\

Hotel Cancellation Prediction

Karim AbouDaoud
900212779
Department of Mathematics and Actuarial Science
The American University in Cairo
Cairo, Egypt
karimaboudaoud@aucegypt.edu

Youssef Nakhla
900201430
Department of Mathematics and Actuarial Science
The American University in Cairo
Cairo, Egypt
youssef_n9212@aucegypt.edu

Model Experimentation

To further examine, possible models that could be used to solve our problem, we decided to dive more into detail and try models that we did not cover in the last phase. We experimented using different ensemble combinations including the ensemble of SVM & Logistic Regression and Logistic Regression & Random Forest. Both ensembles tend to have a very large time complexity to run and both yielded a less performing model than our original models. Hence we decided to remain with the default models and depend on fine-tuning to optimize our model.

Model Selection

Based on the results of the pilot study and the previous phase, the top three most performing models were KNN, Random Forest and ANN Recitfied Linear Unit. Hence, we must dissect the pros and cons of the three models and their respective adaptability to our dataset. Our dataset consists of strictly categorical binary features with a large number of observations, hence it is vital to take that into consideration while choosing the most appropriate model.

Decision trees, particularly Random Forests, provide interpretability, capture nonlinear relationships, and robustness to outliers. However, they are prone to overfitting and display significant variance, particularly on smaller datasets, which may have an impact on their performance. Artificial Neural Networks using Rectified Linear Unit activation functions can learn complex patterns and achieve great accuracy, but they need large processing resources and lack interpretability. K-Nearest Neighbors, on the other hand, are simple to design and can efficiently capture local trends, but they use a lot of memory and have computational complexity, especially with large datasets. Given the dataset's characteristics of around 120k observations and 45variables, Random Forest appears as the optimal choice due to its mix of interpretability, computational economy and performance. Its ability to handle complex relationships, robustness to outliers, and scalability make it well-suited for predicting hotel cancellations while mitigating the limitations posed by the dataset size and feature complexity.

Model Fine-Tuning and Data Preprocessing

The first step of fine-tuning a model is to define the model which has been done in the earlier stage. Since our problem is to predict hotel cancellations, our main performance metrics would be the model accuracy, precision, recall and F1-score. It is important to note the performance of our Random Forest model before any fine-tuning as seen below:

```
Accuracy Score of Random Forest is : 0.8743539101567425
F1 score: 0.825
Classification Report :
              precision    recall  f1-score   support

           0       0.89      0.92      0.90     15018
           1       0.85      0.80      0.82      8779

    accuracy                           0.87     23797
   macro avg       0.87      0.86      0.86     23797
weighted avg       0.87      0.87      0.87     23797
```

Our first step to fine-tune our model was to perform feature selection to examine the most ‘influential’ features that affect our prediction. We then defined the hyperparameters of our model which are number of estimators, maximum depth, minimum samples split, minimum sample leaf, and bootstrap. The number of estimators is a hyperparameter that defines the number of decision trees in the random forest ensemble. Increasing the number of estimators typically improves the performance of the model, but it also increases computational cost. The max_depth parameter controls the maximum depth of each decision tree in the ensemble. A deeper tree can capture more complex relationships in the data, but it also increases the risk of overfitting. The minimum samples split parameter sets the minimum number of samples required to split an internal node. It helps prevent overfitting by controlling the minimum size of the nodes in the tree. The minimum sample leaf parameter defines the minimum number of samples required to be at a leaf node. Similar to minimum sample leaf, it helps prevent overfitting by controlling the minimum size of leaf nodes. Bootstrap is a Boolean parameter that indicates whether bootstrap samples are used when building trees. If True, each tree is built on a bootstrap sample (sampling with replacement), which introduces randomness and helps improve the robustness of the model. If False, the entire dataset is used to build each tree. A grid search was implemented to search for the best value of each hyperparameter to use in our model. We then calculated the loss function of our model before applying the optimized hyperparameters, which had a value of 0.360 which represents the average negative log likelihood of the predicted probabilities compared to the true binary labels. The model was retested with the implementation of the best hyperparameters according to the grid search which yielded a less performing model than the original model. The same result occurred when we attempted to implement boosting, sagging and gating to our model along with different combinations of each, for example an ensemble bagging and boosting. It was vital that we revisit our data preprocessing steps and feature

engineering, hence we went back to our original dataset before any data cleaning and started exploring greater options for feature engineering. The table below shows the changes in the features, as in what it used to be and what we changed it to.

Variable	Original Form	Changed Form
Country	One hot encoded to two options; Portugal or International	Each country got a different numerical encoding and remained under one column.
Agent	Dropped	Missing values filled with 0
Lead Time	Normalized using MinMax Scaler	Normalized using Log
Arrival Date Month/Year	Dropped	Kept and one hot encoded into numerical values
Babies and Children	Combined into one variable called kids, then dropped	Left as separate variables
Reservation Status Date	Kept as it is	Extracted the month and year and dropped original column
Outliers dropped out of dataset		

We also implemented a seed to ensure that any random operations performed later in the code will produce the same results each time the code is run. Our model was retrained and retested on the new dataset and obtained a much better performing model as shown below with an accuracy of approximately 93% and 0.206 entropy loss function.

```
Accuracy Score of Random Forest is : 0.9294739260280267
F1 score: 0.910
Classification Report :
              precision    recall  f1-score   support

           0       0.92      0.97      0.94      2574
           1       0.95      0.88      0.91      1779

    accuracy                           0.93      4353
   macro avg       0.93      0.92      0.93      4353
weighted avg       0.93      0.93      0.93      4353
```

To further improve our model we performed a hyperparameter grid search to find the best hyperparameters to use to for our model. We also split the dataset into training and testing data such that 80% of the data is used for training and 20% is reserved for testing. This yielded a better performing model, yet we further tested and dissected the accuracies of the model by examining the training and testing accuracies separately. We tested splitting the data such that 95% of the data is used for training and 5% is reserved for testing. The results of this model design is by far our highest performing one, with 99.7% training accuracy, this suggests that the model fits the training data very well and can accurately predict whether a booking will be canceled based on the features provided during training. The 94% testing accuracy indicates that the model performs well on unseen data as it correctly predicts the cancellation status of bookings with a high level of accuracy.

```
Training Accuracy: 0.996517027863777
Test Accuracy: 0.9377440845393982
F1 score: 0.921
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.97      0.95      2568
           1       0.96      0.89      0.92      1785

    accuracy                           0.94      4353
   macro avg       0.94      0.93      0.93      4353
weighted avg       0.94      0.94      0.94      4353
```

Utility Application

After finalizing our model design, we used pickle to save our trained model to use in our utility application. It's crucial to make this machine learning model accessible to customers and users for utilization. To achieve this, we will develop a web-based application offering a user-friendly interface. This interface will allow users to input their datasets or instances for prediction and apply machine learning algorithms to generate accurate predictions.

The application will be structured to guide users through the model and provide predictions for new instances. We'll leverage Streamlit, a Python package, to design and implement the application. Streamlit transforms the backend Python code containing the

machine learning algorithm into an application accessible through a generated URL, serving as the frontend. The Python code will handle both the interface design and the execution of the machine learning algorithm to deliver relevant information to users. To ensure seamless accessibility without assuming that the model resides on the user's machine, we'll deploy both the source code and the application on Streamlit Cloud. Users will interact with the application through an API, sending features and receiving predictions as responses.

Data Preprocessing

Once the data for the instance requiring prediction is inputted, the feature variables will undergo data preprocessing. Categorical variables will undergo one-hot encoding, a crucial step to maintain consistent model learning and performance. User validation will be implemented into the utility application such that the user will not be able to enter data of wrong types or missing values. Following this preprocessing, the instance will be presented to the user, reflecting the applied data preprocessing steps. The data is then prepared for input into the model.

Machine Learning Model

In the subsequent step, the application will present the selected machine learning model, which in this case is the Random Forest Classifier. The user input will be presented to our model and the prediction will be then generated based on the training data that our model has received.