**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.

A screenshot of a computer

Description automatically generated

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 plottedfor,

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.

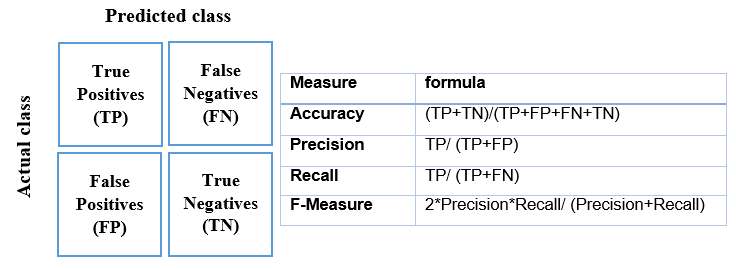


Fig 9: Confusion matrix and performance metrics

[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**

A computer screen shot of a diagram

Description automatically generated

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.