Eindopdracht 2

Mohammed Al Hor

2023-01-22

Opdracht 2: Hotel Bookings

1. Welke data preparatie stappen u neemt (en waarom).

First, we load the data and check if there are missing values.

There are no missing values in the data, let's move on.

First off, we data contains columns with categorical variables; 'is_canceled', 'country' and 'market_segment','is_repeated_guest'. The remaining numeric variables have are on different scales and thus must be scaled before analysis. The following section of the data preparation deals with this.

The aforementioned categorical variables are converted to dummy variables and the remaining numeric variables are scaled.

In this last step of data preparation the data set is split into a training and test set. The training set will be used to train the model, whereas the test set will be used to evaluate the model.

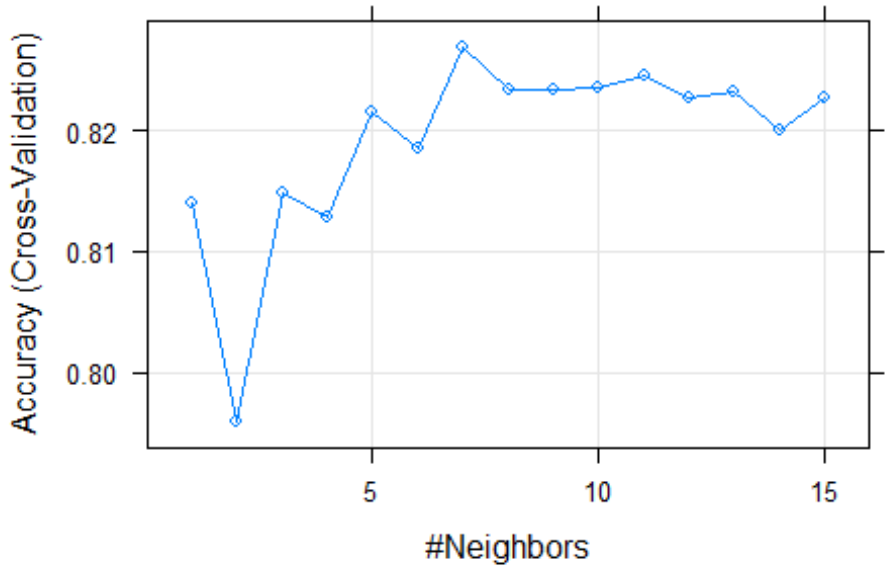
We use random sampling and we end up with a training set contains that contains 60% of the observations (11879) and a test set that contains 40% (7918) of the observations.

Welke methode(n) u gebruikt en, indien van toepassing, hoe u tot uw uiteindelijke voorspelmodel(len) komt (dus: keuze van parameters, etc.).

K-nearest neighbours:

Now that we did the data preparation, the fun can begin. The first method we consider is of course the 'lazy' learner method. KNN (K-nearest neighbour) is a simple algorithm that stores all available cases and classifies new data using similarity measures. KNN is quick and thus we start off with this method. We perform this method on the training data, try different values of K (amount of nearest neighbours we should consider) using cross-validation and make predictions on the test data. Now that we have our KNN models we can start looking the different values for K and pick the best one.

The plot below shows the accuracy of the model across different values of K. This 'optimal' value is set and used to train our definitive model. Optimal K is 7. We will get into the evaluation of this model in the next question.



XGBoost:

The second method we consider is the popular and powerful 'XGBoost classifier'. XGBoost is an extension to gradients boosting decision trees. It's fast, because of it use of parallel and distributed computing, and accurate. Let's delve into this algorithm and apply it to our data.

In the following section we building the model in order to predict whether a customer would cancel their booking or not. We use the cross validation method to determine the optimal value for the hyperparameters. We pick the best model and we use this model to make predictions on the test data.

```
## eXtreme Gradient Boosting
##
## 11879 samples
##    35 predictor
##     2 classes: 'no', 'yes'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 9503, 9504, 9502, 9504, 9503
## Resampling results across tuning parameters:
##
##   eta  max_depth  nrounds  Accuracy   Kappa
```

```
##   0.1  1          100      0.8203548  0.4992738
##   0.1  1          500      0.8461148  0.5980299
##   0.1  2          100      0.8442626  0.5899367
##   0.1  2          500      0.8637938  0.6500335
##   0.3  1          100      0.8444312  0.5900815
##   0.3  1          500      0.8520077  0.6155322
##   0.3  2          100      0.8596690  0.6393327
##   0.3  2          500      0.8628676  0.6497175
##
## Tuning parameter 'gamma' was held constant at a value of 0
## Tuning
##
## Tuning parameter 'min_child_weight' was held constant at a value of 1
##
## Tuning parameter 'subsample' was held constant at a value of 1
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were nrounds = 500, max_depth = 2, eta
##  = 0.1, gamma = 0, colsample_bytree = 0.6, min_child_weight = 1 and subsam
ple
##  = 1.
```

Now that we have these optimal hyperparameters we can use these in our final model. See the code for details.

Welke maten u gebruikt om de resultaten te beoordelen (en waarom).

Let us consider both models and look at some metrics that give us insight into the performance of these models. KNN: In the following code we make a prediction using the KNN model on the training set. Furthermore, we examine the following performance metrics: accuracy, precision, sensitivity and specificity.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  5095  731
##        yes  608 1484
##
##                Accuracy : 0.8309
##                  95% CI : (0.8225, 0.8391)
##     No Information Rate : 0.7203
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5731
##
##  Mcnemar's Test P-Value : 0.000856
##
##             Sensitivity : 0.8934
##             Specificity : 0.6700
##          Pos Pred Value : 0.8745
```

```
##          Neg Pred Value : 0.7094
##              Prevalence : 0.7203
##          Detection Rate : 0.6435
##    Detection Prevalence : 0.7358
##       Balanced Accuracy : 0.7817
##
##        'Positive' Class : no
##
```

The confusion matrix presented above provides some interesting insight into the performance of the model. First off, we see an overall accuracy of 83%. In 83% of the cases the KNN model could accurately predict whether a booking would be cancelled or not. The sensitivity of the model is 89%. The sensitivity is also called the true positive rate. This means that KNN correctly identified whether a customer would not cancel the booking in 89% percent of the cases. The specificity of this model is 67% (True negative rate). In 67% of cases KNN could identify the cases where a customer would cancel his/her booking. Lastly, we look at the precision of the model. This entails the when the model predicted the 'positive' class, how accurate was it? The precision is 87%. The model was correct 87% of the time.

XGBoost:

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   no  yes
##        no  5291  671
##        yes  412 1544
##
##                Accuracy : 0.8632
##                  95% CI : (0.8555, 0.8707)
##     No Information Rate : 0.7203
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.648
##
##  Mcnemar's Test P-Value : 4.512e-15
##
##             Sensitivity : 0.9278
##             Specificity : 0.6971
##          Pos Pred Value : 0.8875
##          Neg Pred Value : 0.7894
##              Prevalence : 0.7203
##          Detection Rate : 0.6682
##    Detection Prevalence : 0.7530
##       Balanced Accuracy : 0.8124
```

```
##
##        'Positive' Class : no
##
```

4. Uw conclusies

The model is used to make predictions on the test data and the results are shown above in a confusion matrix. With an accuracy of 86% the XGBoost model outperforms KNN. In other words, the XGBoost model accurately predicts whether a booking is cancelled in 86% of cases, compared to 83% of KNN. Furthermore, the sensitivity of this model is close to 93%. It can accurately predict the 'positive class' (no) in 93% of cases and again outperforms the previous model. The specificity (true negative rate) and precision of the model are 70% and 89% respectively. The XGBoost model outperforms KNN across the board. See the code added in the R-markdown file for a more thorough look into the methodology.