ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

Project I

Predicting Hotel Booking Cancellations

Christina Kataki

Professor:
Dimitrios Karlis

April 2024

Contents

1 Introduction

Over the years, the hotel industry has changed with a majority of bookings now made through third parties (e.g. Booking.com, trivago.gr). Online travel agencies (OTAs) have made cancellation policies the centerpiece of their marketing campaigns, rather than just a little print at the bottom of the page. As a result, clients are accustomed to free cancellation policies and the financial consequences are tremendous. According to a D-Edge Hospitality Solutions survey, the cancellation rate increased by 6% over the previous four years, hitting nearly 40% in 2018. Knowing which reservations are most likely to be canceled helps hotels to organize their operations more effectively. Certain aspects of the reservation itself might serve as reliable predictors of cancellation rates. It is important to investigate the types of bookings that are being canceled.

The main goal of this project is to investigate different approaches that can forecast whether or not a reservation will be canceled. Specifically, four distinct machine learning models will be fit on a given dataset using the following 4 methods:

- K-nearest neighbors (KNN) Algorithm
- Decision Tree Algorithm
- Random Forest Algorithm
- Support Vector Machine (SVM) Algorithm

The four models will be evaluated on a test set of the data and depending on how accurately they predict cancellations, a final model will be chosen.

2 Exploratory Data Analysis (EDA)

Exploratory Data Analysis (EDA) is a process of describing the data by means of statistical and visualization techniques in order to bring important aspects of that data into focus for further analysis. This involves inspecting the dataset from many angles, describing and summarizing it without making any assumptions about its contents. EDA is a significant step to take before diving into statistical modeling or machine learning, to ensure the data is really what it is claimed to be and that there are no obvious errors.

First of all, “Garbage in, garbage out” applies to all the models that we use for modeling purposes. The cleaner and more informative the dataset we feed into our machine learning algorithms, the better they learn. There are no missing values in the dataset but there are two observations that have a different format in the reservation date column. Since the aforementioned observations constitute a minute proportion of the whole dataset (only 2 out of 2000) they should not impact model training and hence will be dropped.

During the analysis, an initial search for outliers in the dataset was conducted. It is probably worth adding that the ’outliers’ that are sometimes automatically plotted with boxplots are only ’outliers’ in that they are at the edges of the data. They are not usually ’outliers’ in the sense of being unrepresentatively or erroneously wild, so removing them is a serious step and should be taken cautiously. It is most effective when there is a high level of confidence that the outlier results from a measurement error, which can be challenging to determine during analysis in many cases. In this case, the decision was made to retain them for the analysis and model training.

Moreover, in order to make the reservation date column meaningful, it was divided into distinct day, month, and year columns.

2.1 Quantitative variables

In the following visualization (Figure 1), some of the quantitative variable histograms are shown.

Based on the histograms:

- It is evident that bookings with two adults are the most common. The same is true for bookings with no children.

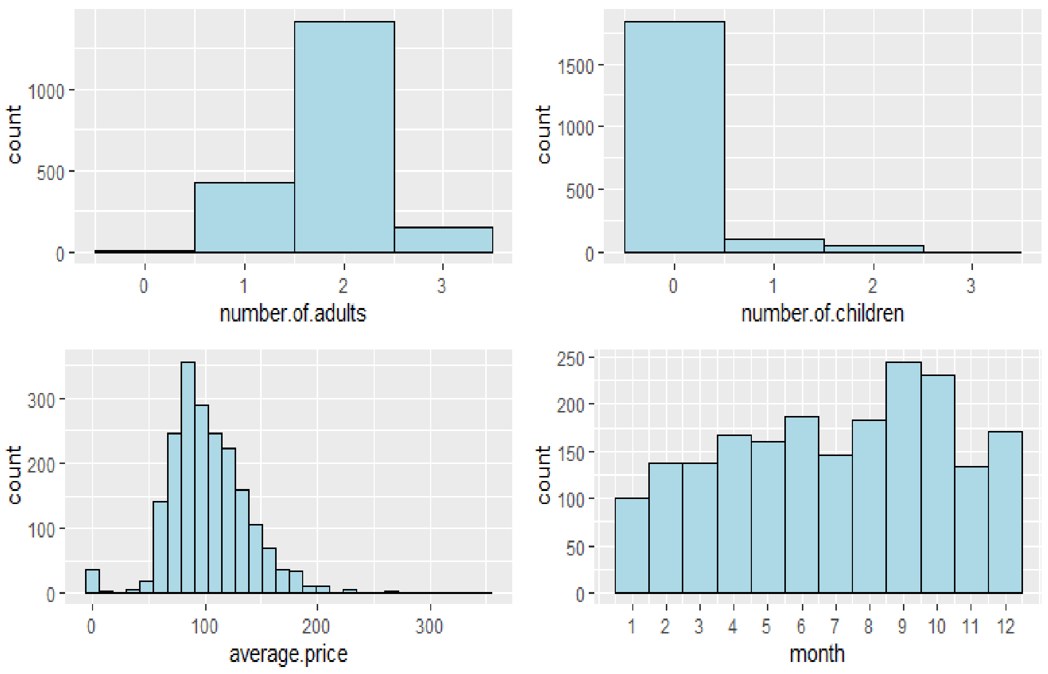

Figure 1: Histograms for numerical variables

- The average booking cost for the majority of individuals falls between \$50 and \$150.
- September and October are the peak months for reservations.

2.2 Categorical Variables

Pie charts (Figure 2) lead us to the following conclusions:

- Meal of type 1 is the most prevalent.
- In 76.28% of the bookings, the type 1 room is preferred.
- Most individuals opt for an online reservation.
- Approximately one third of the bookings were canceled.

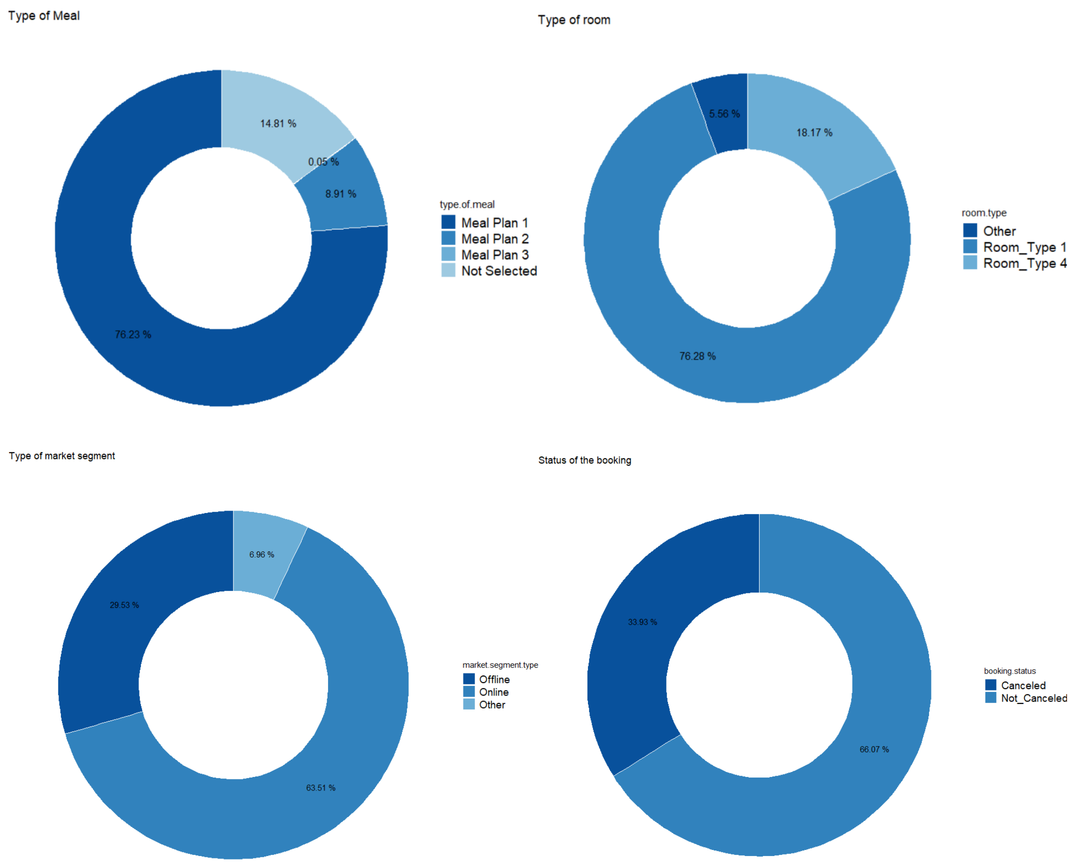

Figure 2: Pie charts for nominal variables

2.3 Pairwise Associations

In the upcoming visual (Figure 3), exploring the connection between the status of the booking (canceled or not canceled) and specific numeric and binary factors reveals interesting patterns.

It is clear that lead time is the most highly correlated feature with whether or not a booking is canceled. It makes sense that as the number of days between the booking date and the supposed arrival date increases, customers have more time to cancel the reservation and there is more time for an unforeseen circumstance derailing travel plans to arise.

Interestingly, the total number of special requests is the second feature with the strongest correlation to our cancellation target. As the number of special requests made increases, the likelihood that a booking is canceled decreases. This suggests that engagement with the hotel prior to arrival and feeling like their needs are heard may make a customer less likely to cancel their reservation.

Finally, a customer's prior history with the hotel (measured by the number of previous bookings) does not seem to be highly correlated with whether or not the current booking will be canceled. On the other hand, whether or not a customer is a repeated guest is more highly correlated with whether or not the current booking will be canceled.

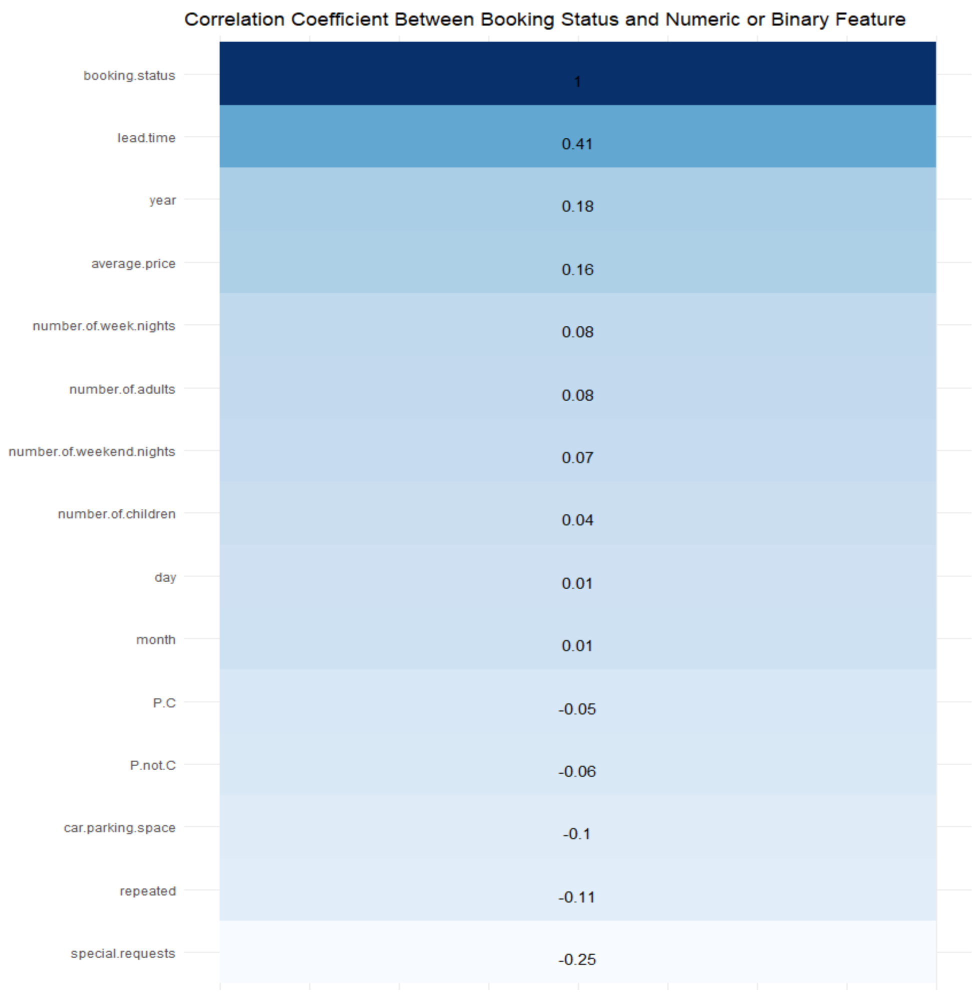

Figure 3: Correlations

3 Machine Learning Modelling

In the entire assignment, package `caret` was used to perform hyperparameter tuning, train and evaluate models. `caret` is an R package for building and evaluating machine learning models. It provides an interface for major machine learning algorithms.

Before feeding the data to the models for training, some transformations were applied. Specifically:

- Although reservation date was split into day, month and year for the purposes of EDA, a format much more interpretable by the models, the cyclic nature of months is absent in this representation. To correct this, both sine and cosine transformations of the month were applied, effectively representing the periodic variations in monthly data throughout the years.
- Label encoding was applied to non-numeric columns, including the target variable ”booking.status”.
- Data was standardized so that all features have a mean value of 0 and variance 1. This prevents features with large scales to dominate the decision making of the model.

Apart from the above transformations, the seed for reproducibility was set and the dataset was split into train and test datasets using an 75:25 ratio. Models were then trained on the train set and evaluated on the test set using the following metrics:

- **Accuracy:** Percentage of correctly classified observations (true positives and true negatives).
- **Specificity:** Percentage of correctly classified non-canceled bookings (true negatives).
- **Sensitivity:** Percentage of correctly classified canceled bookings (true positives).
- **F1 Score:** It is the harmonic mean of precision and recall, where precision is the ratio of true positives to the sum of true positives and false positives, and recall is the ratio of true positives to the sum of true positives and false negatives. It’s calculated as $F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$.

3.1 K-nearest neighbors (KNN) Algorithm

The k-nearest neighbors (KNN) is a non-parametric, supervised machine learning algorithm, which uses proximity to make classifications or predictions about the grouping of an individual data point. Its worth noting that a supervised machine learning algorithm (as opposed to an unsupervised machine learning algorithm) is one that relies on labeled input data to learn a function that produces an appropriate output when given new unlabeled data. KNN is one of the popular and simplest classification and regression classifiers used in machine learning today.

As already mentioned, the KNN algorithm assumes that similar things exist in close proximity and captures this idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph. There are many ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, the straight-line distance (also called the Euclidean distance) is a popular and familiar choice.

Here's how KNN was employed for the purposes of the assignment:

1. **K for Knowledge:** The first step involved choosing the optimal value of K, which is the number of nearest neighbors that will be used to make the prediction.

2. **Distance Makes the Difference:** The distance between that point and all the points in the training set was calculated.

3. **The Closest Counselors:** The K nearest neighbors were selected based on the distances calculated.

4. **Majority Rules:** The label of the majority class was assigned to the new data point.

5. **Repeating for the Greater Good:** In the test set, steps 2 to 4 were repeated for all the data points in the test set.

6. **The Performance Checkup:** Finally, the model's accuracy was evaluated.

Hyperparameter Tuning

KNN has two important hyperparameters, one being the number of neighbors K and the distance metric. For the latter, the Euclidean distance was used while the optimal value of K was found with the help of the train function. The train function requires a formula, scaled training dataset, model name, train control method (5 times repeated 10 folds cross-validation), and list of hyperparameters. The model performance was checked for values of K between 2 to 10. After finding the best value of K ($K = 3$) via grid search, the KNN classification model was trained with the scaled training dataset.

Note: Repeated k-fold cross-validation provides a way to improve the estimated performance of a machine learning model. This involves simply repeating the cross-validation procedure multiple times and reporting the mean result across all folds from all runs.

Model Evaluation

The confusion matrix, model accuracy, P-Value, model sensitivity, and other important metrics were extracted in order to determine the stability and performance of the model. The model achieved an accuracy of 79.2%, indicating that nearly 79% of the bookings were correctly classified as canceled or not canceled. While the model performed better at identifying bookings that wouldn't be canceled (Specificity: 84.84%), it performed lower in predicting actual cancellations (Sensitivity: 68.23%). This indicates a potential bias towards the majority class (not canceled bookings) which might exist in the data. Balanced accuracy, which adjusts for the imbalance between the two classes, was determined to be 76.54%. This suggests the model performs moderately well on both classes, even if the dataset might be imbalanced. However, with the F1 score being 69.04%, there's potential for improvement with the use of alternative classification algorithms which will be explored subsequently.

Additionally, it's worth noting that the label encoding used for nominal categorical variables may not capture the true relationships between categories. This could potentially limit the model's performance and generalizability. Reevaluating the encoding strategy or exploring alternative encoding methods could be beneficial to improve the model's effectiveness and reliability.

3.2 Decision Tree Algorithm

Decision trees are special in machine learning due to their simplicity, interpretability, and versatility. Similar to KNN, they are also a supervised machine learning algorithm. They serve as the foundation for more advanced techniques, such as bagging, boosting, and random forests.

A decision tree is a flowchart-like tree structure where each internal node denotes the feature, branches denote the rules and the leaf nodes denote the result of the algorithm.

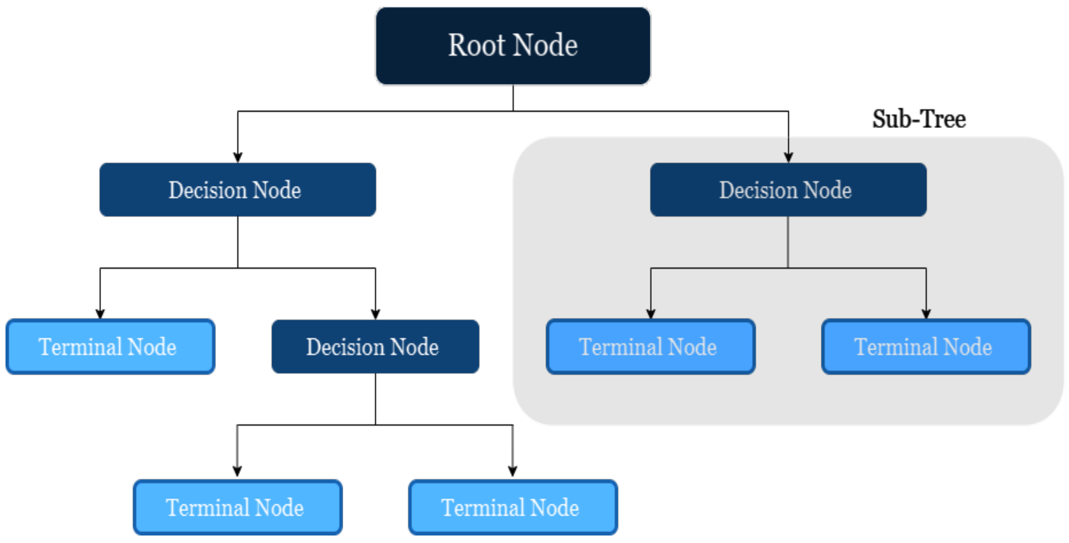

Figure 4: Decision Tree

Here is how a decision tree works (Figure 4):

It starts with a root node that signifies the whole population or sample, which then separates into two or more uniform groups via a method called splitting. When sub-nodes undergo further division, they are identified as decision nodes, while the ones that don't divide are called terminal nodes or leaves. A segment of a complete tree is referred to as a branch.

It is obvious that one of the most critical steps in building a decision tree is to determine how to split the data at each internal node. This process is known as splitting a tree, and it directly impacts the accuracy and performance of the resulting model. There are different methods to split a tree but, since the target variable is categorical only Gini Impurity and Information Gain will be mentioned.

- **Gini Impurity**
  Gini Impurity is the most popular and easiest way to split a decision tree. It is a measure of the likelihood of misclassification of a random sample. In other words, it measures the probability of the randomly chosen element being wrongly labeled if it were labeled randomly according to the distribution of labels in the dataset. The lower the Gini Impurity, the higher the homogeneity of the node. It is worth mentioning that it is preferred to Information Gain because it does not contain logarithms which are computationally intensive.

- **Information Gain**
  Information Gain is a measure of the reduction in entropy achieved by partitioning the examples according to a given attribute. Entropy can be defined as the measure that tells us how disorganized and mixed up our data is. The higher the information gain, the more valuable the feature is in predicting the target variable. So, the attribute that maximizes information gain is chosen as the splitting criterion for building the decision tree.

Hyperparameter Tuning

Package `caret` allows tuning of only one hyperparameter of decision trees, the complexity parameter *cp*. Any split that does not decrease the overall lack of fit by a factor of *cp* is not attempted. The main role of this parameter is to save computing time by pruning off splits that are obviously not worthwhile. A grid search was performed in the range (0.01, 0.5) with step 0.01. The optimal value of *cp* was found to be 0.01.

Evaluation

For the sake of this assignment both splitting methods (Gini Impurity and Information Gain) were employed. The two produced models demonstrated similar performance with the Information Gain model achieving slightly higher in accuracy and specificity (Table 1). On the other hand, Gini Impurity model reached greater sensitivity (72.35%) and F1 score (71.1%), possibly making it a more desirable model since the focus is on detecting cancellations reliably. In comparison to KNN, both models are superior increasing overall accuracy, sensitivity and F1 score.

	Accuracy	Sensitivity	Specificity	F1 Score
Information Gain Model	80.4%	70%	85.76%	70.83%
Gini Impurity Model	80%	72.35%	83.93%	71.1%

Table 1: Decision Tree Models Metrics

The variable importance plot containing the ten most important features according to the Gini Impurity model is displayed below in Figure 6. The feature ”lead.time” appears to have the most influence on the model, which comes as no surprise given the correlation analysis in the EDA section.

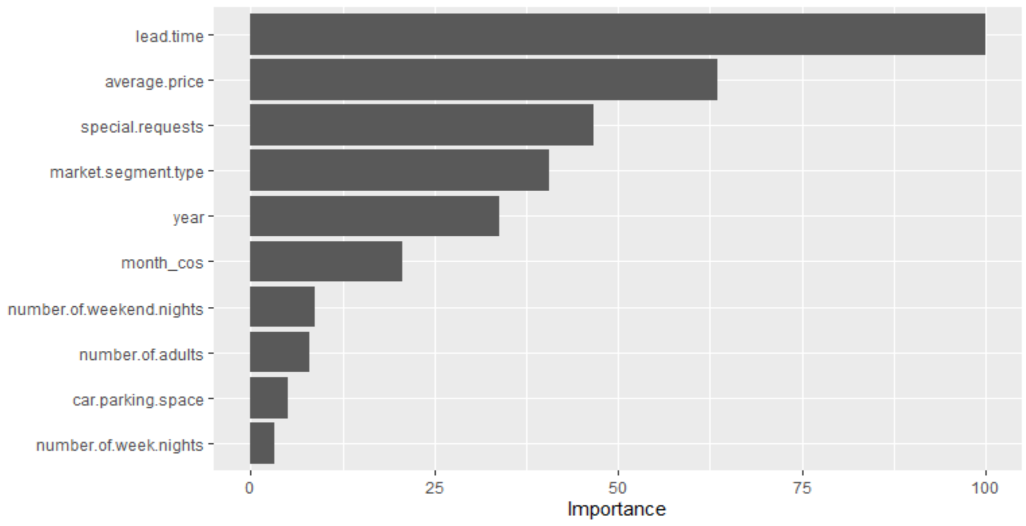

Figure 5: Variable Importance Plot

3.3 Random Forest Algorithm

A random forest algorithm consists of many decision trees. Each tree is constructed using a random subset of the data set to measure a random subset of features in each partition. The 'forest' generated by the random forest algorithm is trained through bagging or bootstrap aggregating. Bagging is an ensemble meta-algorithm that improves the accuracy of machine learning algorithms.

The random forest algorithm establishes the outcome based on the predictions of the decision trees. It predicts by taking the average or mean of the output from various trees (Figure 6). Increasing the number of trees increases the precision of the outcome. Unlike a single decision tree, which can be prone to overfitting and may have lower accuracy due to its sensitivity to variations, random forest tends to offer improved accuracy.

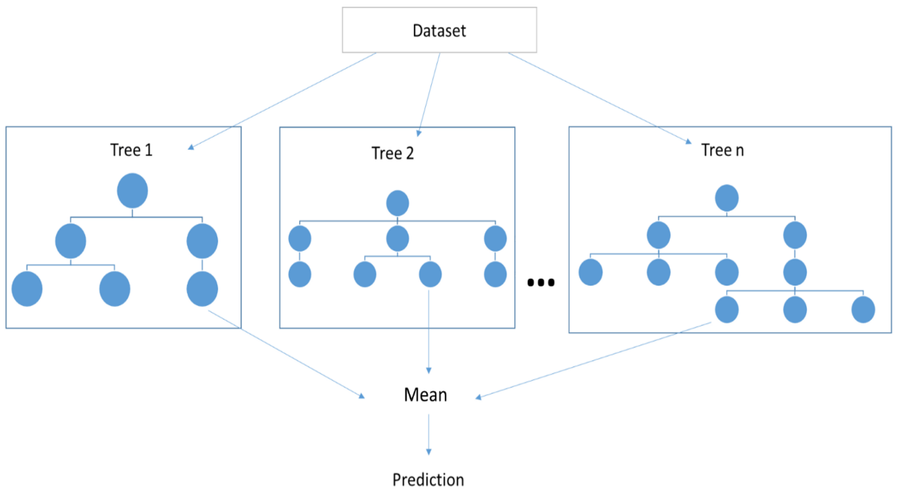

Figure 6: Random Forest

Understanding Bagging in Random Forest

In 1996, Leo Breiman introduced the bagging algorithm, which has three basic steps:

1. **Bootstrapping:** Bagging uses bootstrapping to create varied training data subsets with resampling. Through this method, data points are ran-

domly selected with replacement, allowing for the possibility of choosing the same instance multiple times within a sample.

2. **Parallel training:** These bootstrap samples are then trained independently and in parallel with each other using weak or base learners.

3. **Aggregation:** Finally, an average or a majority of the predictions are taken to compute a more accurate estimate. For classification tasks, like here, the class that receives the most votes among the classifiers is selected, termed as hard voting or majority voting.

The random forest algorithm is essentially a bagging algorithm: it draws random bootstrap samples from the training set. However, in addition to the bootstrap samples, it also selects random subsets of features for training each individual tree. Unlike bagging, where each tree is provided with the complete set of features, this random feature selection ensures that the trees are more independent from each other. This often results in improved predictive performance due to a better balance between variance and bias, making random forest faster than traditional bagging. Importantly, each tree in a random forest learns from only a subset of features.

Hyperparameter Tuning

The package `caret`, allows tuning of only one hyperparameter for random forests: *mtry*, which determines the number of features randomly selected as candidates at each split. In other words, *mtry* controls how much randomness is added to the decision tree creation process. A grid search was performed in the range $(1, 5)$ and the optimal value of *mtry* was found to be 5.

Model Evaluation

The Random Forest model delivered promising results on unseen data. It achieved an accuracy of 83.4%, confidently classifying over four-fifths of the data points (confidence interval: 79.84% - 86.56%). This significantly outperforms a random guess (66%), highlighting the model's ability to learn meaningful patterns. The model demonstrated a strong ability to identify non-cancelled bookings (specificity: 91.51%), although there's potential to improve its accuracy in classifying cancelled bookings (sensitivity: 67.64%). Overall, the balanced accuracy (79.58%) suggests fair performance across both positive and negative classifications, something which is further supported by the F1 score of 73.48%.

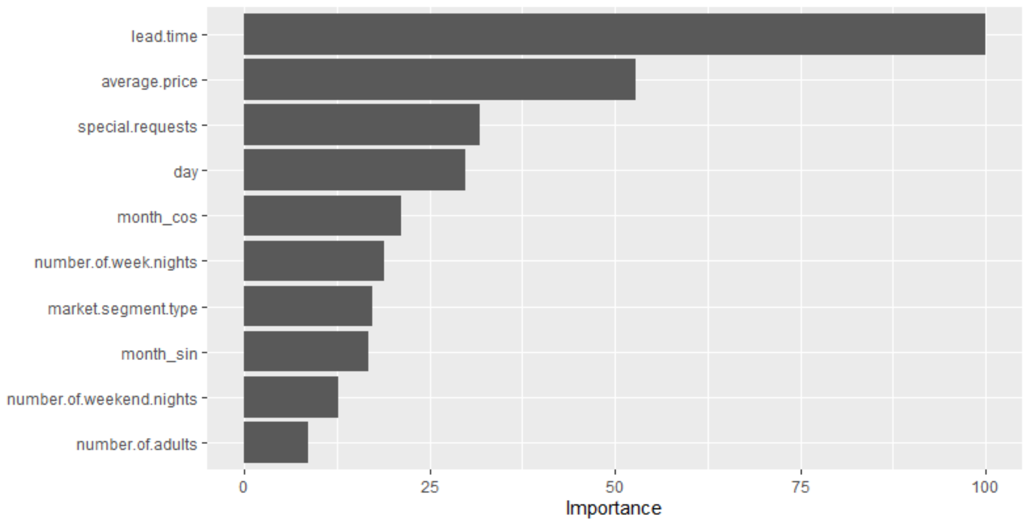

Figure 7: Variable Importance Plot

Examining feature importance (Figure 7) revealed differences between the decision tree and random forest models. The decision tree prioritized ”lead.time” for making crucial splits within its structure, followed by features like ”average.price” and ”special.requests.” Even though the Random Forest highlighted the aforementioned variables as consistently important, it also identified ”day” as a relevant feature, potentially reflecting the influence of different decision paths across multiple trees. This suggests that the decision tree prioritizes specific decision points, while the random forest captures broader feature influences.

3.4 Support Vector Machine (SVM) Algorithm

Support Vector Machines (SVMs) are supervised learning models primarily used for classification, although they can also be adapted for regression tasks. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that a new data point can be easily placed in the correct category in the future. This best decision boundary is called a hyperplane. The idea is to maximize the margin, which is the distance between the hyperplane and the closest data points of each category.

SVMs have different types and variants that provide specific functionalities and address specific problem scenarios. Here are two types:

- **Simple or Linear SVM**
  A linear kernel is used by SVMs to draw a decision boundary that divides classes in a straight line. They work well in situations where a linear approximation is sufficient or where the data can be divided linearly. Since the choice boundary in a SVM is a hyperplane in the input feature space, these models are computationally economical.

- **Kernel or Non-Linear SVM**
  In situations where a straight line cannot separate the data in the input feature space, nonlinear SVMs are used. They achieve this by using kernel functions that, by implicit data mapping, place the data into a feature space of higher dimension where a linear decision boundary is located. The polynomial, sigmoid, and Gaussian (RBF) kernels are common kernel functions utilized in this kind of SVM. Compared to linear SVMs, nonlinear SVMs can capture complicated patterns and achieve higher classification accuracy.

As one might expect, the selection of kernel parameters has a significant impact on SVMs. In situations where there are more features per object than there are training data samples, SVMs may not function well. This makes intuitive sense since it suggests that the high-dimensional feature space is substantially bigger than the samples. As more unobserved samples are added, the classification performance deteriorates because there are less efficient support vectors to place the ideal linear hyperplanes on.

Hyperparameter Tuning

Both a Linear model and a Non Linear SVM model with RBF kernel was fit to the training data in order to determine the best option. As with previous models, hyperparameters were tuned with grid search.

- **Linear SVM**
  In the case of linear SVM, `caret` allows tuning the hyperparameter C, a regularization parameter controlling the trade-off between maximizing the margin and minimizing classification error. Different values of C were explored and the optimal one was found to be $C = 100$.

- **Non Linear SVM**
  In the case of non-linear SVM, apart from the hyperparameter C, `caret` allows tuning σ. This parameter acts like a scaling factor and determines the width or spread of the influence a single data point has on the decision boundary. Grid search led to the optimal values of $C = 10$ and $\sigma = 0.01$.

Evaluation

In the case of SVMs, the RBF kernel model seems to be more appropriate for the particular dataset. Its accuracy is approximately 2% higher than that of the linear SVM, while sensitivity, specificity and F1 score are notably higher as well. Despite that, it falls short compared to random forests in all aspects. Detailed results for the two SVM models are displayed in Table 2.

	Accuracy	Sensitivity	Specificity	F1 Score
Linear SVM	79.6%	64.70%	87.27%	68.32%
Non-Linear SVM	81.6%	67.64%	88.78%	71.42%

Table 2: SVM Models Metrics

4 Final Model and Features Selection

In the process of choosing the final model, the primary concern lies in minimizing the False Negative Rate to mitigate the risk of revenue loss. A False Negative occurs when the model incorrectly predicts that a guest who intends to cancel will not do so. This misclassification could pose a significant risk to a hotel business, leading to unoccupied rooms that could otherwise be resold or listed on online travel platforms.

On the other hand, while minimizing False Negatives is crucial, the implications of False Positives must also be considered. A high False Positive rate means the model wrongly predicts cancellations, potentially leading to overbookings. Acting on these predictions by reselling or relisting rooms could result in double bookings if the anticipated cancellations do not occur and no alternative rooms are available.

Given these considerations, an ideal model should prioritize accurate predictions within the positive class, emphasizing a high recall score to prevent revenue loss and maximize room utilization. However, maintaining a balance between precision and recall is essential to minimize the risk of overbooking.

To achieve this balance, the focus will be placed on optimizing the F1-score. By prioritizing the F1-score, the aim is to choose a model that effectively minimizes both False Negatives and False Positives, ensuring a more reliable and efficient booking management system.

Of those models explored in the previous section, random forest fits this description the most. To determine the most influential variables, a systematic feature selection process was performed using random forest models trained with varying subsets of features. The process began with the most important features, as determined by the random forest model, and progressively incorporated less influential ones. For each subset of features, the models' performances were evaluated on the test set, capturing metrics such as accuracy, sensitivity, specificity, and the F1-score.

In the final random forest model, the selected features were $lead.time$, $average.price$, $special.requests$, day, $month\_cos$, $number.of.week.nights$, and $market.segment.type$. These features were chosen based on their importance and contribution to the model's predictive performance in earlier iterations. Grid search was employed to fine-tune the hyperparameter $mtry$ with values ranging from 1 to 5 and the optimal $mtry$ value was found to be 3.

The evaluation of this final model on the test set produced the metrics of Table 3. The results confirm the model's effectiveness in predicting booking status based on the selected features. Focusing on this feature subset maintains a good balance between performance and computational efficiency.

	Accuracy	Sensitivity	Specificity	F1 Score
Final Random Forest	82.6%	69.41%	89.39%	73.06%

Table 3: Performance Metrics of the Final Random Forest Model

5 Conclusions

In this assignment different machine learning algorithms were explored to predict hotel booking cancellations. The analysis revealed that a significant portion of bookings (nearly 40%) are canceled, posing a challenge for hotel revenue management.

Exploratory Data Analysis (EDA) provided valuable information for the dataset, highlighting factors potentially influencing cancellation rates. These factors included lead time (days between booking and arrival), number of special requests, and guest history. However, the number of previous bookings did not significantly correlate with cancellation likelihood and repeat guests (measured by a binary variable) exhibited a weaker correlation.

Four machine learning models were evaluated: K-Nearest Neighbors (KNN), Decision Tree, Random Forest, and Support Vector Machine (SVM). Among these models, random forest appeared as the most suitable with an accuracy of 82.6%, sensitivity of 69.41%, specificity of 89.39%, and an F1-score of 73.06%. These metrics show the model's capability to effectively identify both cancelled and non-cancelled bookings.

However, it is essential to acknowledge the limitations and potential areas for future improvement. This assignment relied on a single dataset and feature engineering methods may have introduced biases or overlooked relevant predictors. Exploring alternative encoding strategies and incorporating additional features could further enhance the model's predictive power and generalizability.

Bibliography

[1] James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning: with Applications in R*. Springer.

[2] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition.* Springer Series in Statistics.

[3] Kuhn, M. (2020). *caret: Classification and Regression Training.* R package version 6.0-86.

[4] D-Edge Hospitality Solutions. (2018). *Impact of cancellation rates on the hotel industry.*

[5] Ripley, B., Venables, W., & Ripley, M. B. (2020). *Package 'class'. Functions for classification.*

[6] Venables, W. N., & Ripley, B. D. (2020). *Package 'caret'. Functions for training and plotting classification and regression models.*