Forecasting for Airports: Passenger Forecasting Case

02582 Computational Data Analysis

s212487 & s213237

March 21, 2022

1 Introduction

On a daily basis, there is a large number of planes landing and taking off to provide services to millions of passengers transiting through airports. Emerging needs to optimize airports have been identified to ensure the efficiency of their services. This is the case of "Better Forecast", a forecasting solution for airports developed at the Copenhagen Optimization.
Using machine learning approaches, this case aims to contribute its grain of sand by dealing with flight-related data to improve the forecast of the expected number of passengers per flight as a share of seats. This load factor is stated to be the foundation of optimized airport operations.
However, one also needs to take into account the COVID19 situation and how could have impacted on the passenger numbers.

1.1 Data description

Two data-sets were provided containing 8 features (categorical and numerical) related to flight information: (1)the schedule time of the flight, (2)the airline, (3)the flight number,(4)its destination, (5)the aircraft and (6) flight type, (7)its destination code sector and (8) the seat capacity of the flight. The first data-set consists of a total of 39449 observations with the realized data from 2021-01-01 to 2022-02-28, which also includes the obtained load factor per each flight. On the other hand, the second data-set (n=4814 flights) was generated for the period 2022-03-01 to 2022-03-31, to be able to predict its load factors for all flights in March. In that way, the realized schedules (1st data-set) were used to train and select the models with the highest accuracy according to the predicted and the actual number of passengers per flight. As a preliminary step, data preprocessing and feature selection were performed on the original data as described below.

2 Model and methods

With the aim to find the highest accuracy passenger forecasts and best suitable models, we first implemented a preprocessing step, including missing values and normalization, followed by feature selection.

2.1 Data preprocessing and Missing values

In this stage, it was first considered if there were missing values in the whole training set. As all observations (n=39449) have all its corresponding features, no row removal was contemplated.

In terms of the 'Time Schedule' feature, it was split in standard date and time unit values (day, month, year, hour and minute of flight departure), leading to an increase of features in our data-set. Additionally, by inspecting the load factor over time (Figure 1), extra features that were thought to be more representative were added due to the increased trend by the end of the year. These were: Day of the year, day of the week, its quarter, and a Boolean vale to represent if the date belongs either to the start or the end of the month, quarter or year, respectively. Note that if these features resulted to be have a higher influence in the load factor, they would be chosen by feature selection.

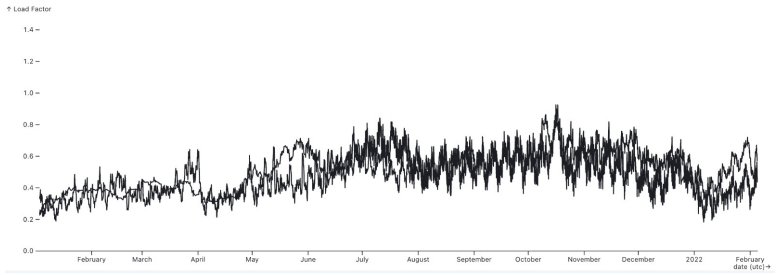


Figure 1: Average of Load Factor per flight in the course of month (from 2021-01-01 to 2022-02-28).

The type of each feature, either categorical or numerical, was then considered (Table 1). As the Aircraft-Type variable had a mix of strings and integers in some observations, they were all converted to a string.

Features	Type	Unique values	Features	Type	Unique values
Airline	Categorical	105	Year	Categorical	2
Flight Number	Numerical	825	Hour	Numerical	Range(0-23)
Destination	Categorical	230	Minutes	Numerical	Range(0-59)
Aircraft Type	Categorical	58	Day of Year	Numerical	Range(1-365)
Flight Type	Categorical	3	Day of Week	Numerical	Range(1-7)
Sector	Categorical	12	Quarter	Numerical	Range(1-4)
Seat Capacity	Numerical	Range(10-451)	Month Start/End	Categorical	2
Day	Numerical	Range(1-31)	Year Start/End	Categorical	2
Month	Numerical	Range(1-12)	Quarter Start/End	Categorical	2

Table 1: Description of feature types of the training-set.

Referring to the categorical features above, the so-called One Hot Encoding procedure was applied to them, converting each value into a new categorical column and assign a binary value of 1 or 0 to those columns. After all, we ended up with a training set of 424 features per each observation. Feature standardization was applied to all numerical features to be centered around the mean with a unit standard deviation ($\mu_i = 0$, $\sigma_i = 1$ per i feature).

2.2 Model selection

Several Machine Learning models were trained to identify the most accurate in terms of passenger forecasting per flight. The most significant ones are presented below, where we used 70% of the observations in order to train the model. The remaining ones were kept to validate the model (Section 3).

Simultaneously, **feature selection** was conducted for each model to find the optimal number of features, specifically we used the wrapper method: Backward elimination. The procedure consists of first feeding all the possible features to the model and iteratively removing the worst performing ones in terms of the performance metric pvalue ($p > 0.05$ was the default threshold). Besides, we applied **K folds cross-validation** to tune and optimize parameters inside the 70% observations assigned for the training for each considered model.

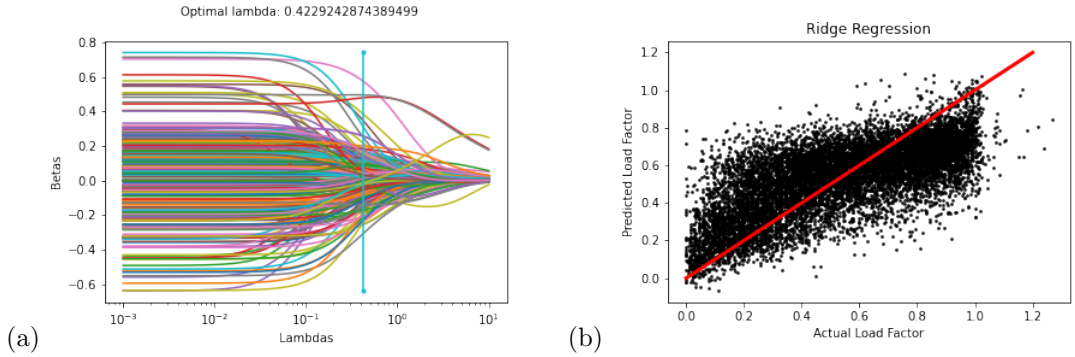
2.2.1 Linear Regression Algorithms

The first supervised models being trained were the Ordinary Least Square (OLS) Regression and the Ridge Regression. Here the goal was to estimate the unknown parameters β in a linear regression model by minimizing the difference between the collected observations and its predictions (Eq. 1 and 2). Firstly, by applying backward elimination for selecting representative features, we ended up with a total of 259 features, being the 62% of the total number. For instance, it was observed that features such the day number of the schedule fight and minutes were excluded, while the day of the week or the year were kept.

$$\beta_{OLS} = min_{\beta} \, ||y - X\beta| \, |_2^2 \tag{1}$$

$$\beta_{RIDGE} = min_{\beta} \, ||y - X\beta| \, |_2^2 + \lambda \, ||\beta| \, |_2^2 \tag{2}$$

Considering the Ridge Regression algorithm, the regularization parameter λ was optimized using 10-fold cross-validation. This parameter is in charge of controlling the amount of shrinkage. Figure 2.a shows the optimal value of λ that has given the smaller mean square error (MSE) for each fold.



(a) (b)

Figure 2: Ridge Regression Model: (a) Estimation of β's parameters as a function of the λ parameter. (b) Distribution of the predicted load factor on the Test Set vs the actual ones.

Figure 2.b presents the distributions of the predicted load factors on the test set as a function of the true values. In this case, but also in the OLS algorithm, the total accuracy resulted to be of less than 50%, reason why it was not selected in the next steps.

2.2.2 K-Nearest Neighbors

The next algorithm considered was the K-Nearest Neighbors as it can be also used for regression problems. As before, features were selected by backward elimination to obtain those that contribute the most.

The hyper-parameter par excellence when using K-Nearest Neighbors algorithm turns out to be the number of neighbors. In this case, its tuning has been performed by a 5 fold CV. Two parallel procedures were used, aiming to find the optimal number of knn depending on the weight function: either 'uniform' (all points in each neighborhood have equal weights) or 'distance' (closer neighbors have greater influence). Results have shown that the highest accuracy was obtained when the number of neighbors was equal to 3 and the weight set to 'distance' (Figure 3.a).
Moreover,two other more features have been optimized using cross-validation: the 'Algorithm' used to compute the nearest neighbors and the 'Power' parameter for the Minkowski metric (p). With respect to the 'Algorithm', no significant differences in the accuracy metric was observed, although technically speaking, 'BallTree' has given the highest value. In contrast, p=1, being equivalent to using the Manhattan distance, has given a significant improvement (Figure 3.b).

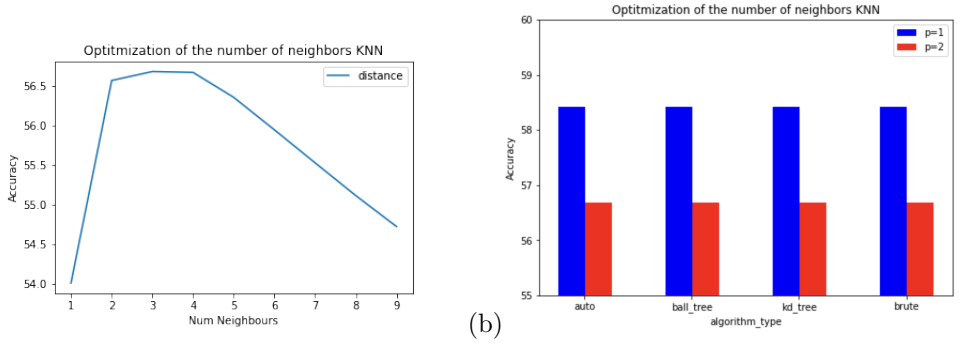


Figure 3: KNN Model Selection with CV: (a) Predicted Accuracy vs. the number of neighbors. (b) Predicted Accuracy vs. the algorithm considered.

2.2.3 Random Forest Algorithm

In this case, a similar procedure as the ones described above has been followed, as it is allowed in regression tasks. In terms of model selection, the parameter to be tuned was the number of estimators (number of trees in the forest) which is the one that usually gives more variability depending on the data. As seen in Figure 4.a, the optimal number of estimators found was 300.

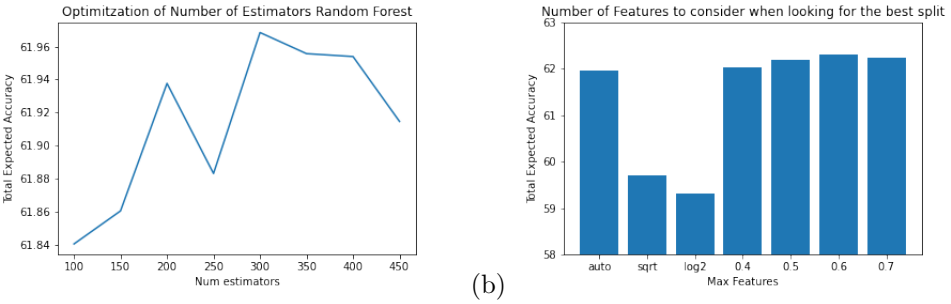


Figure 4: Random Forest Model Selection with CV: (a) Predicted Accuracy vs. the number of estimators. (b)Predicted Accuracy vs. the number of features.

The next parameter optimized was the number of features to consider when looking for the best split ($max_{features}$). A value of 0.6 has been to give the highest accuracy (Figure 4.b). Lastly,

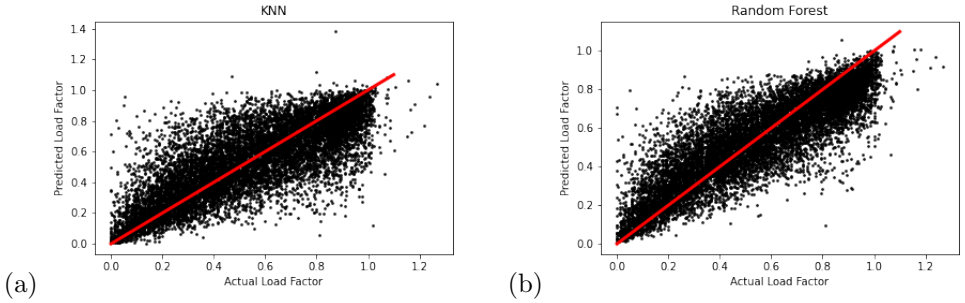
min_samples_split and max_samples parameters have also been tuned. However, no improvement was observed when compared to the default values.

3 Model validation

The K-Nearest Neighbors and Random Forest (presented in section 2.2) were validated considering their higher accuracy compared to the linear regression models. To assess them, the predicted number of passengers per flight was compared to the real-world observations with the purpose to estimate the expected total accuracy, being the mean of accuracies per flight (Equation 3). The actual and forecasted number of passengers was obtained by multiplying the load factor by the seat capacity. Note that we just considered 30% of the observations that were not used to train the model. The few flights where the actual passengers were found to be zero were also excluded from the accuracy calculation due to the inability to have a division by zero.

$$\text{Accuracy per flight}(\%) = \left(1 - \left|\frac{\text{Actual passengers-Forecasted passengers}}{\text{Actual Passengers}}\right|\right) \times 100\% \qquad (3)$$

Considering the K-Nearest Neighbors, an accuracy of **57.57%** was obtained with the remaining 30% of the dataset (testing data), after tuning the parameters (model selection). This significantly improves the 52.02% obtained if the algorithm was used with default parameters. For the Random Forest, an accuracy of **61.23%** was observed after model selection, which slightly improves the 60.49% obtained if the algorithm was set to the default parameters. See figure 5 for the distribution of the predicted load factor values vs the true ones for both cases.



(a) (b)

Figure 5: Predicted Load Factor vs Actual Load Factor values. (a) KNN. (b) Random Forest.

4 Results

By considering section 3, the best model resulted to be Random Forest Regressor, being the one used to predict the load factors (numbers of passengers divided by seat capacity) for all flights in March (from 2022-03-01 to 2022-03-31).

In order to estimate **the expected total accuracy**, the data-set to train the model was used (realized data). We split it into a training and a testing set, this second one having the size of the future schedule data (n=4813, being the last values of the data). Then, the remaining data was used again to train the Random Forest Regressor (with the according tuned hyper-parameters). In this context, a total expected accuracy of **59.43%** was obtained, which is considered to be the expected total accuracy when predicting the values for flights in March. Overall, the accuracy is thought to be far from perfect, which suggests that future work on estimating new relevant features should be done.