Case 1

02582 Computational Data Analysis

s164397 & s164419

March 21, 2022

Introduction/Data description

The data provided for Case 1 consists of 39389 observations of flight data in a time series format. The response variable which we want to predict is *LoadFactor*, which is the share of seats occupied by passengers on a flight. Observations with a LoadFactor of 0 (60 observations) are dropped to avoid issues later on. Given that this is a numeric value shows that we are dealing with a regression problem. There are 8 features which we will utilise to predict the target variable. All variables are listed in Table 1.

Name	**Value**	**Type**
LoadFactor	[0;1]	Numeric
ScheduleTime		Numeric
SeatCapacity	[10;451]	Numeric
FlightNumber		Categorical
Destination	$[A\text{-}Z]^+$	Categorical
AircraftType	$[A\text{-}Z0\text{-}9]^+$	Categorical
FlightType	$\{J, C, G\}$	Categorical
Sector	$[A\text{-}Z]^+$	Categorical
Airline	$[A\text{-}Z]^+$	Categorical

Table 1: Overview of input and output variables

Model and method

Model selection

In this section we describe how we select the best performing model and justify our decisions. The goal is to balance bias and variance such that we obtain a model which generalises well to unseen data.

With the high number of categorical variables we are motivated to utilise the properties a tree based method as these handle categorical variables very well. We ended up going for the Random Forest regression because of its simplicity, speed and explainability.

The tree based methods offers a balance between variance and bias through its many hyperparameters. A rule of thumb is that the bigger the tree, the larger the variance.

Factor handling

We were given quite a few categorical variables which we have then chosen to represent as factor variables since a lot of machine learning algorithms have a hard time using labels as input variables. We have chosen to deal with this by using one-hot-encoding of the categorical variables so we can represent them numerically where columns for each of the unique values of a given category is created where a 1 is used to indicate the presence of said value.

This resulted in a staggering **1236 columns** from our original 9. In order to avoid overfitting we would like to lower the number of features. We made the choice to keep only the columns which met a certain variance threshold. This will cause the extremely sparse one-hot-encoded columns to be dropped as they would have to have a certain amount of 1s given the threshold. Through hyperparameter tuning we found the threshold to be $\tau = 14 \cdot 10^{-4}$ **resulting in 439 features**.

Feature transformation

To retain all information packed in `ScheduleTime` while also having it being easily interpreted by the Random Forest algorithm as well as humans we opted for converting it to its corresponding UNIX timestamp which is the number of seconds passed since January 1st 1970. This way we still have all time-related information for each individual flight in a single numeric variable as opposed if we were to decompose the time onto several columns. It might be worth considering doing both for future reference as it would capture a broader image with little to no detriments.

Model validation

Splitting strategy

Firstly, we will put aside 20% of the latest acquired data in order to validate the model's performance later on. The rest of the data will be used to select the best model using cross-validation.

Due to the data being a time series we have to be careful during cross validation. When data is independently distributed you would usually shuffle the data before splitting to introduce randomness in terms of how the data was gathered. It is quite easy to convince yourself that flight data is affected by the time of which the observation was made. This can be seen in Figure 1 where we plot the average LoadFactor for each week of the year. This motivates us to employ a different splitting strategy.

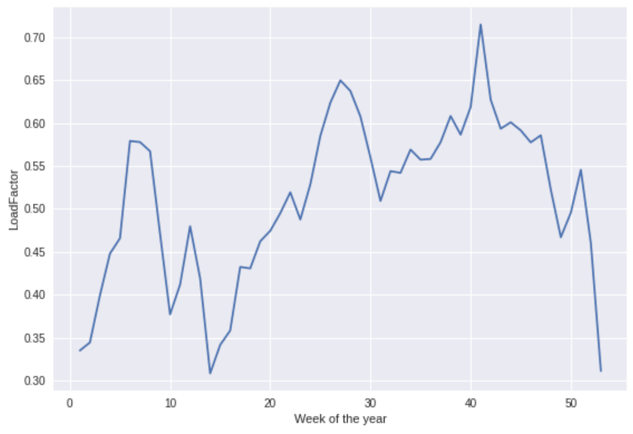

Figure 1: Example of how `LoadFactor` varies depending on the season

As the realised data is the month prior to the month we are to predict a strategy could be to make n cuts in the data and make sure that the test data is immediately followed by the training data. This is illustrated in Figure 2 and is done to as closely reflect what we are trying to do for the final model.

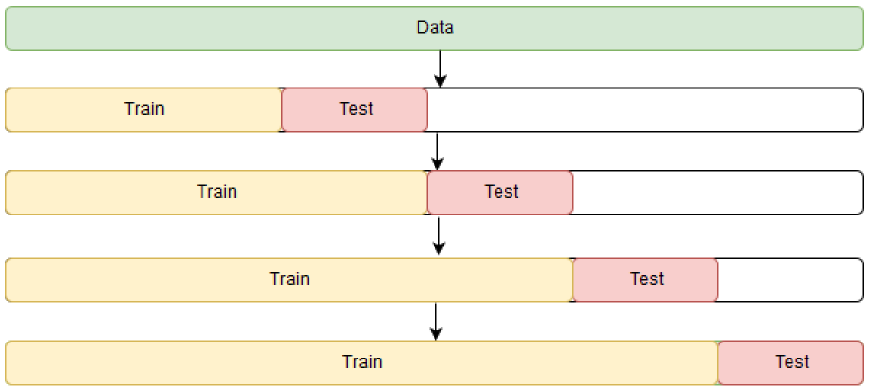

Figure 2: Example of Time Series Split with $n = 4$

We will be using $n = 5$ splits such that the set set grows from being 1/5th, to 2/5th ... 4/5th while always being followed directly by the test set. This has the advantage of making it reflect the actual problem more but comes at the expense of being harder to assess robustness as we only have n static folds to score on as we saw a large variance of accuracy score across folds.

Hyperparameters

To find the best suited hyperparameters we used a tool by the name of *Optuna* [1] which enables us to easily do bayesian optimisation. This was done as opposed to grid search to hopefully decrease the number of trials having to be run to obtain good results. Each trial of parameters was evaluated such that we are looking to maximise average accuracy per flight (as described in the case description) over the time series splits. We chose to average the splits' accuracies such that the optimisation would weigh the splits, i.e. time, equally. The hyperparameters tuned were `n_estimators`, `max_leaf_nodes` as well as the variance threshold.

Results

For the final model we found the hyperparemeters as shown in Table 2

Name	**Value**
`n_estimators`	1100
`max_leaf_nodes`	2500
τ	$14 \cdot 10^{-4}$

Table 2: Hyperparameters chosen for the final model.

Another advantage in regards to interpretability of the random forest is that we have direct and easy access to the *feature importances*, which are a metric which states how important a given feature was in regards to obtaining good results. As seen on Figure 3 we can see that the scheduled time as well as whether or not the flight was from the the largest airline made good variables to predict the LoadFactor.

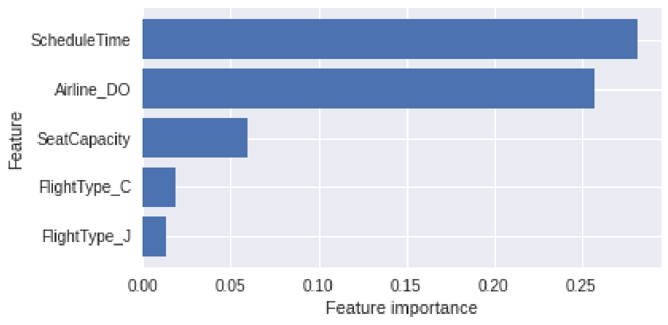

Figure 3: Feature importance of the final model

To validate these hyperparameters we used them on the remaining 20% of unseen data and got an accuracy per flight of 46.48%

References

[1] Takuya Akiba et al. “Optuna: A Next-generation Hyperparameter Optimization Framework”. In: *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2019.