Case 1

02582 Computational Data Analysis

s174173 & s183920

March 21, 2022

Introduction/Data description

Knowing how many passengers will be in an airport is crucial when planning how many staff members are required across the different sections of an airport among others things. Copenhagen Optimization is a company, that specializes in planning and analyzing airport operation. Copenhagen Optimization has developed a forecasting solution for the number of passengers called Better Forecast and to help improve this tool, this report aims at exploring machine learning methods for forecasting the number of passengers.

To forecast the number of passengers, an airport data set containing the variables seen in table 1, is used to fit various machine learning models. Here we aim at predicting the LoadFactor as this can easily be translated to number of passenger by the use the Seat Capacity variable and has the advantage of lying in a range around 0 to 1 making it suitable for most machine learning problems. A total of 39449 labeled observations from Jan 1st 2021 to Feb 28th 2022 were given for model selection and training. 4813 unlabelled flights observed between Mar 1st and Mar 31st 2022 were given as a test set.

Variable	**Description**
ScheduleTime	Date and time of departure for the flight
Airline	Abbreviation code for airline
FlightNumber	ID number of the flight
Destination	Abbreviation code for destination airport
AircraftType	Abbreviation code for aircraft type
FlightType	J (Scheduled flight), C (charter flight)
Sector	Country code for destination
Seat Capacity	Number of seats on the flights
LoadFactor	Share of seats occupied by passengers on the flight

Table 1: An overview and description of the varaibles contained in the airport data set. The table has been taken from the case description

To evaluate the performance the performance accuracy is defined by

$$Accuracy = 100\% - \left| \frac{y_{true} - y_{pred}}{y_{true}} \right| \quad (1)$$

where y indicates the number of passengers.

Data processing and feature engineering

Exploratory data analysis was performed to reveal missing values, skewed/unbalanced feature distributions. Observations with a 0 LoadFactor were removed to accommodate the accuracy metric. FlightNumbers were initially not unique across airline carriers, that is, some distinct carriers had flights that used the same FlightNumber. As we were unable to find real-world examples indicating a significance of shared FlightNumbers, each FlightNumber was prefixed with the airline abbreviation code to ensure intra-carrier uniqueness.

The exploratory analysis showed no missing values and no thought was therefore given on how to handle these as methods such as imputing the missing the value depends on what type of variable is missing.

The time-series aspect of the data was theorized to play a crucial role in predictive performance due to seasonal,weekly and daily trends and Covid-19 trends, so several features were derived from the ScheduleTime variable. These included: TimeOfDay, DayOfMonth, DayOfYear, DayOfWeek and isHoliday. Where isHoliday is based on common holidays such as christmas and new years eve along with some North American holidays such as 4th of July and Canada day, as around 96.5 % of the observations have their destination in either USA or Canada. Instead of using one-hot-encoding or a simple linear time-value, the first four of the mentioned features were Sin and Cos encoded to emphasize the cyclical patterns. [1]. For instance:

$$TimeOfDayCos = \cos\left(\frac{2\pi \cdot SecondsOfDay}{24 \cdot 60 \cdot 60s}\right) \quad (2)$$

DayOfWeek was one-hot-encoded since weekdays were deemed extra important. FlightType was likewise one-hot-encoded. Two dataset variants (X_{large} and X_{small}) were designed based on the remaining categorical features; Airline, Destination, AircraftType, FlightType, and Sector. X_{large} included the full one-hot-encoding resulting in more than 1400 features whereas X_{small} employed summary statitistics to reduce dimensionality. More specifically, LoadFactor mean and precision (inverse variance) estimates were calculated for each level of each feature. For instance, for an observation k with FlightNumber *BA811*, the FlightNumberMean is the average LoadFactor of all flights in the training split with FlightNumber *BA811*, *excluding* flight k. Finally, all features were standardized to zero mean and unit variance.

[1] https://towardsdatascience.com/cyclical-features-encoding-its-about-time-ce23581845ca

Relative difference (and by extension the accuracy metric) can be unforgiving for ground truth values close to 0. For instance, given ground truth passenger counts of 1 and 9, a prediction of 5 would yield -400% and 55% accuracy, respectively, despite the absolute prediction error being the same. For small differences between y_{true} and y_{pred}, relative difference is approximated by the MAE in log space[2]:

$$|\log y_{true} - \log y_{pred}| \approx \left| \frac{y_{true} - y_{pred}}{y_{true}} \right| \tag{3}$$

Since most algorithms can optimize wrt. MAE, a log-transformed variant of LoadFactor was designed. Model predictions made in log-space were inverse-transformed via an exponential before calculating accuracy. This simple transformation was theorized to help models shift attention towards flights with a low LoadFactor.

Models and methods

Model architectures

To find a suitable model for the problem, both classical and modern regression methods were tried. These methods included decisions trees, random forests, K-nearest neighbors (KNNs), Adaboost with a regression tree (AdaBoost RT), multi-layer perceptrons (MLP) and a Long Short Term Memory neural network (LSTM). The former algorithms were chosen as they cover a large part of the regression methods introduced in the course, are easy to implement and work well with the cross validation grid search method offered by `sklearn`. Decision trees and derivatives hereof are especially nice as they offer a large amount of non-linear modelling potential, with a variety of regularization methods to prevent overfitting. The AdaBoost addition sees multiple regression trees fitted in a sequential manner, where troublesome observations of prior trees are given more weight. The Random forest modification fits multiple trees, however, a feature subset is sampled for each fit. The LSTM and MLPs were added as deep learning methods have seen good success in recent years. The "No free lunch theorem" states that there is no single model suitable for all tasks, so we decided to test multiple architectures.

Hyperparameter-spaces

Tree-based methods shared similar hyperparameter search spaces: max-depth, min-samples-leaf, and n-estimators. For KNN, the only hyperparameter was the number of neighbors. For LSTMs, sequence length, and number of hidden units was optimized, where the sequence length refers to the numbers observations to use for predicting a single y, this was believed to add extra emphasis on the time-series aspect of the data. Lastly, for MLPs hyperparameters included

[2]https://en.wikipedia.org/wiki/Relative_change_and_differenceLogarithmic_scale

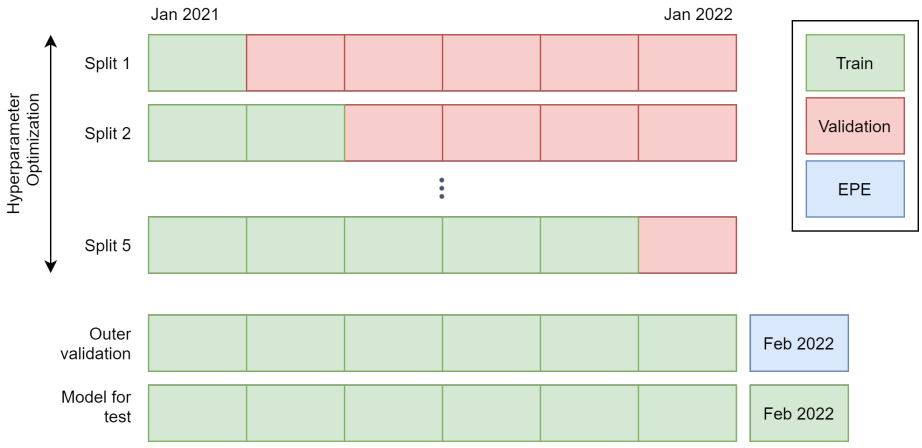

Figure 1: Validation scheme. The expected prediction accuracy (blue) was calculated only after each hyperparameter optimization had concluded.

dropout-rate, early stopping and number of hidden units. For all methods, both log and identity transform were tested, and where applicable, the MAE optimization criterion was tested against the default.

Model selection and validation

For each model architecture, hyperparameter candidates were narrowed by trial and error and subsequently optimized via either grid search or optimization toolkits. The latter was necessary in regression trees with MAE-criterion due to compute limitations. Prediction performance was measured by accuracy, and estimated via time-series aware cross validation splits. Put shortly, the time-aware aspect ensured that each model was validated only on data observed on a later date than the the data used for training the model. The expected prediction error would then mimic our ability to predict the future. Since we cannot assume the i.i.d condition on time-series data, this strategy kept the validation accuracies unbiased. Firstly, all observations of February 2022 were set aside for final calculation of expected prediction accuracy. During hyperparameter optimization the remaining data was partitioned in 5 time-based splits and the inner validation accuracy of these were used for hyperparameter optimization (see figure 1 for details). Finally, a model was chosen for predicting the Load-Factors of March 2022, by using the model with the largest outer validation accuracy (accuracy on the February 2022 data).

Results

Deep learning methods suffered from overfitting despite significant regularization (dropout ≥ 0.8), however, using early stopping the MLP achieved a fine accuracy. Half of the methods performed better in log-space, however, the AdaBoosted Regression Tree in identity-space achieved the highest score. All models performed significantly better using the reduced feature space X_{small} with hand-engineered features instead of the full one-hot encoding.

Model	Expected Prediction Accuracy
KNN	59.2* %
Decision Tree	60.5 %
AdaBoost RT	**64.3** %
Random Forest	63.8*%
LSTM	20.3 %
MLP	62.5* %

Table 2: Validation accuracy for each model with optimal parameters based on the training accuracy. *Performed best with log-transform. All models performed better on X_{small}. See appendix for list of optimal hyper parameters.

As the validation accuracy corresponds to our expected test accuracy, we choose the AdaBoost Decision Tree algorithm for making predictions on the test set from March.

The results are available as predicted LoadFactors both in a separate text file (predictions.txt) and as a column appended to the original test excel file (predictions.xlsx).

Closing remarks

This work presented several feature-engineering and data-processing methods for the prediction of passenger load factor in planes. We expect our LoadFactor predictions for March 2022 to achieve an accuracy of 64.3%. Although this value constitutes the best of five models and thus incurs a selection bias, we find this to be almost negligible as Adaboost achieved high accuracy for different sets of hyperparameters. For future work, it could be interesting to include other factors such as Corona disease numbers and ticket prices. Furthermore, optimization and prepossessing choices were often biased to accommodate the limited computing power. It could be interesting to redo the analysis on a more powerful machine as this would allow full grid-searches and methods such as the "1 standard deviation rule".

Furthermore, it is worth noticing that the contest on getting the expected accuracy as close as possible to the accuracy obtained on the test set, could be won by changing all the LoadFactor predictions to 0, as our expected prediction accuracy would be exactly 0.0% due to definition of the accuracy.

Appendix

Optimal hyper parameters

Here we present a list of the optimal hyper parameters producing the accuracies displayed in table 2.

- KNN: Number of Neighbours = 7
- Decision Tree: Max depth = 71, Min samples leaf = 22
- AdaBoost DT: max depth = 24, Number of estimators = 40
- Random Forest: criterion=MAE, min_sample_leaf=6, n_estimators:25
- LSTM: Sequence length = 5, Number of hidden units = 64
- MLP: Dropout: 0.8, activation=Leaky ReLU, hidden units=[200, 100, 100]

Accuracy across hyperparameters

Below is seen plots showing the performance of some of the models. Here the parameters are shown in different ways as e.g. colors, axis values and markers. Furthermore, the label "Inner CV accuracy" refers to the mean accuracy obtained when fitting cross validation for the model, while "Outer CV accuracy" refers to the accuracy obtained by using the model to predict on the validation set from February 2022.

Figure 2: Decision tree performance on the inner CV loop. The highest inner CV accuracy was found for a decision tree with max depth at 71 and min samples leaf at 22.

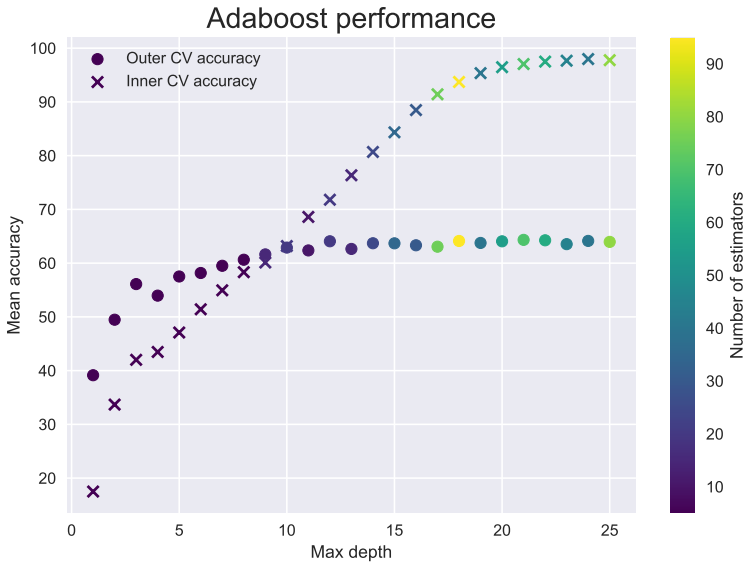

Figure 3: AdaBoost RT perfomance for both outer and inner CV loop. It is seen that the inner CV accuracy keeps increasing with the max depth, but that it does not affect the outer CV accuracy. The optimal AdaBoost DT was found with a max depth of 24 and 40 estimators.