], size=20)

plt.show()
```

Data Separation

```
train_data, test_data, train_target, test_target = train_test_split(
    data_encoded.iloc[:, :-1], data_encoded.iloc[:, -1], test_size=0.15, random_state=42
)
val_data, test_data, val_target, test_target = train_test_split(
    test_data, test_target, test_size=0.20, random_state=42
)
```

 # These will be used for Undersampling later.

```
train_data_us = train_data
test_data_us = test_data
```

```
val_data_us = val_data
val_target_us = val_target
train_target_us = train_target
test_target_us = test_target
```

Split for over sampled data training

```
print("train_data.shape: ",train_data.shape)
print("train_target.shape: ",train_target.shape)
print("val_data,shape: ", val_data.shape)
print("val_target,shape: ", val_target.shape)
print("test_data.shape: ",test_data.shape)
print("test_target.shape: ",test_target.shape)

train_data.head()
```

Over Sampling

```
# Apply SMOTE for oversampling the minority class

smote = SMOTE(random_state=42)
train_data_osampled, train_target_osampled = smote.fit_resample(train_data, train_target)

train_data_osampled.shape, train_target_osampled.shape

train_data_osampled

train_target_osampled.head()
```

Standardization on oversampled data

```
scaler = StandardScaler()
train_data_osampled = scaler.fit_transform(train_data_osampled)
test_data_osampled = scaler.transform(test_data)
val_data_osampled = scaler.transform(val_data)

train_data_osampled

test_data_osampled

val_data_osampled
```

Machine Learning algorithms on Over-Sampled data

Logistic Regression

```
from sklearn.metrics import classification_report
# Logistic Regression
logistic_regression = LogisticRegression()

# Train the Logistic Regression classifier
logistic_regression.fit(train_data_osampled, train_target_osampled)

# Make predictions on the train and test data
logistic_regression_train_pred = logistic_regression.predict(train_data_osampled)
logistic_regression_test_pred = logistic_regression.predict(val_data_osampled)

# Print performance metrics for Logistic Regression
print('Logistic Regression Classifier:')
print('Training Accuracy:', accuracy_score(train_target_osampled,
logistic_regression_train_pred))
print('Validation Accuracy:', accuracy_score(val_target, logistic_regression_test_pred))
print('Confusion Matrix:\n', confusion_matrix(val_target, logistic_regression_test_pred))

Log_report = classification_report(val_target, logistic_regression_test_pred)
print(Log_report)

# Use the trained model to make predictions on the unseen data
unseen_pred = logistic_regression.predict(test_data_osampled)
print('Unseen Test Data Accuracy:', accuracy_score(test_target, unseen_pred))
```

KNN

GridSearch for KNN

```
param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11 ],
    'weights': ['uniform', 'distance'],
    'algorithm': ['ball_tree', 'kd_tree', 'brute']
}

# Create a KNN classifier
knn = KNeighborsClassifier()

# Perform grid search to find the best parameters
grid_search = GridSearchCV(estimator=knn, param_grid=param_grid, scoring='accuracy',
cv=5, n_jobs=-1, verbose=1)
grid_search.fit(train_data_osampled, train_target_osampled)
```

```
best_params = grid_search.best_params_

print("Best Parameters:", best_params)
```

KNN with best parameters

```
knn_clf = KNeighborsClassifier(**best_params)

knn_clf.fit(train_data_osampled, train_target_osampled)

train_pred = knn_clf.predict(train_data_osampled)
val_pred = knn_clf.predict(val_data_osampled)


print('K-Nearest Neighbors Classifier:')
print('Training Accuracy:', accuracy_score(train_target_osampled, train_pred))
print('Validation Accuracy:', accuracy_score(val_target, val_pred))
print('Confusion Matrix:\n', confusion_matrix(val_target, val_pred))

knn_report = classification_report(val_target, val_pred)
print(knn_report)

# Use the trained model to make predictions on the unseen data
unseen_pred = knn_clf.predict(test_data_osampled)
print('Unseen Test Data Accuracy:', accuracy_score(test_target, unseen_pred))
```

Decision Tree

Grid Search

```
dt_params = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 5]
}

dt_clf = DecisionTreeClassifier(random_state=42)
dt_grid_search = GridSearchCV(dt_clf, dt_params, cv=5, n_jobs=-1, verbose=3)
dt_grid_search.fit(train_data_osampled, train_target_osampled)
dt_best_params = dt_grid_search.best_params_
dt_best_params
```

decision tree with best params

```
dt_clf = DecisionTreeClassifier(**dt_best_params, random_state=42)
```

```
dt_clf.fit(train_data_osampled, train_target_osampled)
train_pred = dt_clf.predict(train_data_osampled)
val_pred = dt_clf.predict(val_data_osampled)
print('Decision Tree Classifier:')
print('Training Accuracy:', accuracy_score(train_target_osampled, train_pred))
print('Validation Accuracy:', accuracy_score(val_target, val_pred))
print('Confusion Matrix:\n', confusion_matrix(val_target, val_pred))

dt_report = classification_report(val_target, val_pred)
print(dt_report)

unseen_pred = dt_clf.predict(test_data_osampled)
print('Unseen Test Data Accuracy:', accuracy_score(test_target, unseen_pred))
```

Random Forest

Grid Search

```
rf_params = {
   'n_estimators': [100, 300, 500],
   'criterion': ['gini', 'entropy'],
   'max_depth': [None, 5, 10, 15],
   'min_samples_split': [2, 5, 10],
   'min_samples_leaf': [1, 2, 5]
}
rf_clf = RandomForestClassifier(random_state=42)
rf_grid_search = GridSearchCV(rf_clf, rf_params, cv=5, n_jobs=-1, verbose=1)
rf_grid_search.fit(train_data_osampled, train_target_osampled)
rf_best_params = rf_grid_search.best_params_
```

Random Forest with Best Parameters

```
rf_clf = RandomForestClassifier(**rf_best_params, random_state=42)
rf_clf.fit(train_data_osampled, train_target_osampled)
train_pred = rf_clf.predict(train_data_osampled)
val_pred = rf_clf.predict(val_data_osampled)
print('Random Forest Classifier:')
print('Train Accuracy:', accuracy_score(train_target_osampled, train_pred))
print('Test Accuracy:', accuracy_score(val_target, val_pred))
print('Confusion Matrix:\n', confusion_matrix(val_target, val_pred))

rf_report = classification_report(val_target, val_pred)
print(rf_report)

# Use the trained model to make predictions on the unseen data
```

```
unseen_pred = rf_clf.predict(test_data_osampled)
print('Unseen Test Data Accuracy:', accuracy_score(test_target, unseen_pred))
```

Random forest feature importance's

```
feat_importances = pd.Series(rf_clf.feature_importances_, index=data_encoded.columns[:-
1])
feat_importances.nlargest(10).plot(kind='barh')
plt.title('Feature Importances')
plt.show()
```

Under-Sampling

```
# These will be used for Undersampling later.

train_data_us = train_data
test_data_us = test_data
val_data_us = val_data
val_target_us = val_target
train_target_us = train_target
test_target_us = test_target

#unseen_data_us = unseen_data

# variable copies for undersampling
under_train_data = train_data_us
under_train_target = train_target_us
under_val_data = val_data_us
under_val_target = val_target_us
under_test_data = test_data_us
under_test_target = test_target_us
```

Random Under Sampling (RUS)

```
import random
from imblearn.under_sampling import RandomUnderSampler

def undersample(train_data, train_target):
    rus = RandomUnderSampler(random_state=0)
    train_data_resampled, train_target_resampled = rus.fit_resample(train_data,
train_target)

    return train_data_resampled, train_target_resampled
```

```
train_data_usampled, train_target_usampled = undersample(under_train_data,
under_train_target)

print(train_data_usampled.shape)
print(train_target_usampled.shape)

print("train_data_usampled.shape: ",train_data_usampled.shape)
print("train_target_usampled.shape: ",train_target_usampled.shape)
print("under_val_data: ",under_val_data.shape)
print("under_val_target: ",under_val_target.shape)
print("under_test_data: ",under_test_data.shape)
print("under_test_target: ",under_test_target.shape)
```

Standardization of Undersampled Data

```
scaler = StandardScaler()
train_data_usampled_scaled = scaler.fit_transform(train_data_usampled)
val_data_scaled = scaler.transform(under_val_data)
test_data_scaled = scaler.transform(under_test_data)
```

Logistic Regression on Undersampled Data

```
logistic_regression = LogisticRegression()

logistic_regression.fit(train_data_usampled_scaled, train_target_usampled)

logistic_regression_train_pred = logistic_regression.predict(train_data_usampled_scaled)
logistic_regression_val_pred = logistic_regression.predict(val_data_scaled)

print('Logistic Regression Classifier:')
print('Train Accuracy:', accuracy_score(train_target_usampled,
logistic_regression_train_pred))
print('Test Accuracy:', accuracy_score(under_val_target, logistic_regression_val_pred))
print('Confusion Matrix:\n', confusion_matrix(under_val_target,
logistic_regression_val_pred))

rf_report_us = classification_report(under_val_target, logistic_regression_val_pred)
print(rf_report_us)

# Use the trained model to make predictions on the unseen data
unseen_pred = logistic_regression.predict(test_data_scaled)
print('Unseen Test Data Accuracy:', accuracy_score(under_test_target, unseen_pred))
```

KNN on Undersampled Data¶

Grid Search

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV

# Define the parameter grid
param_grid = {
    'n_neighbors': [3, 5, 7, 9, 11 ],
    'weights': ['uniform', 'distance'],
    'algorithm': ['ball_tree', 'kd_tree', 'brute']
}

# Create a KNN classifier
knn = KNeighborsClassifier()

# Perform grid search to find the best parameters
grid_search = GridSearchCV(estimator=knn, param_grid=param_grid, scoring='accuracy',
cv=5, n_jobs=-1, verbose=1)
grid_search.fit(train_data_usampled_scaled, train_target_usampled)
beu_params = grid_search.best_params_

print("Best Parameters:", best_params)
```

KNN with best parameters

```
knn_clf = KNeighborsClassifier(**best_params)

knn_clf.fit(train_data_usampled_scaled, train_target_usampled)

train_pred = knn_clf.predict(train_data_usampled_scaled)
val_pred = knn_clf.predict(val_data_scaled)

print('K-Nearest Neighbors Classifier:')
print('Train Accuracy:', accuracy_score(train_target_usampled, train_pred))
print('Validation Accuracy:', accuracy_score(under_val_target, val_pred))
print('Confusion Matrix:\n', confusion_matrix(under_val_target, val_pred))

knn_report_us = classification_report(under_val_target, val_pred)
print(knn_report_us)

unseen_pred = knn_clf.predict(test_data_scaled)
print('Unseen Test Data Accuracy:', accuracy_score(under_test_target, unseen_pred))
```

Decision Tree CLassifier on Undersampled Data

Grid Search

```
dt_params = {
```

```
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 5]
}
dt_clf = DecisionTreeClassifier(random_state=42)
dt_grid_search = GridSearchCV(dt_clf, dt_params, cv=5, n_jobs=-1, verbose=1)
dt_grid_search.fit(train_data_usampled_scaled, train_target_usampled)
dt_best_params = dt_grid_search.best_params_
print(dt_best_params)
```

Decision Tree with Best parameters

```
dt_clf = DecisionTreeClassifier(**dt_best_params, random_state=42)
dt_clf.fit(train_data_usampled_scaled, train_target_usampled)
train_pred = dt_clf.predict(train_data_usampled_scaled)
val_pred = dt_clf.predict(val_data_scaled)
print('Decision Tree Classifier:')
print('Train Accuracy:', accuracy_score(train_target_usampled, train_pred))
print('Validation Accuracy:', accuracy_score(under_val_target, val_pred))
print('Confusion Matrix:\n', confusion_matrix(under_val_target, val_pred))

rf_report_us = classification_report(under_val_target, val_pred)
print(rf_report_us)

# Use the trained model to make predictions on the unseen data
unseen_pred = dt_clf.predict(test_data_scaled)
print('Unseen Test Data Accuracy:', accuracy_score(under_test_target, unseen_pred))
```

Random Forest Classifier on Undersampled Data¶

Grid Search

```
rf_params = {
    'n_estimators': [100, 300, 500],
    #n_estimators : This is the number of trees you want to build before taking the maximum
voting or averages of predictions.
    #Higher number of trees give you better performance but makes your code slower.
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 5]
}
rf_clf = RandomForestClassifier(random_state=42)
rf_grid_search = GridSearchCV(rf_clf, rf_params, cv=5, n_jobs=-1, verbose=1)
rf_grid_search.fit(train_data_usampled_scaled, train_target_usampled)
rf_best_params = rf_grid_search.best_params_
```

Random Forest with Best parameters

```
# Train random forest classifier with best parameters and measure performance
rf_clf = RandomForestClassifier(**rf_best_params, random_state=42)
rf_clf.fit(train_data_usampled_scaled, train_target_usampled)
train_pred = rf_clf.predict(train_data_usampled_scaled)
val_pred = rf_clf.predict(val_data_scaled)
print('Random Forest Classifier:')
print('Train Accuracy:', accuracy_score(train_target_usampled, train_pred))
print('Validation Accuracy:', accuracy_score(under_val_target, val_pred))
print('Confusion Matrix:\n', confusion_matrix(under_val_target, val_pred))

rf_report_us = classification_report(under_val_target, val_pred)
print(rf_report_us)

# Use the trained model to make predictions on the unseen data
unseen_pred = rf_clf.predict(test_data_scaled)
print('Unseen Test Data Accuracy:', accuracy_score(under_test_target, unseen_pred))

# Plot feature importances for the random forest classifier with undersampled data
feat_importances = pd.Series(rf_clf.feature_importances_, index=data_encoded.columns[:-
1])
feat_importances.nlargest(10).plot(kind='barh')
plt.title('Feature Importances')
plt.show()
```

Comparison of Results

Under Sampled Results

```
undersampled_results = {'Logistic Regression': {'Training Accuracy (UnderSampled)': 77.79,
'Validation Accuracy (UnderSampled)': 77.92, 'Testing Accuracy (UnderSampled)': 78.23},
        'K-Nearest Neighbours': {'Training Accuracy (UnderSampled)': 99.26, 'Validation
Accuracy (UnderSampled)':84.58 ,'Testing Accuracy (UnderSampled)': 84.38},
        'Decision Tree': {'Training Accuracy (UnderSampled)': 91.46,'Validation Accuracy
(UnderSampled)':85.38 , 'Testing Accuracy (UnderSampled)': 85.21},
        'Random Forest': {'Training Accuracy (UnderSampled)': 97.92 ,'Validation Accuracy
(UnderSampled)':89.34 , 'Testing Accuracy (UnderSampled)': 89.25}}

US_Data_Results = pd.DataFrame(undersampled_results).transpose()
US_Data_Results
```

Over Sampled Results

oversampled_results = {'Logistic Regression': {'Training Accuracy (OverSampled)': 79.30,'Validation Accuracy (OverSampled)': 78.15, 'Testing Accuracy (OverSampled)':77.68},
'K-Nearest Neighbours': {'Training Accuracy (OverSampled)': 99.38,'Validation Accuracy (OverSampled)': 86.72, 'Testing Accuracy (OverSampled)': 87.87},
'Decision Tree': {'Training Accuracy (OverSampled)': 99.46,'Validation Accuracy (OverSampled)': 87.08, 'Testing Accuracy (OverSampled)': 88.33},
'Random Forest': {'Training Accuracy (OverSampled)': 99.46 ,'Validation Accuracy (OverSampled)': 90.55, 'Testing Accuracy (OverSampled)': 89.99}}

OS_Data_Results = pd.DataFrame(oversampled_results).transpose()
OS_Data_Results

Word Count Note:

Total Words: 13421
Words after excluding tables, references, and appendices: 11153.
Words after excluding tables also: 11153-612(words in tables) = 10541 (approx. to 10000)

========== THE END ==========