Nova University of Lisbon

NOVA IMS Information Management School

Postgraduate Program in Enterprise Data Science & Analytics

Book.me

# Group Project - Supervised Learning

## Book Me

## Data Science and Machine Learning

Ana Lúcia Barriga, 20211812

Ana Rita Coelho, 20211335

João Frederico Monteiro, 20211786

Manuel Félix, 20211333

Manuel Fernandes, M20180396

Professors: Carina Albuquerque, Lara Oliveira and Roberto Henriques

Spring Semester 2022

## Contents

## 1 Background

Following the immense success on our clustering and marketing strategy, *BookMe* company administrators were followed up with a reach-trial that forecast a revenue increase by 3.14x. Purely based on this speculation, and partially due to watching the *Iron Man 3* movie, they decided to using the leftover seed money raised to create a "J.A.R.V.I.S." of sorts (AI bot) to predict which customers have a higher probability of churn depending on their satisfaction and characteristics. Initially, they wanted to purchase a trial version from Amazon, but when they looked at the price tag, they decided that our data scientists team was the right choice to develop it.

Similarly to the first process, our team broke the challenge into smaller chunks: use data sets containing general information regarding the customers behaviour and satisfaction available; perform feature selection carefully; experiment with several machine learning algorithms provided by *scikit-learn* (because it's free!) such as Decision Trees, Neural Networks, Support Vector Machines (SVMs), Bayesian and Ensemble methods (mostly tree-based such as Random Forests, Gradient Boosting and Extra Trees). The final task was to assess performance and determine the model to use in building J.A.R.V.I.S.

## 2 Data Exploration

Fortunately for us, we had already gathered most of the information required for this analysis (Figure 1), as we were building the model based on the same parameters extracted for our business case and marketing plan, so we revisited our variables (as a sanity check) and concluded similarly that: missing values appear in *Year_Birth* (required replacing); the various satisfaction variables showed different rating ranges, with some starting in 0 while others in 1; the satisfaction variable *WiFi* showed a maximum rating of 6 while every other variable had a maximum of 5; there are 1051 underage customers, where we considered they would have travelled as companions (and eligible for ratings) we consider that they can be travelling themselves or they can be considered as travel companions. So, no alterations are to be performed on these customers. Moving to the analysis of the categorical variables (Figure 5, Figure 6), we can see that most users are antique users, they registered more than a year ago in the platform. Also, most users travel by business. Regarding room type, the preference overall falls in single and double over suite. There's a trend when travelling by business, most bookings are single rooms. When travelling by leisure, most bookings are double rooms. Using the describe method: we didn't found any univariate variables.

Similarly, nothing pointed out that any outliers would be presented here: we conducted skewness and kurtosis evaluations to find possible outliers, resulting in some minor flags in Staff, Amenities, CheckOut and Cleanliness (negatively skewed variables, with absolute skewness values over 0.5) but ultimately not representing any outliers. *RewardPoints* remains the most mysterious variable, as there are only some guidelines on its representation and there's no direct relation between number of Reward Points and any other variable. However, we chose not to categorize them as outliers (even though it has a positive kurtosis value, to which we have identified some outliers using a histogram in Figure 3 and box plots in Figure 2) for our business case.

We were also able to bring our previous insights on the different ratings and buckets of variables, as per the Spearman correlation heat map (Figure 4) described previously (with *Comfort* and *FoodDrink*, *Staff* and *BarService* and between *OnlineBooking* and *BarService* correlations being higher than 0.7). However, we would discuss whether we would keep the rating buckets, as not to skew the supervised model with our co-linearity bias.

## 3 Data Pre-Processing

With our insights gathered from our first assignment and double-checked, we proceeded to clean our data set, either by handling any data inconsistencies, missing values and non-normalized data, and by creating additional features that can be valuable in the context of our problem. However, not all data transformations should be done directly on the imported and explored *train.csv* data set, since it will be split in both train and validation components.

No duplicated records were found within the *train.csv* data set and existence of outliers (customers who rate all satisfaction variables equally or customers with few reward points) was also disregarded, since both situations are valid from our point of view. Underage customers were not considered an issue as well. Other data inconsistencies in the data set were fixed, namely replacing the ratings of 6 in the *WiFi* variable to 5, the maximum value allowed, so that all satisfaction variable lie within the same range; and changing some of the values in the *Longevity* variable, as some customers wrote *"y"* instead of *"yes"* as an answer.

Some features, created from the ones available in the data set, were also created ahead. The variable *Gender* was created from the variable *Name*, from the prefix *"Mr."* or *"Ms."*; on its turn, the variable *RewardPoints* was transformed into a categorical variable *RewardPoints_Bins*, in which customers are separated in bins of equal width - *bronze*, *silver*, *gold* and *platinum* - according to their number of reward points. Consequently, the variables *Name* and *RewardPoints* are dropped from the data set. In accordance to the correlations between the satisfaction variables, we also created three buckets of variables - *Rating1*, composed by *Comfort*, *ReceptionSchedule*, *FoodDrink* and *Location*; *Rating2*, composed by *Amenities*, *Wifi*, *OnlineBooking*, *Staff* and *BarService*; and *Rating3*, composed by *PriceQuality*, *RoomSpace*, *CheckOut*, *CheckIn* and *Cleanliness*.

As the *train.csv* data set will be split for both training and validation tasks, the replacement of the missing values detected on the variable *Year_Birth* and the scaling of the data set must be performed carefully using only data belonging to the train data set. This way, no information from the validation data set is used to fill missing values and scaling only considers the training data as well. This way, such tasks, along with the conversion of data types *int64* to *int8* to account for performance issues, are done within each stratified $k$-fold cross-validation iteration, the elected method to evaluate our models. The missing values of the column *Year_Birth*, which appear to be of the type *missing at random* (MAR) since that about 81.5% of customers who do not fill their age are women. We explored whether the filling of these missing values would be done using the $k$-Nearest Neighbors (KNN) Imputer, mean, median or mode, and similar, equivalent results were attained with each one of them. Thus, for the sake of simplicity and to decrease the computational time associated with the feature selection and model evaluation tasks, we decided to replace the missing values using the mode of the variable *Year_Birth*. The scaling was performed using the *MinMaxScaler*. Finally, note that the proper creation of dummy variables for the categorical predictors that will be considered is only performed after the task of feature selection, again within each iteration of stratified $k$-fold cross-validation after the split into training and validation data sets.

Now that we have accounted for all these data cleaning and pre-processing tasks, we moved forward to understand and select which features should be used in our model using feature selection.

## 4 Feature Selection

One of the new challenged we faced was the decision of which features to use in our model, and how many. Feature selection involves several methods that allow us to select the features, among all the features available (both the original ones and the newly created ones during data pre-processing), that are relevant to be used during model building, thus allowing the creation of more accurate, efficient models. Multiple methods can be used for feature selection. They can be summarized into filter methods, which are the simpler and based essentially on statistical methods; wrapper methods, that search for the best selection on features based on a predictive algorithm; and embedded methods, which are actual machine learning algorithms which perform the selection of features during model training.

However, firstly, we aimed to understand the weight of the *Churn* variable, our target; this is, how does the churn behaviour of our clients change according to the values in each one of our predictor variables. In order to achieve this, we have drawn some proportion bar charts (Figure 7, Figure 8, Figure 9 and Figure 10) that illustrate such dependency. This helps us, in a quick and relatively easy manner, to pinpoint patterns in each of the features and understand how they would likely affect the overall model. When observing the plots for the categorical variables, we can indeed corroborate the findings from our data exploration task; for instance, that older customers (*Longevity = "yes"*) are less likely to churn, or that customers who travel by business reasons or book single rooms are also less likely to churn. On its turn, the churn behaviours appears to be more uniform when observing the plot for the variable *RewardPoints_Bins*. By analysing the plots for rating variables, we can state that it is easier to explain the churn behaviour associated to *OnlineBooking*, *BarService*, *PriceQuality* and *CheckIn*, in each poorer ratings lead the customers to churn more and greater ratings lead them to churn less, than to the churn behaviour associated to *Comfort*, *RoomSpace* or *CheckOut* variables. Few hypothesis can be formulated: it is possible that, perhaps, these latter variables are less important for the customer to churn; or, on the contrary, they may be associated to some kind of distinct behaviour that our machine learning models should consider. Or ultimately, they might have no weight whatsoever. We would require to run a feature selection in detail to try to ascertain which ones would likely be kept and which ones discarded, but our team would revisit the plots if needed to help make those decisions.

In our general approach for feature selection, we employed distinct techniques, belonging to filter, wrapper and embedded methods categories. For categorical variables, the Chi-Square test, which tests the independence between each categorical variable and the target, was used. The result suggested that all categorical variables in the data set were relevant for the prediction, however, we ended up disregarding *RewardPoints_Bins* due to the poor weight of the target variable relationship, loose interpretation of this variable and disregard based on the promotional packages we had suggested in the business case with the marketing plan. This would avoid selection bias when building the model, as we would be able to tweak it to reinforce specific target demographics, and our goal is to create an unbiased model from scratch. As such, *Longevity*, *RoomType*, *TypeTravel* and *Gender* were kept as predictors for the models. Regarding the numerical variables, the backward (RFE) wrapper method using logistic regression, several tree-based algorithms and support vector machines (SVMs), Lasso regression and the *spearman* correlation between variables were considered to understand which would be the most relevant features.

The decision on which feature selection techniques we would use, in particular with the RFE algorithm, was highly dependent on the machine learning algorithms we would consider to attain the best model. In fact, we ended up performing the task of feature selection a few times as our models were being developed in close affinity, in order to achieve the best one given the outcomes and conclusions drawn from the several modelling stages and algorithms. As a consequence of following this approach, there were slight changes throughout the modelling tasks related to the numerical features considered for the models. The close interaction between the feature selection and modelling tasks is further described in the following section. The *k*-fold cross-validation was employed with the default value of *k*=10 folds.

A first pass on feature selection was done to select the most relevant numerical variables. This was conducted using the RFE algorithm with logistic regression, decision tree, random forest, gradient boosting (considering the default values of the algorithms) and SVM with linear kernel (separately), Lasso regression and correlation. At this point, we established to select only the most important six numerical features. The results are summarized in table in Figure 11. From it, we realize that variables such as *Comfort*, *Amenities*, *Staff*, *OnlineBooking* and *RoomSpace* should be kept, whereas the impact of others such as *Year_Birth*, *ReceptionSchedule*, *Location*, *Wifi* and *PriceQuality* should be further investigated.

## 5 Modelling

Prior to running a feature selection, our team was notified that they would be bench-marked based on a national rating (based on the F1 Score, explained later in this chapter), and that the highest standing team (per submission in *Kaggle*) would receive a fully paid 5 day-stay in Galapagos in a Resort Hotel. Immediately, our focus was torn between two visions: the hard working employee vision that wanted a well-earned break, therefore striving to produce the best fit-to-model to achieve the highest ranking; and the data scientist vision that relied in producing a step-by-step iteration that would be more time-consuming, taking longer to converge into higher F1 Scores, producing the best model while being able to explain it in every step of the process. After a long debate, our team decided to try different approaches.

We created two different teams that would focus on two different approaches: one focused on the F1 outcome "blindly" by using two subsets of features and trying to maximize the hyper-parameters on each of the models, and one set on defining a proper feature baseline with default values of hyper-parameters. This caused two different sets of iterations before settling in a baseline. Given that these are two inter-dependent activities, and based on the features initially proposed by the feature selection task, we quickly set on the following modelling and evaluation pattern: define a list of algorithms to employ; run through the selected features starting from the proposed ones; tweak each parameter to their maximum potential (using *Grid Search*); and assess and compare the performances of the models. Therefore, we ended up having five different stages within our modelling task, as described in the several subsections below. Each of these stages had a specific goal; reaching the model with best performance when adding up their outcomes.

Before moving in detail to each stage of our modelling task, we had to make some decisions regarding which machine algorithms are we considering, how are we performing grid search for tuning such algorithms, and which evaluation metrics are we using to assess each model's performance.

Regarding the choice of the algorithms, we decided to explore distinct ones belonging to different machine learning "families": neural networks (multi-layer perceptron (MLP)), SVMs, KNN, bayesian or probabilistic (naïve Bayes), decision trees and ensemble methods which are tree-based (random forest, gradient boosting and extra trees). We already know that the MLP is a fully-connected neural network that can have several hidden layers, thus allowing to understand any kind of linear and non-linear relationship between the inputs provided in the input layer and the output (prediction of the target variable); and that decision trees are greedy, non-parametric approaches built in a top-down recursive divide-and-conquer manner that select the most relevant features to the model while it is being built. However, the remaining ones were completely novel, so we wrote a short summary on each one of them to comprehensively understand their fundamentals, pros and cons (subsections 7.2, 7.3, 7.5, 7.6, 7.7, 7.8, 7.9 and 7.10).

The optimization stage then comes via *Grid Search*. Its purpose is to scan across multiple possible combinations of parameters and hyper-parameters of each algorithm in order to determine the best set of them that contributes to the best performance. First of all, it is important to mention that this procedure was performed considering the whole data set for training, thus the best combination of parameters for each algorithm is based on the performance it has solely on training data. We could have done this procedure by splitting the data set in training and validation, but, with the modelling approach used, we also keep an eye on the performance results using the validation data to account for overfitting. The several grid search runs are available in the annex, in Figures 13, 14, 15, 16 and 17. Finally, when making the choice of the best set of parameters and hyper-parameters, the focus was put on the combination that leads to the best performance and not necessarily on the computation time each algorithm takes to run for each combination. Since the algorithms Logistic Regression, KNN, SVM and naïve Bayes do not require an extensive parameter tuning, grid search was not performed for them; SVM as tested using a linear kernel and the default parameters provided by *scikit-learn* were used for the remaining ones.

Motivated by the *Kaggle* competition (or rather in the prize associated), the elected metric for evaluating each model's performance was the F1 Score, expressed in the equation below, thus providing a balance between precision and recall. Precision is related to how often is our model correct when it predicts that our customer churns, while recall is related to how often the model predicts that a customer churned when it actually churned. As precision is preferred when we want to avoid false positives and recall when we want to avoid false negatives, F1 Scores ends up considering both these aspects.

$$F1\ Score = 2 \times \frac{precision \times recall}{precision + recall} \tag{1}$$

As previously mentioned, the stratified *k*-fold cross-validation technique was used when evaluating and comparing models. Briefly, this technique splits the whole *train.csv* data set into *k* subsets of equal sizes, maintaining the proportion of the labels in the target variable in each subset, and then on each fold (iteration), it uses one subset for validation and the remaining ones for training until all portions of the original data set have been used for both training and validation. At the end, it averages the estimates of each fold to get a final score. In this case, as the data set presents moderate dimensions (15589 rows), we have chosen to use the usual *k*=10 folds (by far the most used value).

## 5.1 *1st Trial*

As mentioned, we had created two teams in order to create the best model: one focused on the F1 Score (1st Trial) and the other on replicating a step-by-step approach in feature selection trying to understand how the model would perform, improving it with each step (2nd Trial). As we were dealing with both Train and Validation F1 Scores, we would also track the difference (in relative error) between them, to ensure that we would minimize the level of overfitting (preferably under 5%).

When starting to develop the model, our team had to figure out in which direction to guide itself, so the approach of the 1st trial was to experiment the model with the features retrieved by the feature selection process. Using such features, we ran grid search in order to find the best possible hyper-parameters for this set of features and each algorithm and first recorded each model's score on both training and validation data set.

As some features suggested by the feature selection process required perhaps some further investigation to realize if they contain relevant information for the model, we ended up tweaking the suggested numerical features and testing slightly different combinations of features (8 to 12 on total) 5 times while keeping the hyper-parameter settings proposed. Again, the main focus was to keep improving the F1 Score value, and minimizing the overfitting (also mentioned as *F1-delta*, since it represents the variation in the F1 Score between the train and validation sets), while recording the models' scores and achieving the best fit.

During this process, it was possible to gain quite a few insights that helped our team to gain a better understanding of how the increasing and removal of variables affected both the F1 Score and the overfitting (table in Figure 18). In short, the more features the models take in, usually the better the F1 Score, but almost always at the cost of a high overfit. In terms of machine learning, considering that the F1 Scores in the training data of SVM, Logistic Regression (around 0.8), KNN (around 0.9) and naïve Bayes (not recorded, eventually discarded) were considerably low when compared to the other models used despite the low overfitting (maximum of 4.3% for KNN recorded), we decided to drop them when moving forward to the next stage. Gradient boosting, random forest and the MLP got exceptional F1 Scores for training data, with the best one of 0.979 being attained by the gradient boosting algorithm. For these three models, overall in all attempts the overfit was kept below 5%; yet, at the end of this "optimization" stage, MLP has shown a lower overfit (only around 2.9%) compared to the ensemble models (between 4% and 5%). The best fits attained for each of these three models composed the first *Kaggle* submissions.

## 5.2 *2nd Trial*

As per the 2nd Trial, the second team was savvy enough to start with only tree-based models, ensemble models and Neural Networks. Going into the second stage, only the random forest, gradient boosting and MLP algorithms were considered. After getting an overall idea of the behaviour of each algorithm, at this stage we aimed to explore which are the essential features leading to the best fits, again based on the feature selection suggestions and on the expertise gained from the previous stage, while trying to keep the F1 Score of the training set high and decrease the performance loss on the validation set. However, given that in general we were only tweaking around the same features, we did not performed grid search at this stage as we kept the models' parameters to default.

We perceived that the most consistent features, this is, the ones leading to consequently good fits, were the four categorical ones - *Longevity*, *TypeTravel*, *RoomType* and *Gender* - and some numerical ones - *Comfort*, *Amenities*, *OnlineBooking* and *PriceQuality*. This set of eight features was called the *Feature Baseline*. At this time, we also observed that, with the introduction of fewer variables into the model, both F1 Score on the training set and *F1-delta* value decrease (table in Figure 19). The greater variations detected while performing the successive tweaks of the predictors were found for the random forest algorithm (the minimum and maximum F1 Scores in the training data varied from 0.932 and 0.998), whereas gradient boosting and neural networks appeared to be more robust to these changes in the predictors. Consequently, in the next stages, less attention was put on the random forest algorithm.

The models obtained in this stage did not, however, lead to any *Kaggle* submission as our main goal here was defining the set of baseline features.

## 5.3 *3rd Trial - Feature Baseline*

At this stage, using the set of eight variables that were considered as the baseline features - *Longevity*, *TypeTravel*, *RoomType* and *Gender*, *Comfort*, *Amenities*, *OnlineBooking* and *PriceQuality* - grid search was performed in order to fine tune the parameters and hyper-parameters of the random forest, gradient boosting and MLP algorithms. After this, the performance of each one of the models on the training and validation sets was recorded.

We were able to find that the right tuning of each model parameters and hyper-parameters allowed us to obtain models that better fit our data. The F1 Score on both training and validation sets was increased for gradient boosting and MLP, and the their overfit was now between *1% and 2%*. By taking an overall look at the results (Table 1 and Figure 20), the performances of our models were at this point very similar, such as they have been normalized. Yet, again we believed we could do best before performing another *Kaggle* submission, so we moved on to the next stage.

## 5.4 *4th Trial - Feature Baseline 2.0*

At this stage, besides the eight baseline features, we considered *Location* and *Staff*, as they were previously suggested by the feature selection process and that they have shown to improve the models' fit in the former stages. Focusing only on gradient boosting and MLP, we performed grid search and recorded the performances.

Comparing the results obtained (Table 1 and Figure 21) against to the ones achieved in former trials, we can see that there is an improvement of the F1 Scores in both training and validation sets against the ones obtained in the 2nd and 3rd trials, but they were not quite as good as some of the F1 Scores reached in the 1st trial. Plus, we did achieved low percentages of overfitting. Summing up the performances, we reached F1 Scores of 0.956 and 0.928 for the training and validation sets, respectively, with the gradient boosting algorithm; and F1 Scores of 0.939 and 0.928 for the training and and validation sets, respectively, with the MLP. The best fit was used in the *Kaggle* submission. A quick comparison was also made with restricting the range of available values in the satisfaction features to [1-5], which ultimately was dropped due to lower F1 Scores.

The choice of either of these models would have been a wise one, as these values are quite sturdy and not prone to overfitting. However, we are greedy data scientists whose main goal is to win the *Kaggle* competition, thus securing the coveted prize!

## 5.5 *5th Trial - Feature Baseline 3.0*

We decided to double down with a gambit: a more extreme model, which would provide us with a higher F1 Score, while sacrificing a bit on the *F1-delta*, thus being more exposed to overfitting. Given the great performance of ensemble tree-based methods, in this last stage we explored the extra trees algorithm. Considering the overall improvements from the 3rd to the 4th trial, as one last chance to attain the best possible model we ended up considering a larger set of features. In order to to that, we performed another feature selection process, this time focusing on 10 numerical ones. We also added the RFE algorithm using extra trees, since we are now running it as well. The results suggested that, plus the previously selected categorical variables *Longevity*, *TypeTravel*, *RoomType* and *Gender*, we should at this time consider the following numerical ones: *Year_Birth*, *Comfort*, *ReceptionSchedule*, *FoodDrink*, *Location*, *Amenities*, *Staff*, *OnlineBooking*, *PriceQuality*, *RoomSpace* and *CheckIn*. The results of this feature selection process are summarized on the table in Figure 12 and Figure 22. On top of these suggestions, We performed grid search and recorded the performance of the algorithms, focusing on MLP and extra trees as an alternative to gradient boosting.

We ended up observing that extra trees provided the best F1 Scores for both training (0.989) and validation (0.941), the highest recorded in our sessions! By considering a lot more features at this stage, we were indeed able to improve the F1 Scores especially at the training level; but again at the cost of higher percentages of overfitting that, however, still lie below 5% (Table 1). This consisted of the last *Kaggle* submission, in a ”go big or go home” fashion.

Every feature used was selected according to the feature selection process, except for *CheckOut*. The reason for inclusion is not entirely explainable: the selection process highlighted *RoomSpace* as a variable to be chosen. After performing both checks with either variable, the F1 Score with *CheckOut* produced higher scores and was ultimately chosen. We are unsure on why this is the scenario, whether if it's related to proportion of churn associated with the variable *RoomSpace*, especially in lower bounds (when it takes a value of 0); with higher correlation between *CheckOut* and the other chosen variables, similar to the ratings defined in the first assignments; or simply divine inspiration by our scientists, in order to explore every single option trying to secure a well-earned break! Regardless of reason, our results were able to propose a higher score and we happily embraced it!

## 6 Conclusions

Ultimately, regardless of our *Kaggle* score and prize, we were extremely satisfied with the model we produced, not only due to the consistency in the process carried out but also because of the evaluation scores resulting from our submissions. We were also able to explore different models outside from the ”standard” packages in *scikit-learn* and understand their fit to purpose.

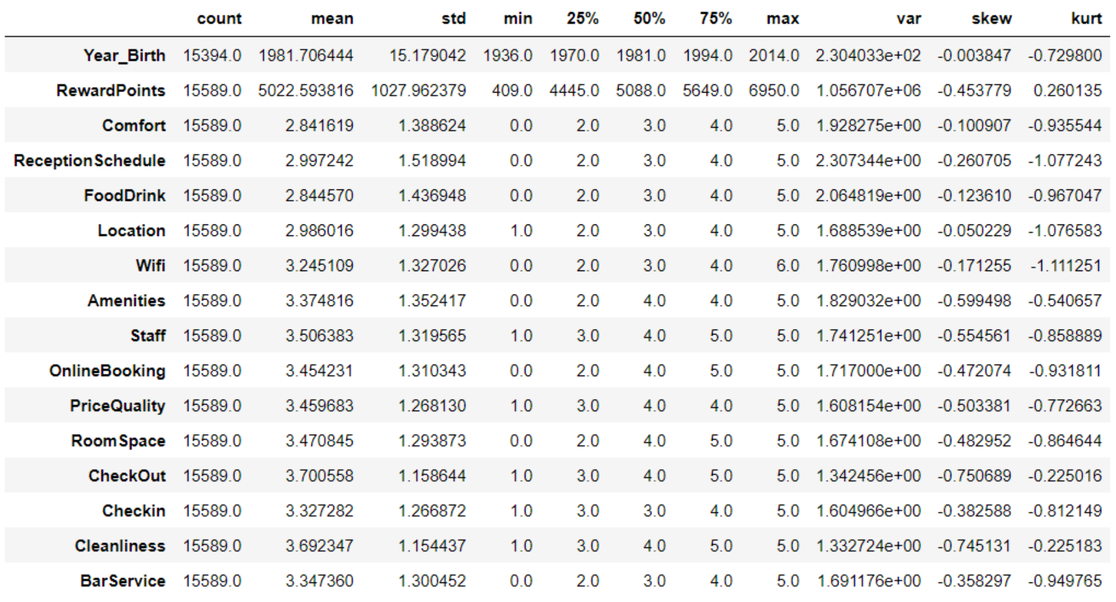
When we proposed our findings to our board of directors, the only natural question after our explanation was whether we were successful in building a J.A.R.V.I.S. model to predict customer churning. We looked at each other, partially amused, partially frustrated, since we had documented all of our findings trying to explain the challenges to produce our model. We briefly asked for 5 minutes and discussed a better way to showcase our findings.

We summoned the board of directors to one of the available meeting rooms and showed up all dressed in Iron-man costumes. We then proceeded to explain how our model would be able to predict the customer churn based on their ratings, while integrating that with our stellar marketing plan. Similar to J.A.R.V.I.S., we named our system *Reservatron 15k*, promising that the number in front had nothing to do with the cost we had spent during the last couple of weeks. We slid in some indicators, such as the F1 Score and *F1-delta*, while keeping it as light as possible, and to our immense pleasure, they approved the directive and *Reservatron 15k* would now become the main model to be used in *Book.me*. Next steps would be to implement it alongside our marketing strategies to overtake *Booking.com* as the preferred option for hotel booking.

As for the submission values in *Kaggle*, we would have to wait until the 19th of June to see the final score. But let's just say that our team would be waiting in swim shorts and bikinis, as we would be confident in our standing and produced values!

# 7 Annex

## 7.1 *Figures and tables*



| | count | mean | std | min | 25% | 50% | 75% | max | var | skew | kurt |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Year_Birth** | 15394.0 | 1981.706444 | 15.179042 | 1936.0 | 1970.0 | 1981.0 | 1994.0 | 2014.0 | 2.304033e+02 | -0.003847 | -0.729800 |
| **RewardPoints** | 15589.0 | 5022.593816 | 1027.962379 | 409.0 | 4445.0 | 5088.0 | 5649.0 | 6950.0 | 1.056707e+06 | -0.453779 | 0.260135 |
| **Comfort** | 15589.0 | 2.841619 | 1.388624 | 0.0 | 2.0 | 3.0 | 4.0 | 5.0 | 1.928275e+00 | -0.100907 | -0.935544 |
| **ReceptionSchedule** | 15589.0 | 2.997242 | 1.518994 | 0.0 | 2.0 | 3.0 | 4.0 | 5.0 | 2.307344e+00 | -0.260705 | -1.077243 |
| **FoodDrink** | 15589.0 | 2.844570 | 1.436948 | 0.0 | 2.0 | 3.0 | 4.0 | 5.0 | 2.064819e+00 | -0.123610 | -0.967047 |
| **Location** | 15589.0 | 2.986016 | 1.299438 | 1.0 | 2.0 | 3.0 | 4.0 | 5.0 | 1.688539e+00 | -0.050229 | -1.076583 |
| **Wifi** | 15589.0 | 3.245109 | 1.327026 | 0.0 | 2.0 | 3.0 | 4.0 | 6.0 | 1.760998e+00 | -0.171255 | -1.111251 |
| **Amenities** | 15589.0 | 3.374816 | 1.352417 | 0.0 | 2.0 | 4.0 | 4.0 | 5.0 | 1.829032e+00 | -0.599498 | -0.540657 |
| **Staff** | 15589.0 | 3.506383 | 1.319565 | 1.0 | 3.0 | 4.0 | 5.0 | 5.0 | 1.741251e+00 | -0.554561 | -0.858889 |
| **OnlineBooking** | 15589.0 | 3.454231 | 1.310343 | 0.0 | 2.0 | 4.0 | 5.0 | 5.0 | 1.717000e+00 | -0.472074 | -0.931811 |
| **PriceQuality** | 15589.0 | 3.459683 | 1.268130 | 1.0 | 3.0 | 4.0 | 4.0 | 5.0 | 1.608154e+00 | -0.503381 | -0.772663 |
| **RoomSpace** | 15589.0 | 3.470845 | 1.293873 | 0.0 | 2.0 | 4.0 | 5.0 | 5.0 | 1.674108e+00 | -0.482952 | -0.864644 |
| **CheckOut** | 15589.0 | 3.700558 | 1.158644 | 1.0 | 3.0 | 4.0 | 5.0 | 5.0 | 1.342456e+00 | -0.750689 | -0.225016 |
| **Checkin** | 15589.0 | 3.327282 | 1.266872 | 1.0 | 3.0 | 3.0 | 4.0 | 5.0 | 1.604966e+00 | -0.382588 | -0.812149 |
| **Cleanliness** | 15589.0 | 3.692347 | 1.154437 | 1.0 | 3.0 | 4.0 | 5.0 | 5.0 | 1.332724e+00 | -0.745131 | -0.225183 |
| **BarService** | 15589.0 | 3.347360 | 1.300452 | 0.0 | 2.0 | 3.0 | 4.0 | 5.0 | 1.691176e+00 | -0.358297 | -0.949765 |

Figure 1: Statistical description of the numerical variables provided by the methods *describe()*, *var()*, *skew()* and *kurt()*. *var* stands for variance, *skew* for skewness and *kurt* for kurtosis.

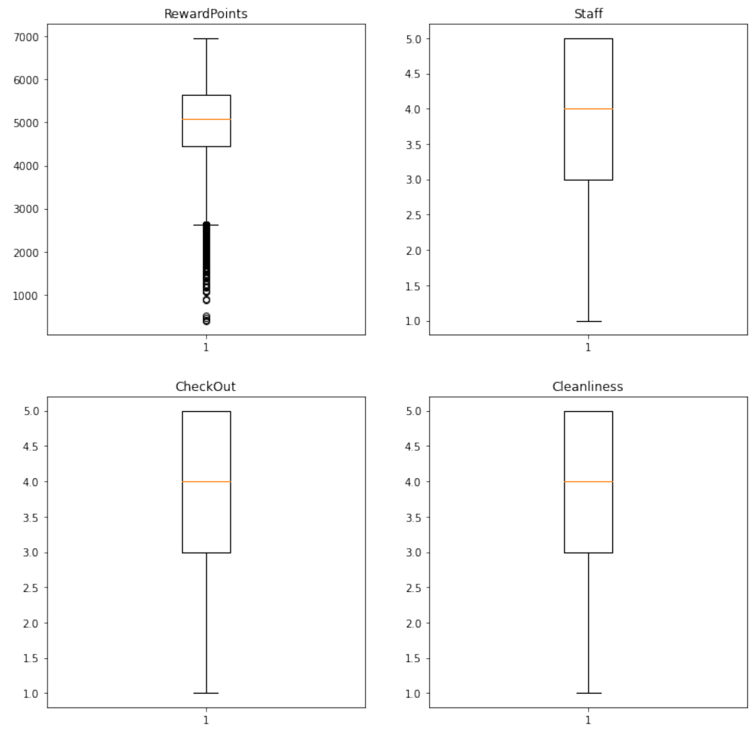


Figure 2: Visual representation of possible outliers in the variables *RewardPoints*, *Staff*, *Checkout* and *Cleanliness* using box plots.

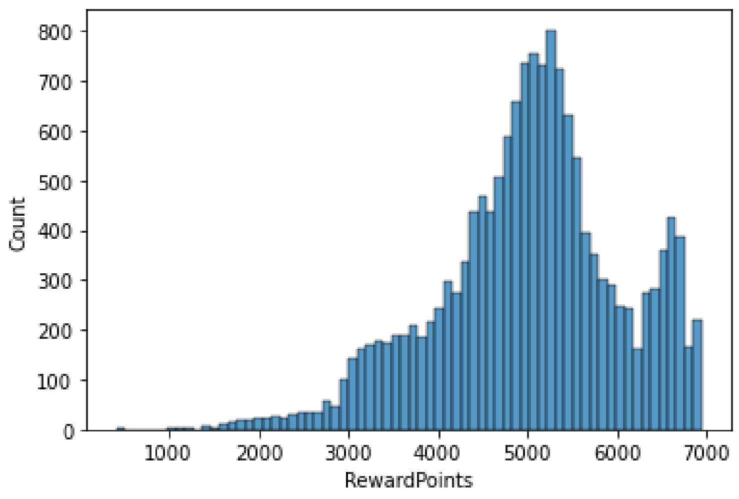


Figure 3: Histogram for the variable *RewardPoints*. Note there are little counts for a numer of reward points below 1000, which could suggest the existence of outliers.

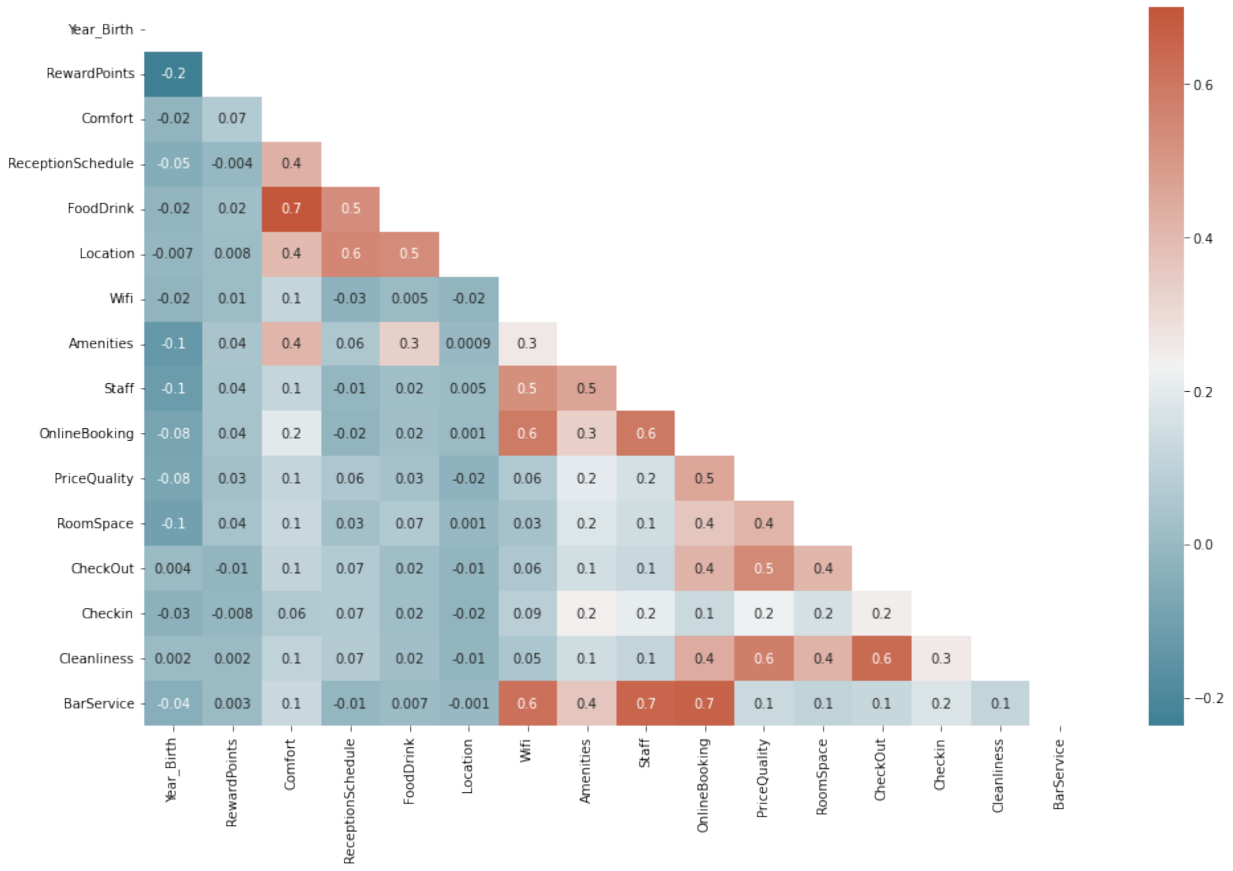
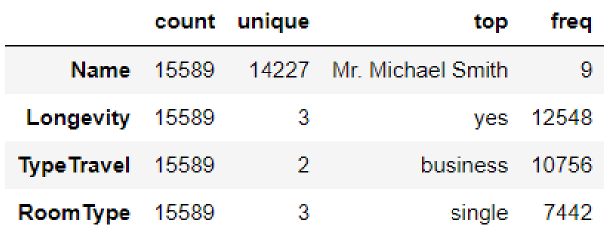


Figure 4: Correlation matrix highlighting the degree of correlation between the several numerical variables. Spearman correlation was used.



| | count | unique | top | freq |
|---|---|---|---|---|
| **Name** | 15589 | 14227 | Mr. Michael Smith | 9 |
| **Longevity** | 15589 | 3 | yes | 12548 |
| **TypeTravel** | 15589 | 2 | business | 10756 |
| **RoomType** | 15589 | 3 | single | 7442 |

Figure 5: Statistical description of the categorical variables provided by the method *describe()*.

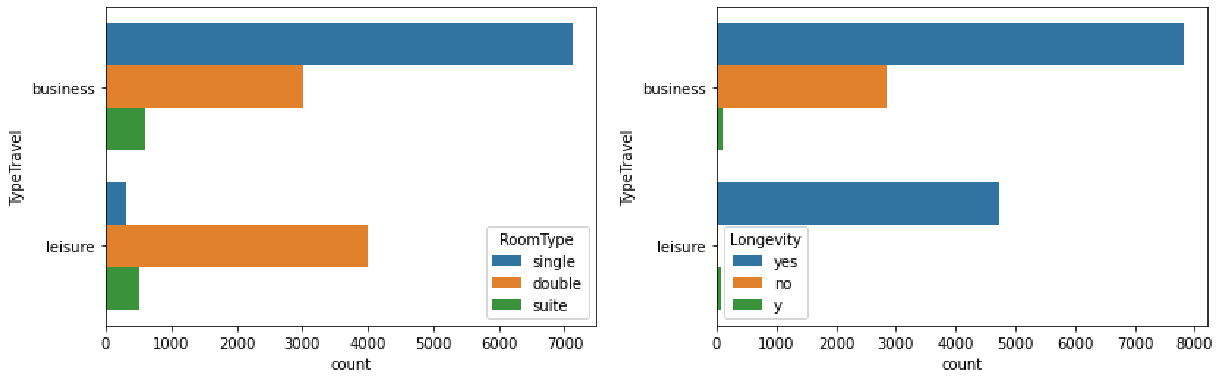


Figure 6: Countplots for the variables *TypeTravel* and *RoomType* (on the left) and for the variables *TypeTravel* and *Longevity* (on the right).

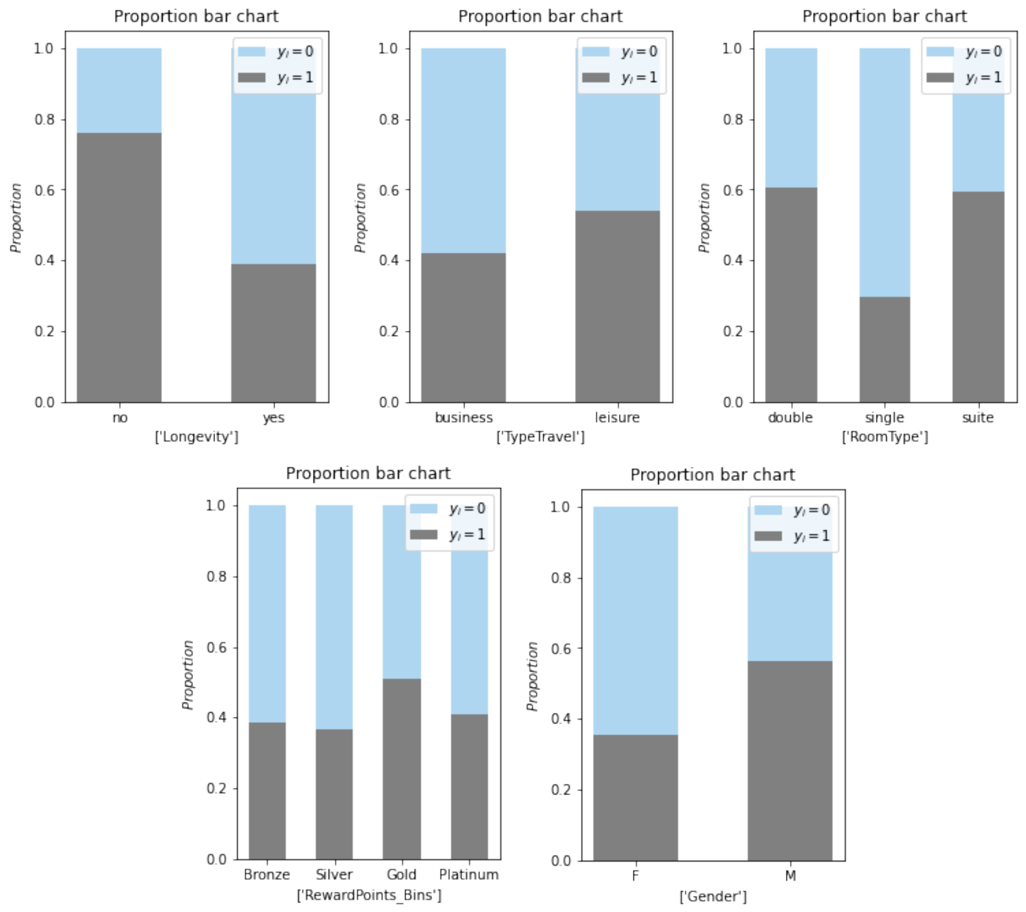


Figure 7: Proportion bar charts: understanding the weight of the target variable on the several categorical variables originally available on the initial data set.

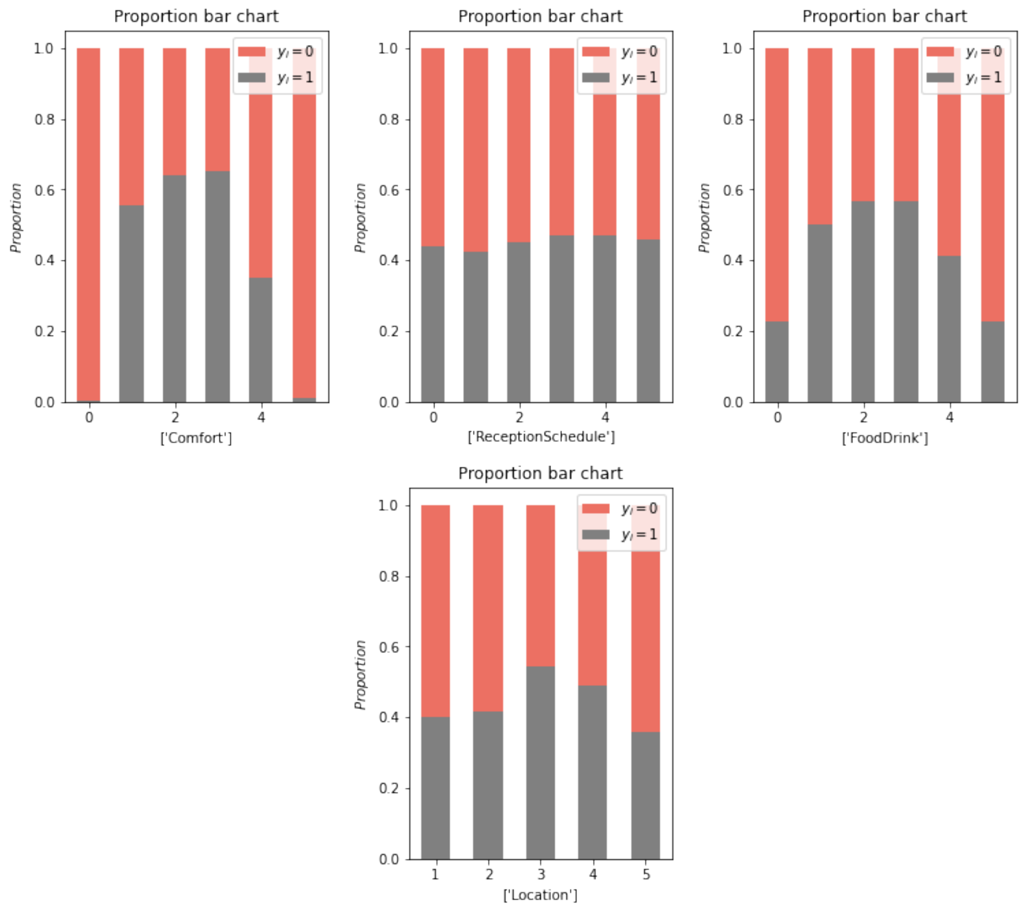


Figure 8: Proportion bar charts: understanding the weight of the target variable on the numerical variables included in *Rating1 - Comfort*, *ReceptionSchedule*, *FoodDrink* and *Location*.

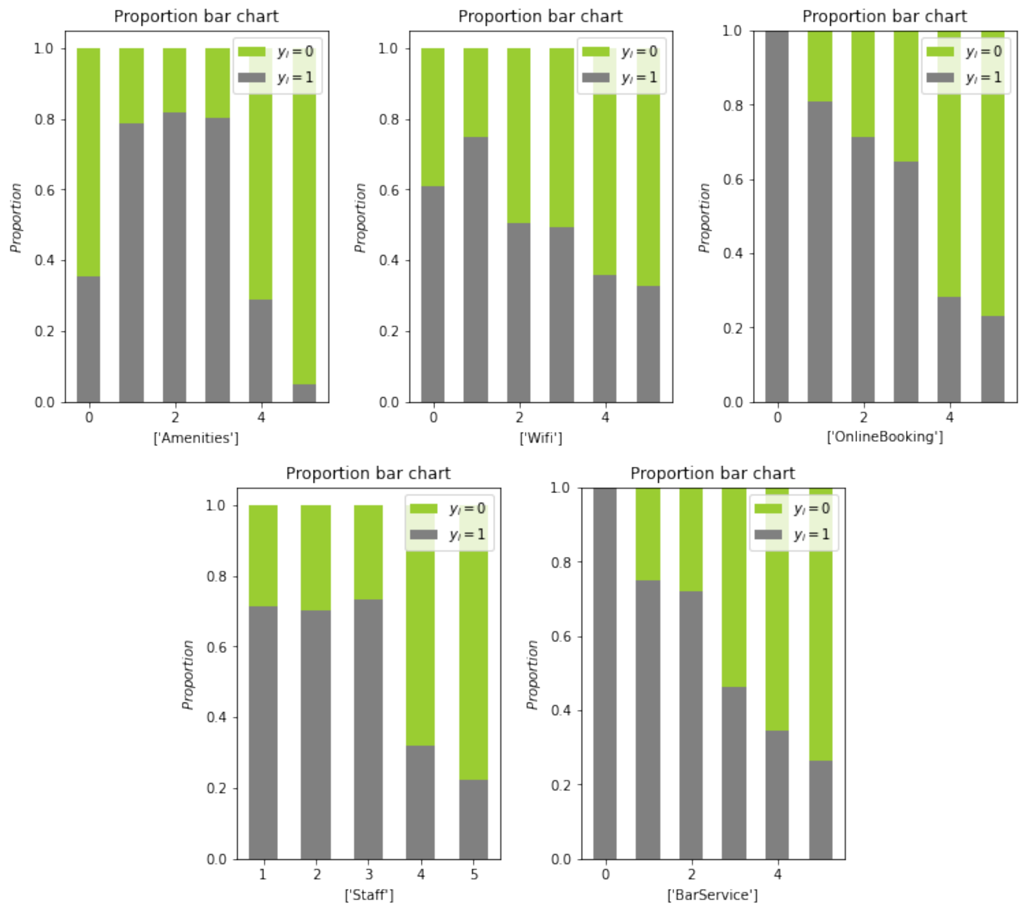


Figure 9: Proportion bar charts: understanding the weight of the target variable on the numerical variables included in *Rating2 - Amenities*, *Wifi*, *OnlineBooking*, *Staff* and *BarService*.

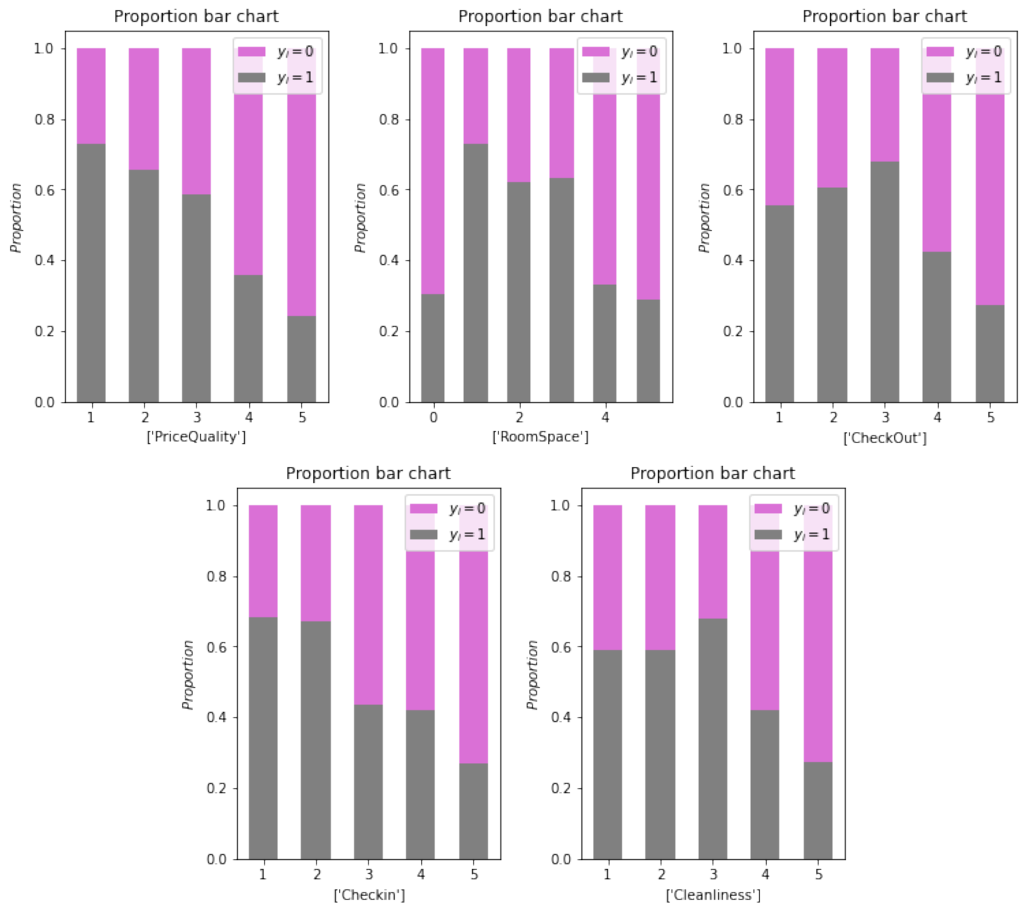
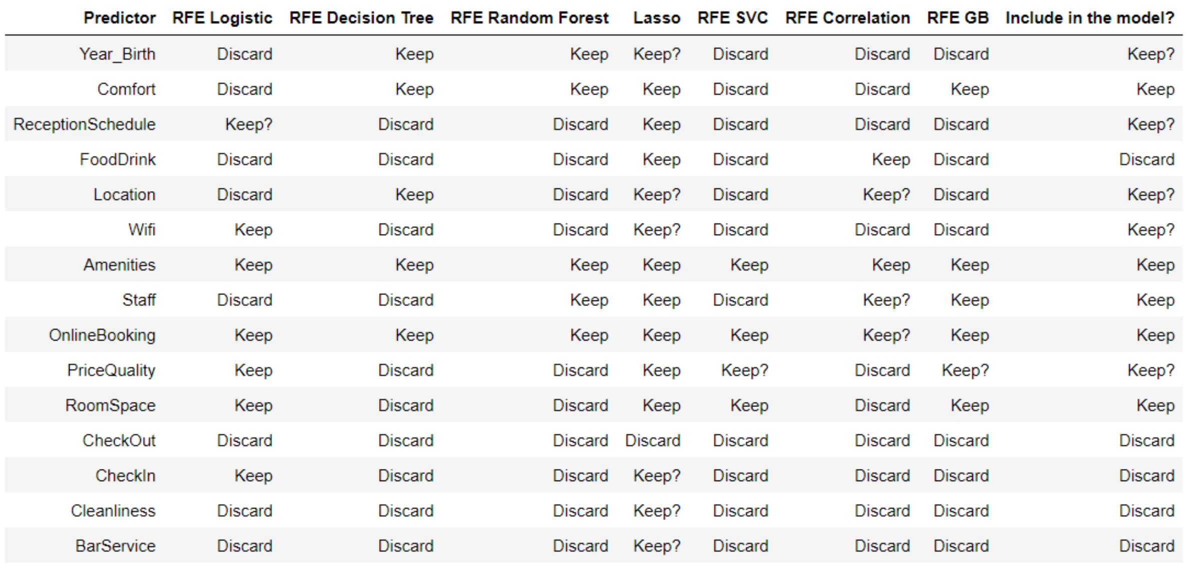
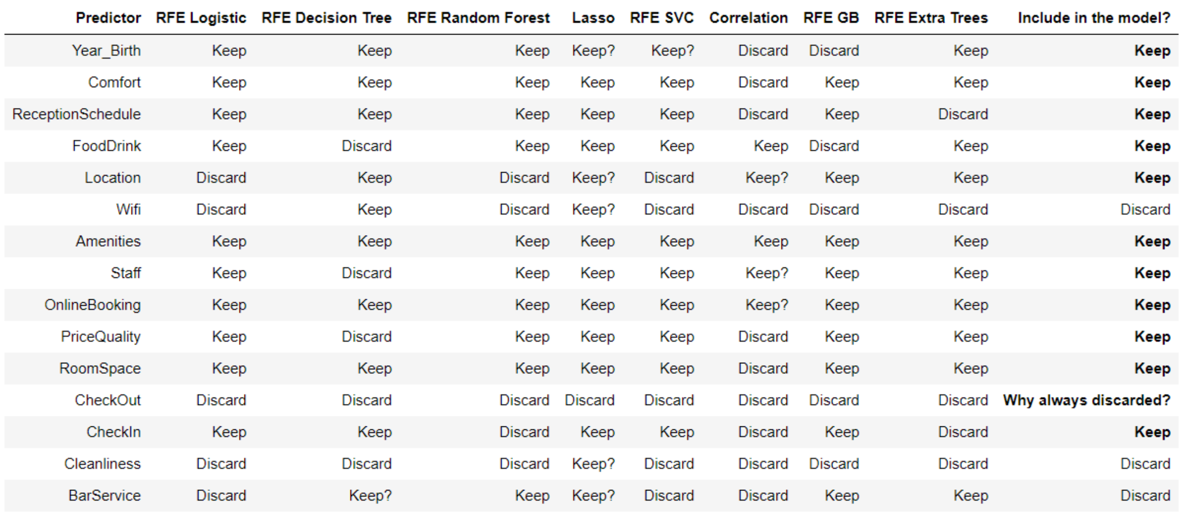


Figure 10: Proportion bar charts: understanding the weight of the target variable on the numerical variables included in *Rating3* - *PriceQuality*, *RoomSpace*, *CheckOut*, *CheckIn* and *Cleanliness*.



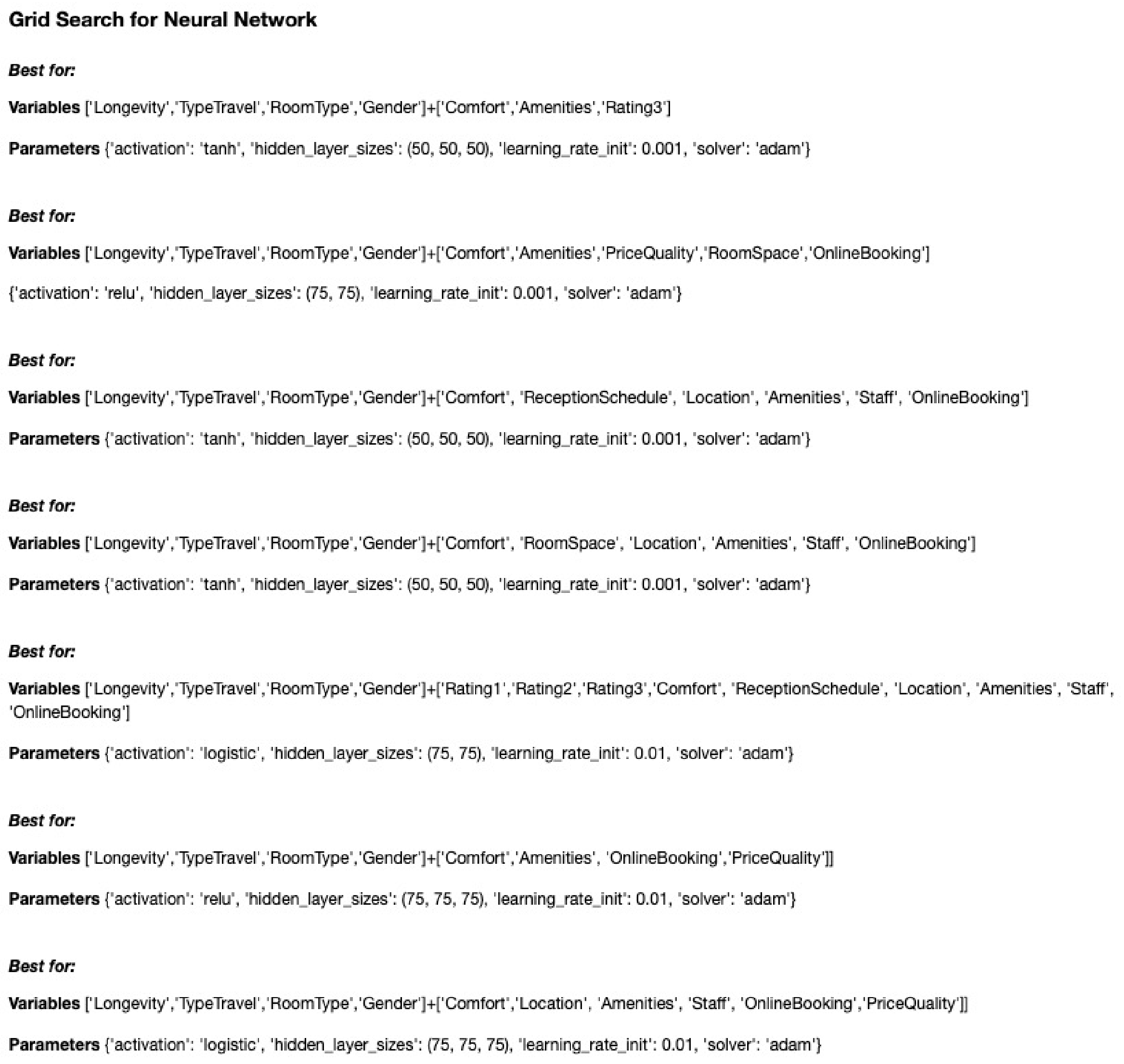
| Predictor | RFE Logistic | RFE Decision Tree | RFE Random Forest | Lasso | RFE SVC | RFE Correlation | RFE GB | Include in the model? |
|---|---|---|---|---|---|---|---|---|
| Year_Birth | Discard | Keep | Keep | Keep? | Discard | Discard | Discard | Keep? |
| Comfort | Discard | Keep | Keep | Keep | Discard | Discard | Keep | Keep |
| ReceptionSchedule | Keep? | Discard | Discard | Keep | Discard | Discard | Discard | Keep? |
| FoodDrink | Discard | Discard | Discard | Keep | Discard | Keep | Discard | Discard |
| Location | Discard | Keep | Discard | Keep? | Discard | Keep? | Discard | Keep? |
| Wifi | Keep | Discard | Discard | Keep? | Discard | Discard | Discard | Keep? |
| Amenities | Keep | Keep | Keep | Keep | Keep | Keep | Keep | Keep |
| Staff | Discard | Discard | Keep | Keep | Discard | Keep? | Keep | Keep |
| OnlineBooking | Keep | Keep | Keep | Keep | Keep | Keep? | Keep | Keep |
| PriceQuality | Keep | Discard | Discard | Keep | Keep? | Discard | Keep? | Keep? |
| RoomSpace | Keep | Discard | Discard | Keep | Keep | Discard | Keep | Keep |
| CheckOut | Discard | Discard | Discard | Discard | Discard | Discard | Discard | Discard |
| CheckIn | Keep | Discard | Discard | Keep? | Discard | Discard | Discard | Discard |
| Cleanliness | Discard | Discard | Discard | Keep? | Discard | Discard | Discard | Discard |
| BarService | Discard | Discard | Discard | Keep? | Discard | Discard | Discard | Discard |

Figure 11: The results of the first ”iteration” of a feature selection process.



| Predictor | RFE Logistic | RFE Decision Tree | RFE Random Forest | Lasso | RFE SVC | Correlation | RFE GB | RFE Extra Trees | Include in the model? |
|---|---|---|---|---|---|---|---|---|---|
| Year_Birth | Keep | Keep | Keep | Keep? | Keep? | Discard | Discard | Keep | **Keep** |
| Comfort | Keep | Keep | Keep | Keep | Keep | Discard | Keep | Keep | **Keep** |
| ReceptionSchedule | Keep | Keep | Keep | Keep | Keep | Discard | Keep | Discard | **Keep** |
| FoodDrink | Keep | Discard | Keep | Keep | Keep | Keep | Discard | Keep | **Keep** |
| Location | Discard | Keep | Discard | Keep? | Discard | Keep? | Keep | Keep | **Keep** |
| Wifi | Discard | Keep | Discard | Keep? | Discard | Discard | Discard | Discard | Discard |
| Amenities | Keep | Keep | Keep | Keep | Keep | Keep | Keep | Keep | **Keep** |
| Staff | Keep | Discard | Keep | Keep | Keep | Keep? | Keep | Keep | **Keep** |
| OnlineBooking | Keep | Keep | Keep | Keep | Keep | Keep? | Keep | Keep | **Keep** |
| PriceQuality | Keep | Discard | Keep | Keep | Keep | Discard | Keep | Keep | **Keep** |
| RoomSpace | Keep | Keep | Keep | Keep | Keep | Discard | Keep | Keep | **Keep** |
| CheckOut | Discard | Discard | Discard | Discard | Discard | Discard | Discard | Discard | **Why always discarded?** |
| CheckIn | Keep | Keep | Discard | Keep | Keep | Discard | Keep | Discard | **Keep** |
| Cleanliness | Discard | Discard | Discard | Keep? | Discard | Discard | Discard | Discard | Discard |
| BarService | Discard | Keep? | Keep | Keep? | Discard | Discard | Keep | Keep | Discard |

Figure 12: The results of the second ”iteration” of a feature selection process.



**Grid Search for Neural Network**

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort','Amenities','Rating3']

**Parameters** {'activation': 'tanh', 'hidden_layer_sizes': (50, 50, 50), 'learning_rate_init': 0.001, 'solver': 'adam'}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort','Amenities','PriceQuality','RoomSpace','OnlineBooking']

{'activation': 'relu', 'hidden_layer_sizes': (75, 75), 'learning_rate_init': 0.001, 'solver': 'adam'}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort', 'ReceptionSchedule', 'Location', 'Amenities', 'Staff', 'OnlineBooking']

**Parameters** {'activation': 'tanh', 'hidden_layer_sizes': (50, 50, 50), 'learning_rate_init': 0.001, 'solver': 'adam'}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort', 'RoomSpace', 'Location', 'Amenities', 'Staff', 'OnlineBooking']

**Parameters** {'activation': 'tanh', 'hidden_layer_sizes': (50, 50, 50), 'learning_rate_init': 0.001, 'solver': 'adam'}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Rating1','Rating2','Rating3','Comfort', 'ReceptionSchedule', 'Location', 'Amenities', 'Staff', 'OnlineBooking']

**Parameters** {'activation': 'logistic', 'hidden_layer_sizes': (75, 75), 'learning_rate_init': 0.01, 'solver': 'adam'}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort','Amenities', 'OnlineBooking','PriceQuality']]

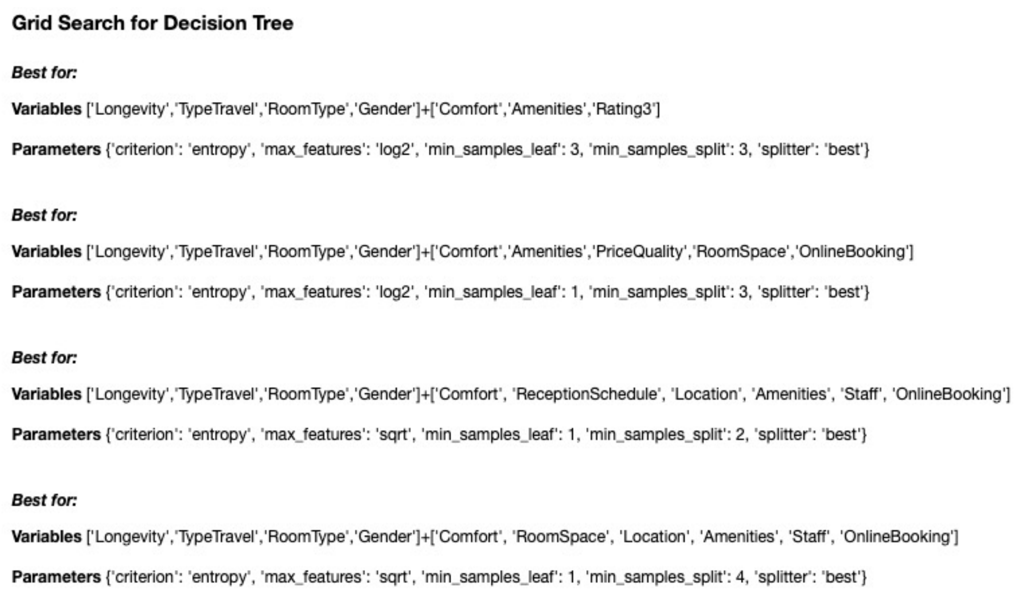
**Parameters** {'activation': 'relu', 'hidden_layer_sizes': (75, 75, 75), 'learning_rate_init': 0.01, 'solver': 'adam'}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort','Location', 'Amenities', 'Staff', 'OnlineBooking','PriceQuality']]

**Parameters** {'activation': 'logistic', 'hidden_layer_sizes': (75, 75, 75), 'learning_rate_init': 0.01, 'solver': 'adam'}

Figure 13: Grid search results for neural networks (MLP), for distinct combinations of predictor variables.



**Grid Search for Decision Tree**

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort','Amenities','Rating3']

**Parameters** {'criterion': 'entropy', 'max_features': 'log2', 'min_samples_leaf': 3, 'min_samples_split': 3, 'splitter': 'best'}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort','Amenities','PriceQuality','RoomSpace','OnlineBooking']

**Parameters** {'criterion': 'entropy', 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 3, 'splitter': 'best'}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort', 'ReceptionSchedule', 'Location', 'Amenities', 'Staff', 'OnlineBooking']

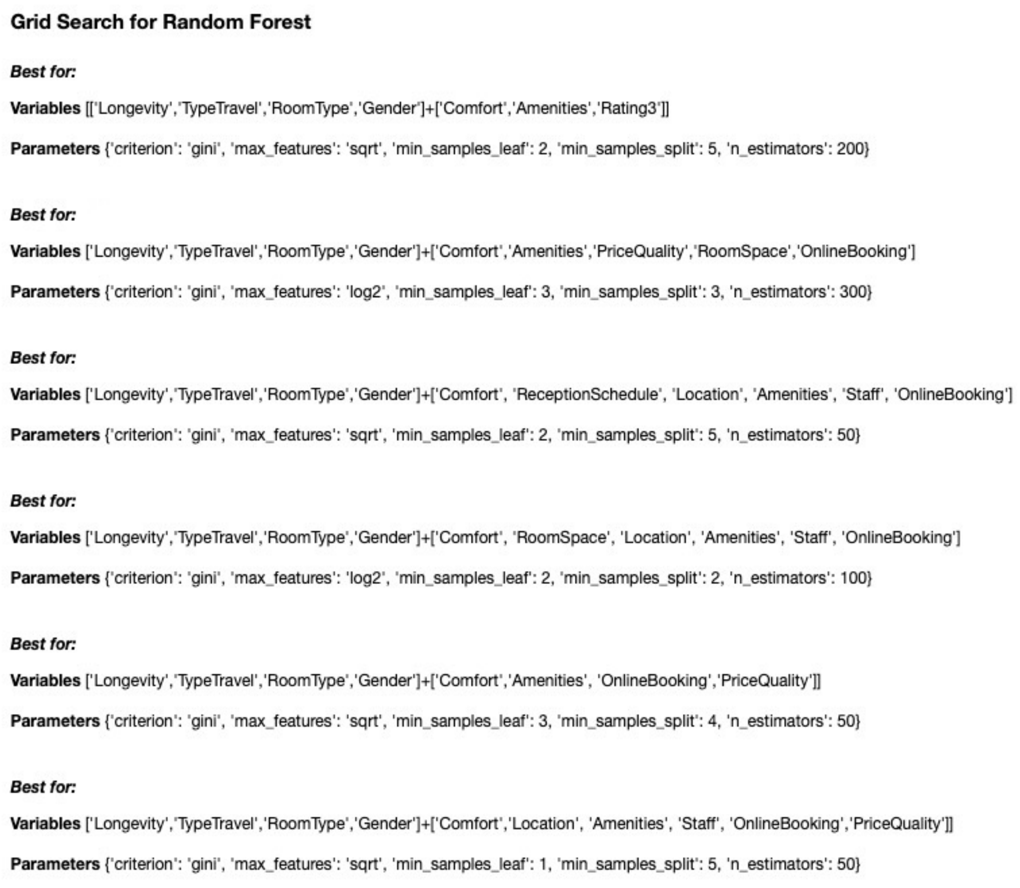
**Parameters** {'criterion': 'entropy', 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter': 'best'}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort', 'RoomSpace', 'Location', 'Amenities', 'Staff', 'OnlineBooking']

**Parameters** {'criterion': 'entropy', 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 4, 'splitter': 'best'}

Figure 14: Grid search results for decision trees, for distinct combinations of predictor variables.



**Grid Search for Random Forest**

***Best for:***

**Variables** [['Longevity','TypeTravel','RoomType','Gender']+['Comfort','Amenities','Rating3']]

**Parameters** {'criterion': 'gini', 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 200}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort','Amenities','PriceQuality','RoomSpace','OnlineBooking']

**Parameters** {'criterion': 'gini', 'max_features': 'log2', 'min_samples_leaf': 3, 'min_samples_split': 3, 'n_estimators': 300}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort', 'ReceptionSchedule', 'Location', 'Amenities', 'Staff', 'OnlineBooking']

**Parameters** {'criterion': 'gini', 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 50}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort', 'RoomSpace', 'Location', 'Amenities', 'Staff', 'OnlineBooking']

**Parameters** {'criterion': 'gini', 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort','Amenities', 'OnlineBooking','PriceQuality']]

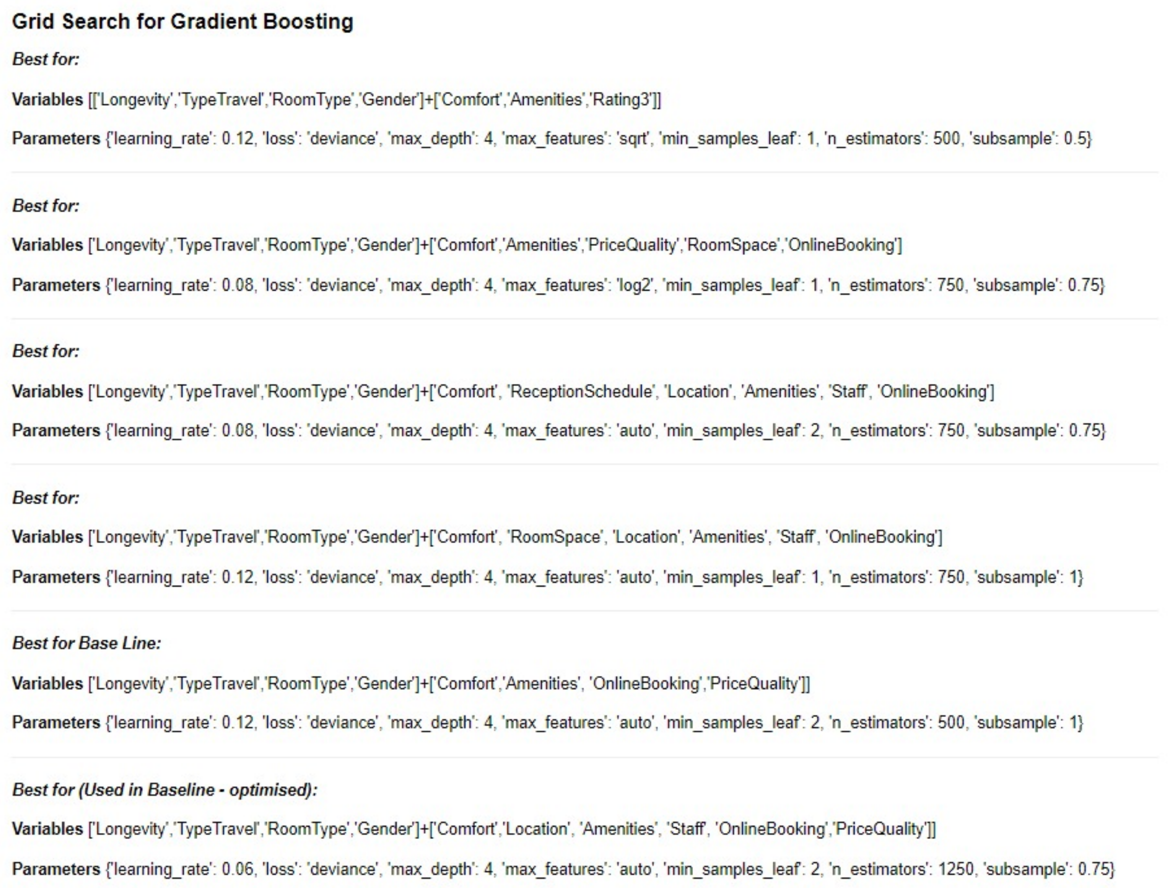
**Parameters** {'criterion': 'gini', 'max_features': 'sqrt', 'min_samples_leaf': 3, 'min_samples_split': 4, 'n_estimators': 50}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort','Location', 'Amenities', 'Staff', 'OnlineBooking','PriceQuality']]

**Parameters** {'criterion': 'gini', 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 5, 'n_estimators': 50}

Figure 15: Grid search results for random forest, for distinct combinations of predictor variables.



**Grid Search for Gradient Boosting**

***Best for:***

**Variables** [['Longevity','TypeTravel','RoomType','Gender']+['Comfort','Amenities','Rating3']]

**Parameters** {'learning_rate': 0.12, 'loss': 'deviance', 'max_depth': 4, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'n_estimators': 500, 'subsample': 0.5}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort','Amenities','PriceQuality','RoomSpace','OnlineBooking']

**Parameters** {'learning_rate': 0.08, 'loss': 'deviance', 'max_depth': 4, 'max_features': 'log2', 'min_samples_leaf': 1, 'n_estimators': 750, 'subsample': 0.75}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort', 'ReceptionSchedule', 'Location', 'Amenities', 'Staff', 'OnlineBooking']

**Parameters** {'learning_rate': 0.08, 'loss': 'deviance', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 2, 'n_estimators': 750, 'subsample': 0.75}

***Best for:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort', 'RoomSpace', 'Location', 'Amenities', 'Staff', 'OnlineBooking']

**Parameters** {'learning_rate': 0.12, 'loss': 'deviance', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 1, 'n_estimators': 750, 'subsample': 1}

***Best for Base Line:***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort','Amenities', 'OnlineBooking','PriceQuality']]

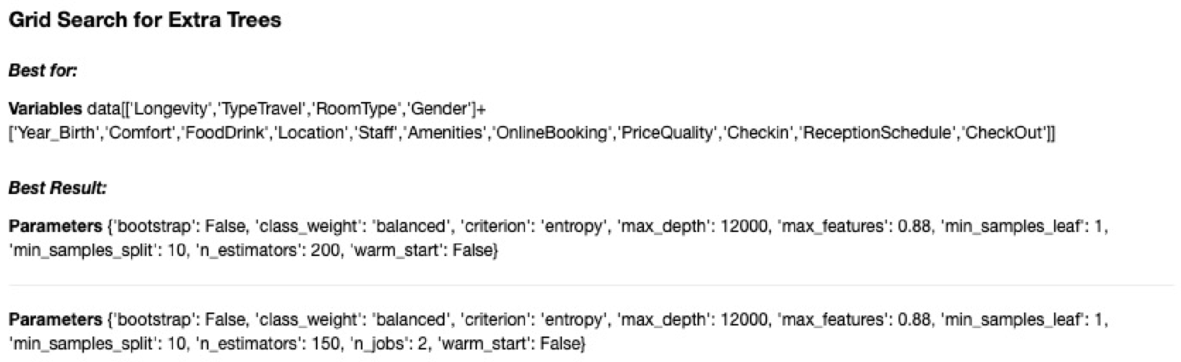
**Parameters** {'learning_rate': 0.12, 'loss': 'deviance', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 2, 'n_estimators': 500, 'subsample': 1}

***Best for (Used in Baseline - optimised):***

**Variables** ['Longevity','TypeTravel','RoomType','Gender']+['Comfort','Location', 'Amenities', 'Staff', 'OnlineBooking','PriceQuality']]

**Parameters** {'learning_rate': 0.06, 'loss': 'deviance', 'max_depth': 4, 'max_features': 'auto', 'min_samples_leaf': 2, 'n_estimators': 1250, 'subsample': 0.75}

Figure 16: Grid search results for gradient boosting, for distinct combinations of predictor variables.



**Grid Search for Extra Trees**

***Best for:***

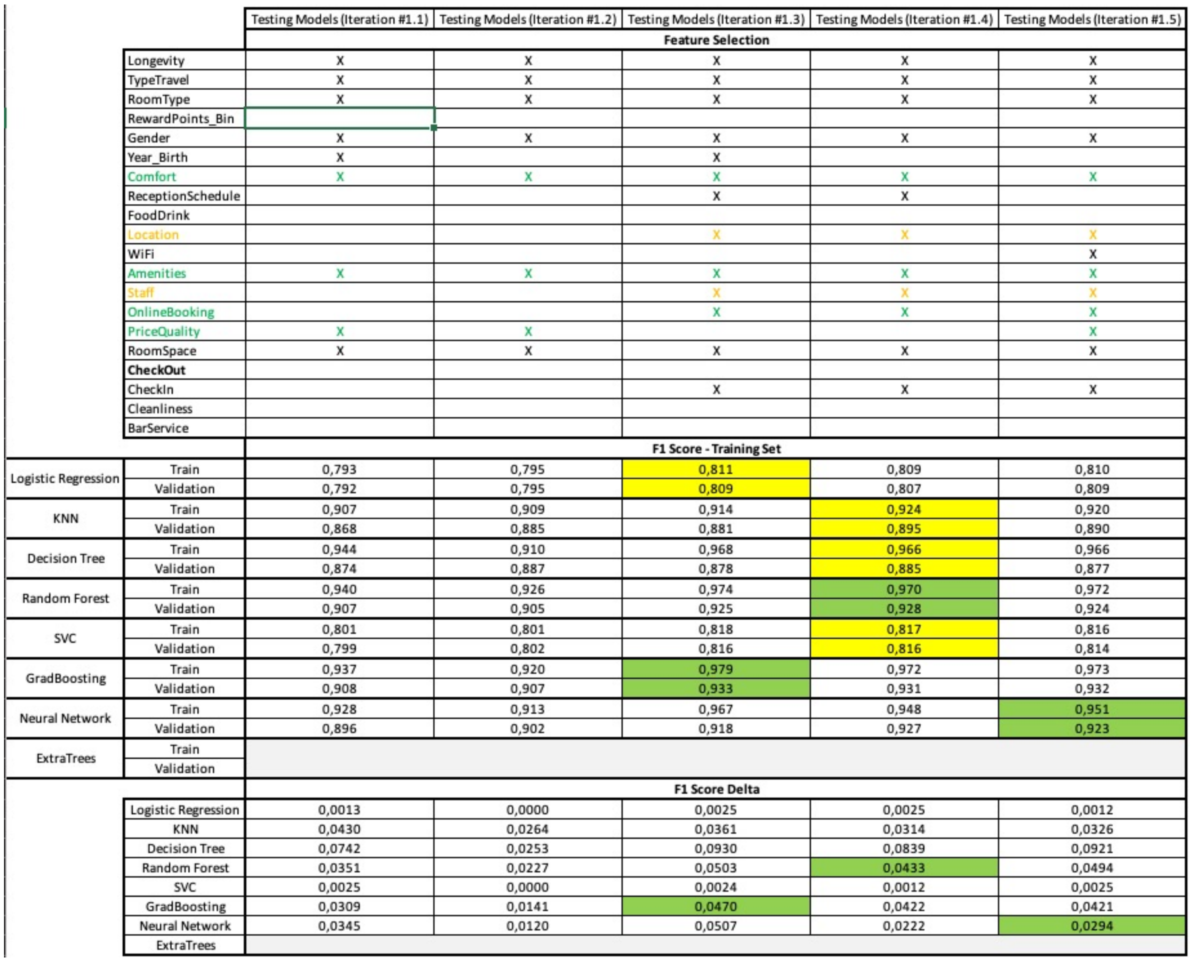
**Variables** data[['Longevity','TypeTravel','RoomType','Gender']+
['Year_Birth','Comfort','FoodDrink','Location','Staff','Amenities','OnlineBooking','PriceQuality','Checkin','ReceptionSchedule','CheckOut']]

***Best Result:***

**Parameters** {'bootstrap': False, 'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 12000, 'max_features': 0.88, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 200, 'warm_start': False}

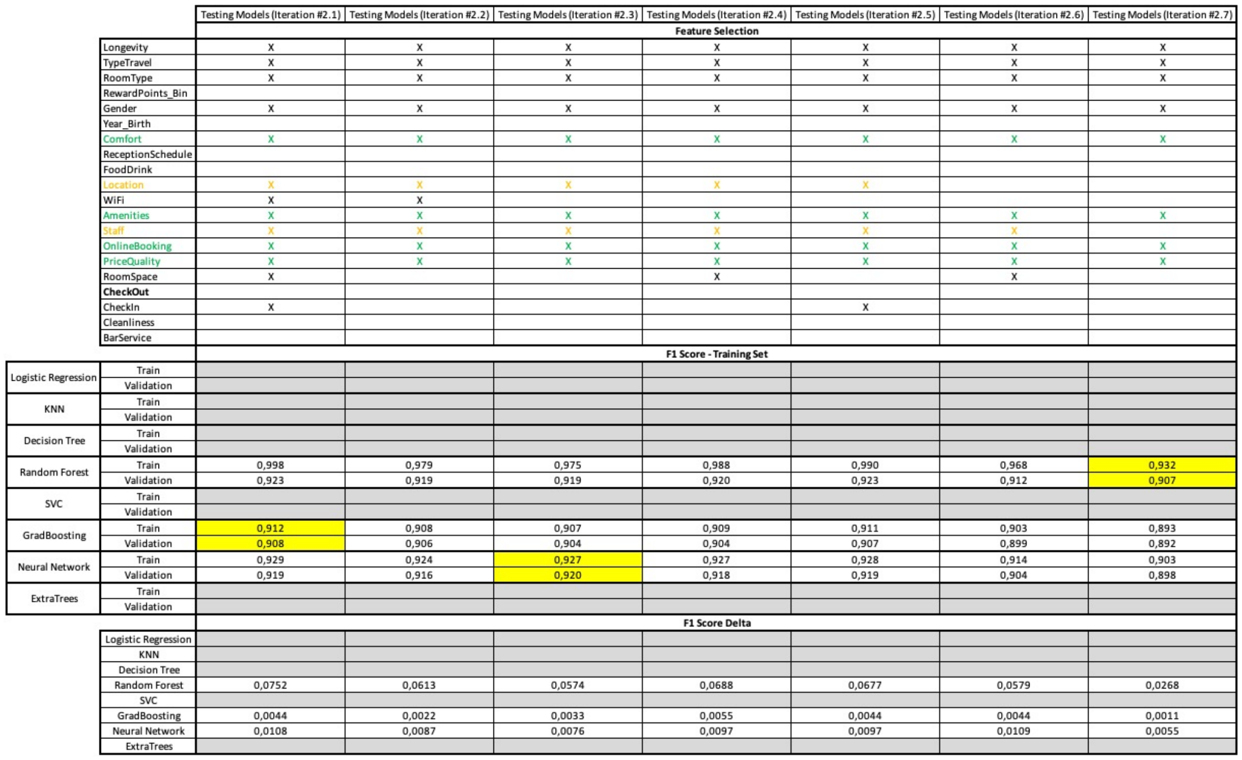
**Parameters** {'bootstrap': False, 'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': 12000, 'max_features': 0.88, 'min_samples_leaf': 1, 'min_samples_split': 10, 'n_estimators': 150, 'n_jobs': 2, 'warm_start': False}

Figure 17: Grid search results for extra trees, for distinct combinations of predictor variables.



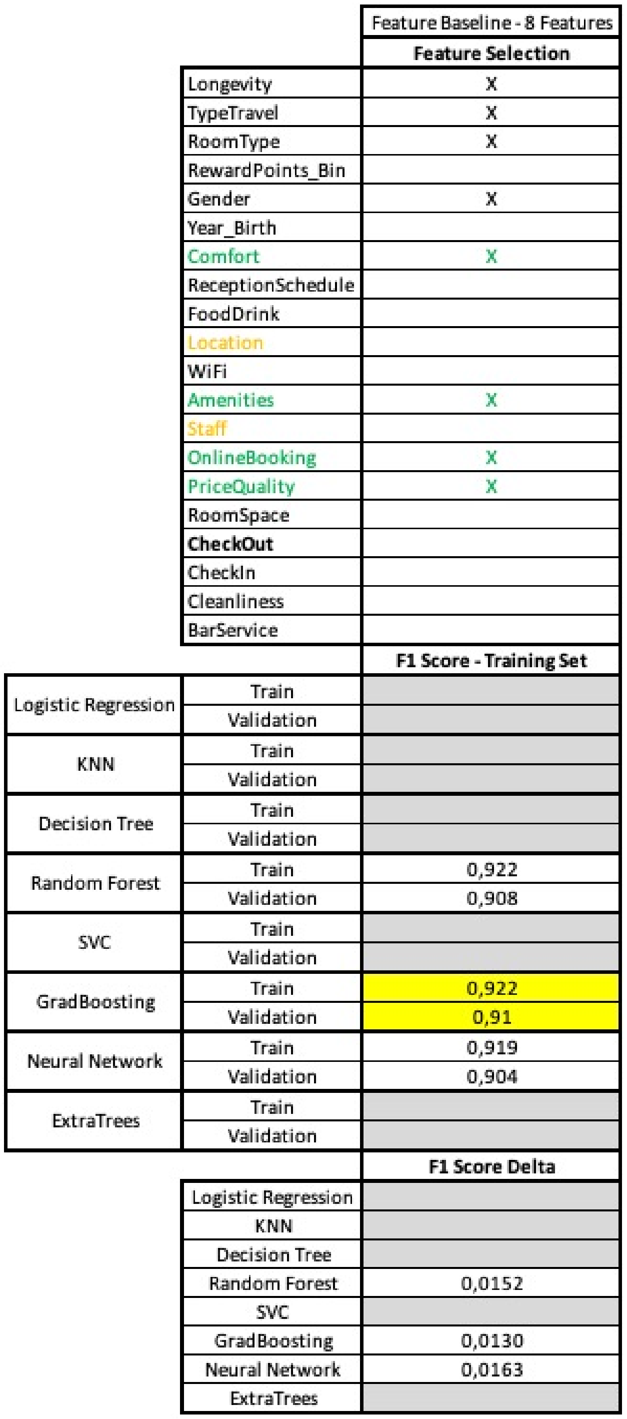
| | | Testing Models (Iteration #1.1) | Testing Models (Iteration #1.2) | Testing Models (Iteration #1.3) | Testing Models (Iteration #1.4) | Testing Models (Iteration #1.5) |
|---|---|---|---|---|---|---|
| | | **Feature Selection** | | | | |
| | Longevity | X | X | X | X | X |
| | TypeTravel | X | X | X | X | X |
| | RoomType | X | X | X | X | X |
| | RewardPoints_Bin | | | | | |
| | Gender | X | X | X | X | X |
| | Year_Birth | X | | X | | |
| | Comfort | X | X | X | X | X |
| | ReceptionSchedule | | | X | X | |
| | FoodDrink | | | | | |
| | Location | | | X | X | X |
| | WiFi | | | | | X |
| | Amenities | X | X | X | X | X |
| | Staff | | | X | X | X |
| | OnlineBooking | | | X | X | X |
| | PriceQuality | X | X | | | X |
| | RoomSpace | X | X | X | X | X |
| | **CheckOut** | | | | | |
| | CheckIn | | | X | X | X |
| | Cleanliness | | | | | |
| | BarService | | | | | |
| | | **F1 Score - Training Set** | | | | |
| Logistic Regression | Train | 0,793 | 0,795 | 0,811 | 0,809 | 0,810 |
| | Validation | 0,792 | 0,795 | 0,809 | 0,807 | 0,809 |
| KNN | Train | 0,907 | 0,909 | 0,914 | 0,924 | 0,920 |
| | Validation | 0,868 | 0,885 | 0,881 | 0,895 | 0,890 |
| Decision Tree | Train | 0,944 | 0,910 | 0,968 | 0,966 | 0,966 |
| | Validation | 0,874 | 0,887 | 0,878 | 0,885 | 0,877 |
| Random Forest | Train | 0,940 | 0,926 | 0,974 | 0,970 | 0,972 |
| | Validation | 0,907 | 0,905 | 0,925 | 0,928 | 0,924 |
| SVC | Train | 0,801 | 0,801 | 0,818 | 0,817 | 0,816 |
| | Validation | 0,799 | 0,802 | 0,816 | 0,816 | 0,814 |
| GradBoosting | Train | 0,937 | 0,920 | 0,979 | 0,972 | 0,973 |
| | Validation | 0,908 | 0,907 | 0,933 | 0,931 | 0,932 |
| Neural Network | Train | 0,928 | 0,913 | 0,967 | 0,948 | 0,951 |
| | Validation | 0,896 | 0,902 | 0,918 | 0,927 | 0,923 |
| ExtraTrees | Train | | | | | |
| | Validation | | | | | |
| | | **F1 Score Delta** | | | | |
| | Logistic Regression | 0,0013 | 0,0000 | 0,0025 | 0,0025 | 0,0012 |
| | KNN | 0,0430 | 0,0264 | 0,0361 | 0,0314 | 0,0326 |
| | Decision Tree | 0,0742 | 0,0253 | 0,0930 | 0,0839 | 0,0921 |
| | Random Forest | 0,0351 | 0,0227 | 0,0503 | 0,0433 | 0,0494 |
| | SVC | 0,0025 | 0,0000 | 0,0024 | 0,0012 | 0,0025 |
| | GradBoosting | 0,0309 | 0,0141 | 0,0470 | 0,0422 | 0,0421 |
| | Neural Network | 0,0345 | 0,0120 | 0,0507 | 0,0222 | 0,0294 |
| | ExtraTrees | | | | | |

Figure 18: Scorecard of the different machine learning models for the 1st trial. The features highlighted in green and yellow are the numerical ones initially proposed by the feature selection process (4 + 6 variables). The results converged for the variables in green (*feature baseline*). The cells in yellow represent the best scores attained for each of the algorithms; while green ones correspond to the models that led to a *Kaggle* submissions. The F1-delta is a measure of overfitting, calculated as 1 - F1 Score Validation/F1 Score Training.



| | | Testing Models (Iteration #2.1) | Testing Models (Iteration #2.2) | Testing Models (Iteration #2.3) | Testing Models (Iteration #2.4) | Testing Models (Iteration #2.5) | Testing Models (Iteration #2.6) | Testing Models (Iteration #2.7) |
|---|---|---|---|---|---|---|---|---|
| | | | | | Feature Selection | | | |
| | Longevity | X | X | X | X | X | X | X |
| | TypeTravel | X | X | X | X | X | X | X |
| | RoomType | X | X | X | X | X | X | X |
| | RewardPoints_Bin | | | | | | | |
| | Gender | X | X | X | X | X | X | X |
| | Year_Birth | | | | | | | |
| | Comfort | X | X | X | X | X | X | X |
| | ReceptionSchedule | | | | | | | |
| | FoodDrink | | | | | | | |
| | Location | X | X | X | X | X | | |
| | WiFi | X | X | | | | | |
| | Amenities | X | X | X | X | X | X | X |
| | Staff | X | X | X | X | X | X | |
| | OnlineBooking | X | X | X | X | X | X | X |
| | PriceQuality | X | X | X | X | X | X | X |
| | RoomSpace | X | | | X | | X | |
| | **CheckOut** | | | | | | | |
| | CheckIn | X | | | | X | | |
| | Cleanliness | | | | | | | |
| | BarService | | | | | | | |
| | | | | | F1 Score - Training Set | | | |
| Logistic Regression | Train | | | | | | | |
| | Validation | | | | | | | |
| KNN | Train | | | | | | | |
| | Validation | | | | | | | |
| Decision Tree | Train | | | | | | | |
| | Validation | | | | | | | |
| Random Forest | Train | 0,998 | 0,979 | 0,975 | 0,988 | 0,990 | 0,968 | 0,932 |
| | Validation | 0,923 | 0,919 | 0,919 | 0,920 | 0,923 | 0,912 | 0,907 |
| SVC | Train | | | | | | | |
| | Validation | | | | | | | |
| GradBoosting | Train | 0,912 | 0,908 | 0,907 | 0,909 | 0,911 | 0,903 | 0,893 |
| | Validation | 0,908 | 0,906 | 0,904 | 0,904 | 0,907 | 0,899 | 0,892 |
| Neural Network | Train | 0,929 | 0,924 | 0,927 | 0,927 | 0,928 | 0,914 | 0,903 |
| | Validation | 0,919 | 0,916 | 0,920 | 0,918 | 0,919 | 0,904 | 0,898 |
| ExtraTrees | Train | | | | | | | |
| | Validation | | | | | | | |
| | | | | | F1 Score Delta | | | |
| | Logistic Regression | | | | | | | |
| | KNN | | | | | | | |
| | Decision Tree | | | | | | | |
| | Random Forest | 0,0752 | 0,0613 | 0,0574 | 0,0688 | 0,0677 | 0,0579 | 0,0268 |
| | SVC | | | | | | | |
| | GradBoosting | 0,0044 | 0,0022 | 0,0033 | 0,0055 | 0,0044 | 0,0044 | 0,0011 |
| | Neural Network | 0,0108 | 0,0087 | 0,0076 | 0,0097 | 0,0097 | 0,0109 | 0,0055 |
| | ExtraTrees | | | | | | | |

Figure 19: Scorecard of the different machine learning models for the 2nd trial. The features highlighted in green and yellow are the numerical ones initially proposed by the feature selection process (4 + 6 variables). The results converged for the variables in green (*feature baseline*). The cells in yellow represent the best scores attained for each of the algorithms. The F1-delta is a measure of overfitting, calculated as 1 - F1 Score Validation/F1 Score Training.

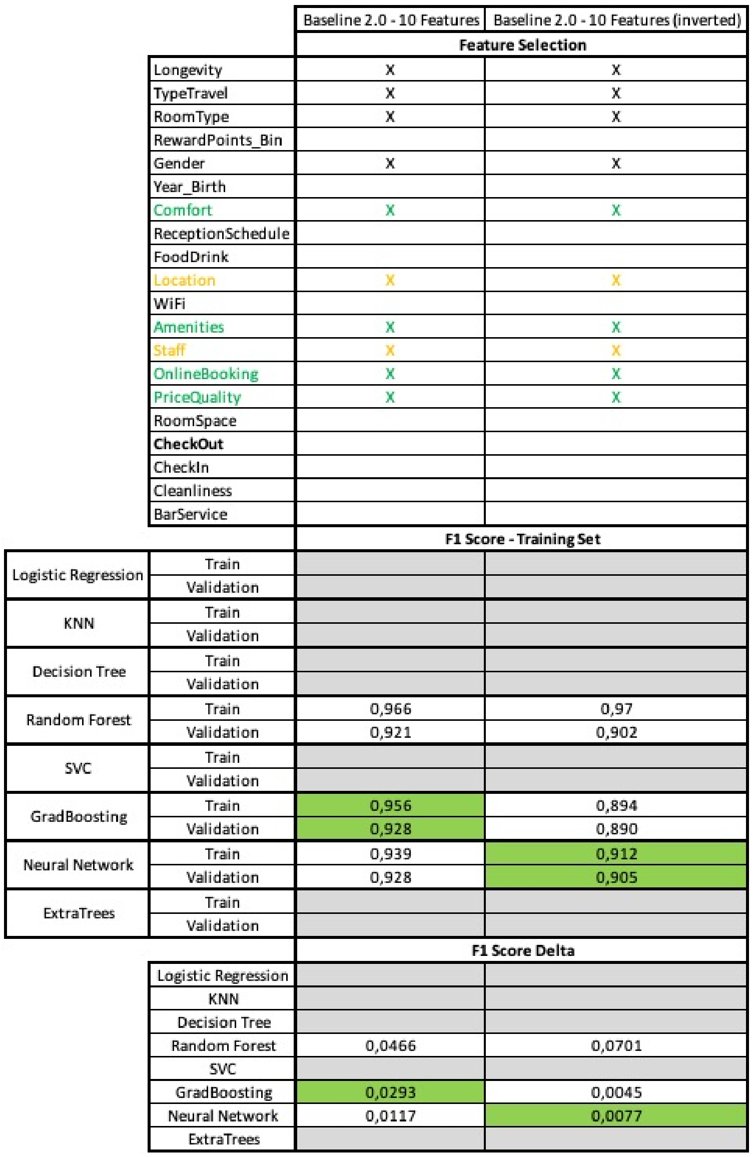


| | Feature Baseline - 8 Features |
|---|---|
| | **Feature Selection** |
| Longevity | X |
| TypeTravel | X |
| RoomType | X |
| RewardPoints_Bin | |
| Gender | X |
| Year_Birth | |
| Comfort | X |
| ReceptionSchedule | |
| FoodDrink | |
| Location | |
| WiFi | |
| Amenities | X |
| Staff | |
| OnlineBooking | X |
| PriceQuality | X |
| RoomSpace | |
| **CheckOut** | |
| CheckIn | |
| Cleanliness | |
| BarService | |

| | | **F1 Score - Training Set** |
|---|---|---|
| Logistic Regression | Train | |
| | Validation | |
| KNN | Train | |
| | Validation | |
| Decision Tree | Train | |
| | Validation | |
| Random Forest | Train | 0,922 |
| | Validation | 0,908 |
| SVC | Train | |
| | Validation | |
| GradBoosting | Train | 0,922 |
| | Validation | 0,91 |
| Neural Network | Train | 0,919 |
| | Validation | 0,904 |
| ExtraTrees | Train | |
| | Validation | |

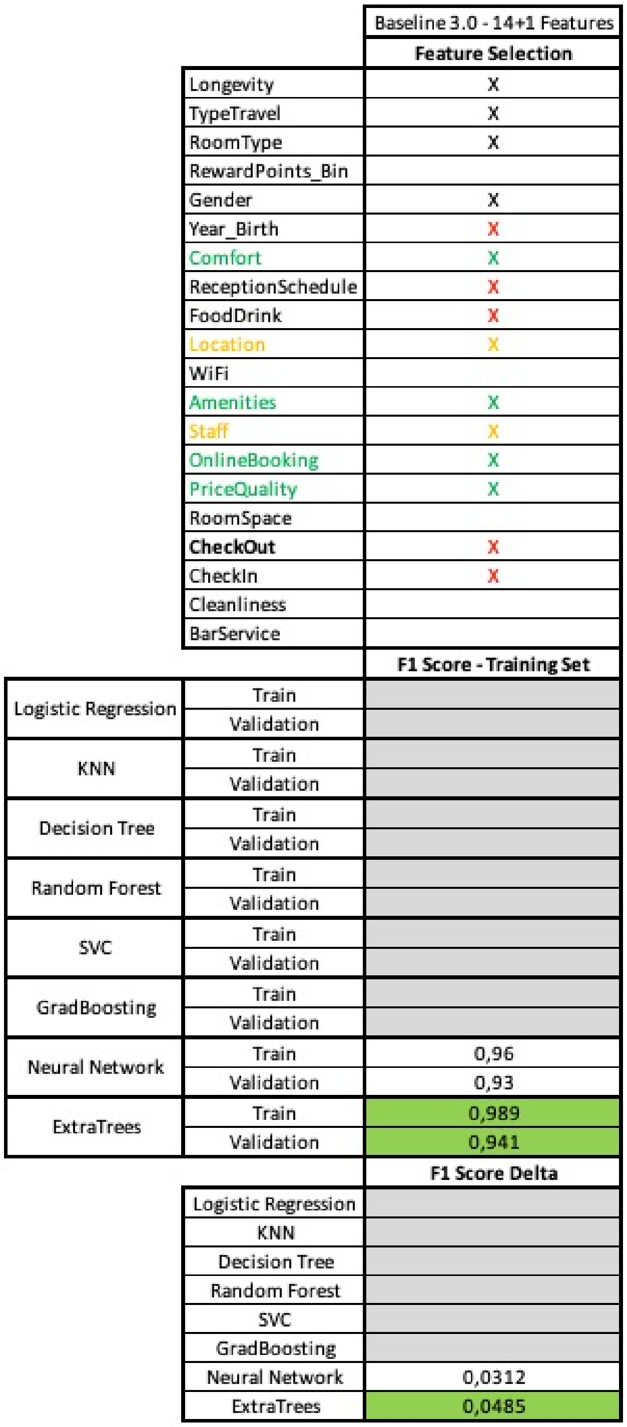
| | **F1 Score Delta** |
|---|---|
| Logistic Regression | |
| KNN | |
| Decision Tree | |
| Random Forest | 0,0152 |
| SVC | |
| GradBoosting | 0,0130 |
| Neural Network | 0,0163 |
| ExtraTrees | |

Figure 20: Scorecard of the different machine learning models for the 3rd trial (*Feature Baseline*). The features highlighted in green and yellow are the numerical ones initially proposed by the feature selection process (4 + 6 variables). The results converged for the variables in green (*feature baseline*). The cells in yellow represent the best scores attained for each of the algorithms. The F1-delta is a measure of overfitting, calculated as 1 - F1 Score Validation/F1 Score Training.



| | | Baseline 2.0 - 10 Features | Baseline 2.0 - 10 Features (inverted) |
|---|---|---|---|
| | | **Feature Selection** | |
| | Longevity | X | X |
| | TypeTravel | X | X |
| | RoomType | X | X |
| | RewardPoints_Bin | | |
| | Gender | X | X |
| | Year_Birth | | |
| | Comfort | X | X |
| | ReceptionSchedule | | |
| | FoodDrink | | |
| | Location | X | X |
| | WiFi | | |
| | Amenities | X | X |
| | Staff | X | X |
| | OnlineBooking | X | X |
| | PriceQuality | X | X |
| | RoomSpace | | |
| | **CheckOut** | | |
| | CheckIn | | |
| | Cleanliness | | |
| | BarService | | |
| | | **F1 Score - Training Set** | |
| Logistic Regression | Train | | |
| | Validation | | |
| KNN | Train | | |
| | Validation | | |
| Decision Tree | Train | | |
| | Validation | | |
| Random Forest | Train | 0,966 | 0,97 |
| | Validation | 0,921 | 0,902 |
| SVC | Train | | |
| | Validation | | |
| GradBoosting | Train | 0,956 | 0,894 |
| | Validation | 0,928 | 0,890 |
| Neural Network | Train | 0,939 | 0,912 |
| | Validation | 0,928 | 0,905 |
| ExtraTrees | Train | | |
| | Validation | | |
| | | **F1 Score Delta** | |
| | Logistic Regression | | |
| | KNN | | |
| | Decision Tree | | |
| | Random Forest | 0,0466 | 0,0701 |
| | SVC | | |
| | GradBoosting | 0,0293 | 0,0045 |
| | Neural Network | 0,0117 | 0,0077 |
| | ExtraTrees | | |

Figure 21: Scorecard of the different machine learning models for the 4th trial (*Feature Baseline 2.0*). The features highlighted in green and yellow are the numerical ones initially proposed by the feature selection process (4 + 6 variables). The results converged for the variables in green (*feature baseline*). The cells in yellow represent the best scores attained for each of the algorithms; while green ones correspond to the models that led to a *Kaggle* submissions. The F1-delta is a measure of overfitting, calculated as 1 - F1 Score Validation/F1 Score Training.



| | Baseline 3.0 - 14+1 Features |
|---|---|
| | **Feature Selection** |
| Longevity | X |
| TypeTravel | X |
| RoomType | X |
| RewardPoints_Bin | |
| Gender | X |
| Year_Birth | X |
| Comfort | X |
| ReceptionSchedule | X |
| FoodDrink | X |
| Location | X |
| WiFi | |
| Amenities | X |
| Staff | X |
| OnlineBooking | X |
| PriceQuality | X |
| RoomSpace | |
| **CheckOut** | X |
| CheckIn | X |
| Cleanliness | |
| BarService | |

| | | **F1 Score - Training Set** |
|---|---|---|
| Logistic Regression | Train | |
| | Validation | |
| KNN | Train | |
| | Validation | |
| Decision Tree | Train | |
| | Validation | |
| Random Forest | Train | |
| | Validation | |
| SVC | Train | |
| | Validation | |
| GradBoosting | Train | |
| | Validation | |
| Neural Network | Train | 0,96 |
| | Validation | 0,93 |
| ExtraTrees | Train | 0,989 |
| | Validation | 0,941 |

| | **F1 Score Delta** |
|---|---|
| Logistic Regression | |
| KNN | |
| Decision Tree | |
| Random Forest | |
| SVC | |
| GradBoosting | |
| Neural Network | 0,0312 |
| ExtraTrees | 0,0485 |

Figure 22: Scorecard of the different machine learning models for the 5th trial. The features highlighted in green and yellow are the numerical ones initially proposed by the feature selection process (4 + 6 variables). The results converged for the variables in green (*feature baseline*). The red ones are the ones considered in this last trial, the majority of them proposed by the feature selection process performed at this stage, with the aim of selecting the 10 best features (4 + 10 variables). The green cells correspond to the models that led to a *Kaggle* submissions. The F1-delta is a measure of overfitting, calculated as 1 - F1 Score Validation/F1 Score Training.

Table 1: Summarization of the models' performances for the *Feature Baseline*, *Feature Baseline 2.0* and *Feature Baseline 3.0* stages for the different considered algorithms considered at this stage. The F1 Scores on both training and validation sets are presented along with the value of the overfitting (F1-delta), calculated as 1 - F1 Score Validation/F1 Score Training.

| | Feature Baseline | | | Feature Baseline 2.0 | | | Feature Baseline 3.0 | | |
|---|---|---|---|---|---|---|---|---|---|
| | F1 Score Train | F1 Score Validation | Overfitting | F1 Score Train | F1 Score Validation | Overfitting | F1 Score Train | F1 Score Validation | Overfitting |
| Random Forest | 0.922 | 0.908 | 0.0152 | 0.966 | 0.921 | 0.0466 | NA | NA | NA |
| Gradient Boosting | 0.922 | 0.910 | 0.013 | 0.956 | 0.928 | 0.0293 | NA | NA | NA |
| Neural Networks (MLP) | 0.919 | 0.904 | 0.0163 | 0.939 | 0.928 | 0.0117 | 0.960 | 0.930 | 0.0312 |
| Extra Trees | NA | NA | NA | NA | NA | NA | 0.983 | 0.941 | 0.0485 |

## 7.2 *Logistic Regression*

The well-known linear regression allows to approximate the relationship between a continuous target variable and a set of predictors. However, many times the target variable is categorical; in this cases, for which the linear regression is not suited, we use an analogous approach - the logistic regression. Therefore, logistic regression is a linear classification technique used in binary classification problems.

The method proposed a model for the *a posteriori* (outcome) probabilities of the classes, this is, for the probabilities of observing each class given the values observed for the set of predictor features. These are estimated using the formula of the logistic function and are expressed in the equations below, in which *x* accounts for the values in the feature vector and $\beta$ stands for the correspondent regression coefficients. It is guaranteed the sum of both probabilities equals 1.

$$P(y=1|\mathrm{x}) = \frac{1}{1+e^{-\mathrm{x}^T\beta}} \tag{2}$$

$$P(y=0|\mathrm{x}) = \frac{e^{-\mathrm{x}^T\beta}}{1+e^{-\mathrm{x}^T\beta}} \tag{3}$$

The coefficients $\beta$ are estimated by the maximum likelihood method, in which the objective is to maximize the conditional log-likelihood function (Equation 4). This is a function of the parameters $\beta$ that expresses the probability of the observed data. The goal is to find the coefficients $\beta$ for which the likelihood of observing the observed data is maximized. Numerical optimization algorithms can be used to solve this optimization problem, such as the gradient ascent method or the Newton method, since the log-likelihood function cannot be analytically optimized.

$$l(\beta) = \log P(y^1, \cdots, y^n | x^1, \cdots, x^n; \beta) \tag{4}$$

## 7.3 *Lasso Regression*

Again recalling the linear regression model, it can be estimated by minimizing the least squares criterion, which minimizes the sum of squared errors. Yet, in the scope of some problems, we may want to be more selective in the selection of features or to penalize larger errors. In such cases, we can use Lasso regression, an alternative to the linear regression, in which the aim is to minimize the sum of squared errors but including a constraint on the coefficients in order to penalize the errors (see Equation 5 in which $\beta$ represents the regression coefficients). This constraint is called a regularization term and is dependent on a parameter $\lambda$ (regularization parameter).

$$\beta_{Lasso} = \arg\min_{\beta} \; \|y - X\beta\|_2^2 + \lambda \|\beta\|_1 \tag{5}$$

This optimization problem cannot be solved using a linear system of equations, so convex optimization methods are required to numerically solve this problem.

From the equation above, it is possible to observe that the model is penalized for the sum of absolute values of the coefficients. This way, by adopting this strategy when determining the coefficients $\beta$, if a feature does not contribute to predict the outcome, Lasso regression will probably set the corresponding coefficient equal to zero. In fact, the coefficients will tend to zero and, overall, will be of smaller magnitudes.

This aspect about Lasso regression allows the creation of simpler, sparser models that use less number of features and parameters. This is why Lasso regressions ends up being usually used in machine learning for the selection of the the most relevant subset of predictor features in the feature selection process.

## 7.4 *k-Nearest Neighbors (KNN) Imputer*

The *k*-nearest neighbors (KNN) imputer is a data transformation method that relies on the KNN algorithm, a supervised machine learning algorithm that assumes similar things exist in close proximity in order to replace the missing values in the data set. In a nutshell, it finds the *k* closest neighbors to the observation with missing data and imputes it based on the the non-missing values in the neighbors.

The process is initialized by the selection of the *k* value, in which it is good to understand that using a low value for *k* will increase the influence of noise, reaching less "accurate" results; whereas choosing a high *k* will tend to blur local effects. One way to define it is by assigning it the square-root of the total number of sample.

With *k* set, for every observation with a missing value in the data set, the algorithm will determine, based on a distance criterion between observations (the most commonly used is the euclidean distance), its *k*-nearest neighbors. Then, for the variable with missing value of that observation:

- If we are dealing with numerical data: the algorithm calculate the mean value of the *k*-sample, in order to fill the missing value by a uniform locally computed mean. However, alternatively, one can choose to assign different weights to each neighbor according to its distance to the observation with the missing value, thus computing a locally weighted mean that will replace the missing value.
- If we are dealing with categorical data: the algorithm will get the mode value of the *k*-sample to fill the missing data.

The contribution of each neighbor in determining the missing value can be uniform or distinctly weighted (according, for instance, to the distance function established). It might also be a good practice to apply KNN Imputer after normalizing the data, if needed, in order not to overemphasize variables that range between larger values.

## 7.5 *KNN algorithm*

Following the description of the KNN imputer, the KNN algorithm is a non-parametric machine learning technique, most often used for classification tasks, in which the classification for a new unclassified instance may be found by simply comparing it to the most similar records in the training set. Again, as described in the previous section, *k* refers to the number of neighbors surrounding the data point that are considered to make the prediction, and the calculations behind the KNN algorithm reside in how distances from a data point to its neighbors are computed. Note that the KNN algorithm lies under the assumption that the classification of an observation will be most similar to the classification of other observations that are nearby, also known as the inductive bias.

The considerations that must be taken when applying the KNN algorithm are the same pointed out when discussing the use of the KNN Imputer in the previous section - suiting *k* given the characteristics of the problem, determining the distance function, and assign the weights to each neighbor. This latter aspect allows the KNN algorithm to be more robust, since assigning greater weights to closer neighbors can smooth out the impact of isolated noisy observations.

A last remark on KNN Imputer and KNN algorithm is that the distance between data points is, by default, calculated based on all attributes of the data point/data set. Despite this can be overcome when pre-processing the data (for instance, by selecting only some relevant features to perform the prediction), it can be pointed out as a difference compared to other non-parametric methods such as decision trees, in which only a subset of the attributes is selected when forming the hypothesis.

## 7.6 ***Random Forest***

Model ensembles combine predictions from machine learning models into a single score, thus improving model accuracy and robustness and reducing overfitting. Random Forests were introduced in the 2000s as powerful ensemble methods.

The idea is to build multiple decision trees from re-sampled data using bootstrap sampling and combine their predictions through voting or averaging. This way, one decision tree is built from each bootstrap sample. However, unlike in the traditional bagging algorithm which follows the same rationale, at each split in the tree only a random subset of features is considered rather than all input variables. The number of subset features to use is usually given as a parameter of the algorithm, as well as the number of samples when performing bootstrapping, the number of trees in the forest, the split criterion, the maximum depth of the trees, the minimum number of samples required to split a node, among others, similarly to those specified when training a Decision Tree.

On the one hand, the bootstrap sampling is used to introduce diversity in the models, which is of major importance in the case of decision tress since mall changes to the training set can result in significantly different tree structures. On the other hand, the averaging smoothes the predictions to behave in a more stable way on new data. Finally, the trees are also trained using different features. All of these contribute to obtaining uncorrelated predictions, which is the major advantage of ensemble methods.

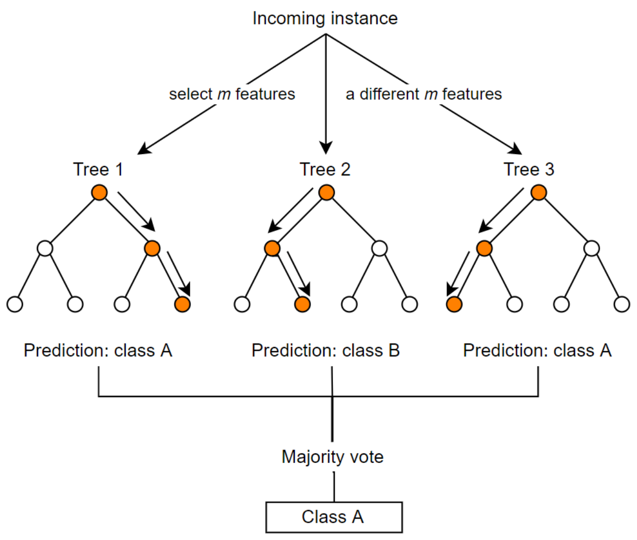


Figure 23: Schematics of the Random Forest ensemble model.

## 7.7 *Gradient boosting*

Boosting is another type of model ensemble that varies from bagging or Random Forests since it is adaptive. The same classification model is applied successively to the training sample and, at each iteration, the algorithm applies greater weight to the records that have been misclassified. It relies on the weak learner hypothesis and on the idea of filtering observations - leaves out observations that the weak learner can handle and focuses on developing new weak learns to handle the remaining difficult observations. This process readjusts the distribution of data and a new base classifier is determined and added as the weights changes, until a certain stopping criterion is met or a given number of iterations achieved. The final boosted classifier is the weighted sum of all base classifiers.

Several boosting algorithms have been developed, such as *AdaBoost*. Such related algorithms were generalized in a statistical framework called gradient boosting. Gradient boosting thus involves three elements:

1. Defining a loss function, to calculate the residuals or errors associated to the misclassifications.

2. Defining the weak learner to make the predictions, which are decision trees constructed in a greedy approach and some eventual constraints. Specifically, regression trees are used, which allows subsequent models outputs to be added to ”correct” the residuals in the predictions.

3. Defining the additive model: trees are added one at a time, recalling that existing trees in the model are not changed. A gradient descent method is used to minimize the loss when adding trees - like the subsequent trees use the errors as a target variable.The output for the new tree is then added to the output of the existing sequence of trees in an effort to improve the final output of the model. Typically, hundreds of trees are built and the final predictions are and additive combination of the former predictions.

As mentioned, gradient boosting is a greedy algorithm. Thus, it will benefit from regularization methods and constraints that may reduce overfitting. Besides constraints when building the decision trees, other improvements such as shrinkage, random sampling and penalized learning can be considered. These constitute parameters to the *scikit-learn* gradient boosting classifier method.

Gradient boosting is indeed a very powerful tool, able to find any non-linear relationship between predictors and target variables. Moreover, it is able to deal with missing values, outliers and categorical features that present high cardinality, thus not requiring an extensive data pre-processing.

## 7.8 *Extremely Randomized (Extra) Trees*

Extra Trees is another ensemble method, quite similar to Random Forest, yet that introduces more variation into the ensemble and therefore changes the way how trees are computed. The main differences between Extra Trees and Random Forest are:

1. Whereas Random Forest uses bootstrap replicas to construct the samples used to build, successively, the decision trees, Extra Trees uses all data available in the training set. Note that in *scikit-learn* it is also possible to use bootstrap replicas (optional parameter), however the entire sample is used by default.

2. When splitting nodes, while Random Forest chooses the optimum split, Extra Trees selects the best split randomly. Nevertheless, once the split points are chosen, both algorithms chose the best one between the considered selected features, which still account for optimization.

These differences promote a reduction in overfitting and, in terms of computational cost, allow the Extra Trees algorithm to be faster.

## 7.9 *Support Vector Machine (SVM)*

SVM is supervised learning algorithm whose basic underlying idea is to define an hyperplane - decision boundary - in a N-dimensional space, in which N is the number of features, that separates the data points. It can be used for both classification and regression problems, as well as on outlier detection tasks. For instance, in the context of classification problems, the aim consists in determining the hyperplane that distinctly classifies the data points.

The equation of a general hyperplane is defined by the equation below, in which *w* is a normal vector orthogonal to the hyperplane and *b* is the offset (distance to the origin if $\|w\| = 1$):

$$w \cdot x + b = 0 \tag{6}$$

If several hyperplanes separate the data points in the feature/input space, then the hyperplane to be chosen must be as far as possible from the closest data points. If we consider $d_+$ as the distance from the hyperplane to the closest point of class $+1$ and $d_-$ as the distance from the hyperplane to the closest point of class $-1$, the margin is defined as:

$$d = d_+ + d_- \tag{7}$$

This means that the aim is to find the hyperplane with the largest margin. Note that it is assumed that $d_+ = d_-$. In fact, the vectors $x_i$ of class $+1(-1)$ that are separated from the hyperplane by a distance $d_+(d_-)$ are the so-called support vectors, which will be the important ones to define the separating hyperplane.

In order to understand which data points are support vectors, a quadratic optimization problem must be solved as suggested in Equation 8 in the case of linearly separable data, in which $\alpha = [\alpha_1 \cdots \alpha_n]^T$and

$H_{ij} = y^{(i)}(x^{(i)} \cdot x^{(j)})y^{(j)}$. This is called the hard margin problem. From the multipliers $\alpha_i$ , we can obtain the support vectors, normal vector, offset and the classification of the data points and determine the equation of the hyperplane.

$$\underset{\alpha}{max} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\alpha^T H\alpha, \;\; s.t. \;\; \alpha_i \geq 0 \,\forall\, i, \; \sum_{i=1}^{n} \alpha_i y^{(i)} = 0 \tag{8}$$

If we are dealing with non-separable data, we may allow data points on the wrong side of the hyperplane but they suffer a penalty. Thus, the idea is to assign a slack variable to each data point so that it equals zero if no margin violation occurs, and it is greater than zero if the data point is on the wrong side of the hyperplane. This is called the soft margin problem and introduces a soft margin penalty ($C$ constraint in Equation 9) in the aforementioned optimization problem:

$$\underset{\alpha}{max} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\alpha^T H\alpha, \;\; s.t. \;\; 0 \leq \alpha_i \leq C \,\forall\, i, \; \sum_{i=1}^{n} \alpha_i y^{(i)} = 0 \tag{9}$$

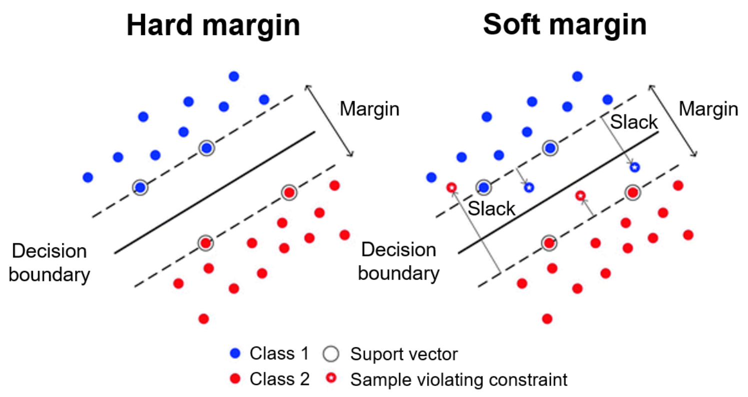


Figure 24: SVM algorithm: hard margin and soft margin.

The first basic SVMs were proposed in the 1960s for binary classification problems with linear decision boundaries. However, most problems are indeed non-linearly separable and require non-linear decision boundaries. In order to solve them, the data ($x$) in the feature space must be mapped into a high-dimensional space ($\phi(x)$) in which it can be separated by an hyperplane. From the Equations 8 and 9, we notice that the SVM algorithm does not require the input vectors but only the inner product between them, which is the motivation for solving the problem in the high-dimensional space. These inner products can be computed using a kernel function that does not require the computation of high dimension features - the kernel trick:

$$k(x^i, x^j) = \phi(x^i) \cdot \phi(x^j) \tag{10}$$

The most common choices are linear, radial basis function (RBF) and polynomial kernels. This way, the non-linear SVM can be trained and tested using lower-dimensional data by replacing the inner products by the kernel.

The kernel function is the real strength of SVM models, allowing them to be generalized and used in different problems with less overfitting. Given their simplicity comparing to other machine learning algorithms as neural networks, they provide very good results. Yet, it can be a challenge to define the right kernel function and are not suitable when dealing with very large, noisy datasets.

## 7.10 *Naïve Bayes classifier*

In machine learning, a classifier works by discriminating different objects based on certain features. Usually, the concept of cost or loss function is used and the best classifier is considered to be the one that minimizes it. This is equivalent to, for given a data point, choosing the class $C_i$ with the greatest *a posteriori* probability $P(C_i|x)$, this is, the most probable class given the observations. This is known as the Bayes classifier and the *a posteriori* probability can be obtained using the Bayes theorem:

$$P(C_i|x) = \frac{P(x|C_i)P(C_i)}{P(x)} \tag{11}$$

$P(x|C_i$ is the conditional probability of observing $x$ given the instance is associated to class $C_i$ (also called the likelihood function), $P(C_i)$ is the *a priori* distribution of the classes, and $P(x)$ is a normalization term that does not influence the decision of the classifier.

However, when the feature vector contains many features, the estimation of the conditional distribution can be a difficult problem. In such cases, using the Naïve Bayes approximation is preferred. The Naïve Bayes classifier is a probabilistic machine learning algorithm is based on the Bayes theorem, but simplifies the problem by assuming that features are conditionally independent:

$$P(x_1 \cdots x_p|C_i) = \prod_{k=1}^{p} P(x_k|C_i) \tag{12}$$

The conditional independence assumption should not be made randomly, which emphasizes the task of performing exploratory data analysis in order to spot and deal with correlations between predictor variables. Otherwise, the Naïve Bayes classifier performs sub-optimally. Yet, considering its simplicity, it often leads to surprisingly good results.

There are some variations of the Naïve Bayes classifier, namely the Gaussian, Bernoulli and Multinomial Naïve Bayes. They vary on characteristics and data types of the predictor variables and their use depends on the context of the problem being solved. For instance, the Gaussian Naïve Bayes classifier is frequently used and assumes the likelihood of the features (continuous) follows a gaussian or normal distribution.

Note that if $P(x|C_i)$ is zero, this is, if there is a completely new value assigned to at least one of the features after the training phase of the classifier, the *a posteriori* probability $P(C_i|x)$ is also null. This is called the zero-probability phenomenon. In order to account for this problematic, a smoothing term is introduced when training the method to adjust the calculation of the probabilities and, therefore, avoid them to be set to be exactly zero.

# 8 References

Larose, D. T. (2015). *Data mining and predictive analytics.* John Wiley & Sons.

Mitchell, T. M., & Mitchell, T. M. (1997). *Machine learning* (Vol. 1, No. 9). New York: McGraw-hill.

Abbott, D. (2014). *Applied predictive analytics: Principles and techniques for the professional data analyst.* John Wiley & Sons.

*https://scikit-learn.org/stable/modules/classes.html*

*https://deepai.org/machine-learning-glossary-and-terms/random-forest*

*https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/*

*https://towardsdatascience.com/all-you-need-to-know-about-gradient-boosting-algorithm-part-1-regression-2520a34a502*
*https://quantdare.com/what-is-the-difference-between-extra-trees-and-random-forest/*
*https://mljar.com/machine-learning/extra-trees-vs-random-forest/*
*https://towardsdatascience.com/an-intuitive-explanation-of-random-forest-and-extra-trees-classifiers-8507ac21d54b*