Case 1

02582 Computational Data Analysis

Magnus Haraldson Høie

March 21, 2022

Introduction/Data description

Github link: `https://github.com/Magnushhoie/02852_Copenhagen_Optimization_Case`

This project build a Random Forest regressor to predict the relative fraction of occupied flight seats (Load Factor) for planned flights from the Copenhagen Optimization flight dataset. The case documents are available for the project Github link.

The training dataset comprises 39 449 flights between the time-period of 1st January 2021 to 28th February 2022. The features include the the flight scheduled calendar time including time of day, flight number, airline, destination aircraft type, flight type, geographical sector and seat capacity. Of these, only the scheduled time and seat capacity feature is continuous and not a categorical feature. Additionally, all flights in the training dataset contains the load factor, which is the percentage of seats occupied at flight time relative to the available seats.

A testing dataset is additionally provided, with the planned flights and same features as described above for March 2022. The test dataset does not contain the load factor.

The aim of the project is to find the best prediction model and feature engineered dataset to minimize the mean absolute error for the March 2022 test dataset. This error is finally calculated as the average accuracy per flight, or the mean absolute error subtracted from 1.

Model and method

The final model is a Random Forest regression model. The dataset is pre-processed to convert all categorical features to continuous features by estimating the average target value for each unique value in the categorical feature. The Random Forest then treats the continuous values as relative ranks for each feature, as described in the Factor Handling section. The Random Forest can in theory split on an exact flight, airport or sector by first splitting on the rank

immediately below, then immediately above the unique value. Since Random Forests are rank-based, no normalization was required for any of the features.

Model selection

I decided on using February 2022 as the validation set for model selection, as it is closest to the test-set month of March 2022 and is in an underrepresented year containing only two month of data in the provided dataset.

For sake of establishing a baseline, I attempted to model the data with sklearn's Linear, Lasso and Ridge models. These all performed significantly worse than my final model as calculated with mean absolute error on the validation month. This was also expected due to the limitations in these models assuming linear relationship between features and target values, and inability to create e.g. flight or airport specific "sub-models" to more closely model the data trends.

For the final model, I decided on using a Random Forest decision tree model. Random Forests consist of an ensemble of decision trees, which individually closely model the input data by mapping successively smaller subsets of the data in a tree-like structure. Each split the tree is based on selection of one feature and a threshold value for that feature, which best splits the remaining samples into two groups such that the samples in each group have a similar mean target value. This metric is named the node or group purity, and is 100 % when all samples in the group share the same average target value.

The decision-tree structure allows the model to exactly fit the patterns of the training set, such as trends in individual airports and flights over time, including specific weekdays or months of the year. The model is not limited to linear relationships. Unlike a linear model, a random forest would not necessarily be biased towards assuming that e.g. the load factor for flights to Copenhagen affect the patterns in load factor for local flights in the US.

I performed some light hyperparameter tuning of the model by setting the max_features variable to 50 %. This lets the individual decision trees only consider a random subset of the features for building the tree, and helps with regularization. I found that adjusting the number of estimators, changing the loss function from mean squared error to mean absolute error or manipulating the allow sample sizes for node splits only significantly worsened results or provided minimal improvements. For the sake of keeping the model simple and avoid possible over-fitting to the training data, I therefore kept the remaining settings at default.

Missing values

The raw dataset did not contain any missing values, while feature engineering was only time-based or consisted of mapping of categorical unique values to a continuous rank-based scale as described in Factor handling.

Factor handling

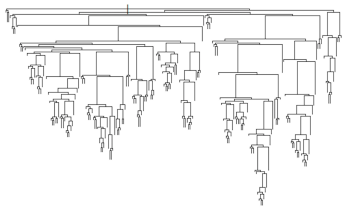

Figure 1: Idealized decision tree splitting on continuous values with high node purity

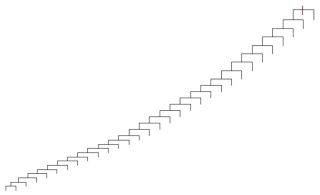

Figure 2: Decision tree with only one-hot encoded features and very low node purity

I first attempted to one-hot encode all categorical features, including but not limited to the destinations, airlines and flight types. This resulted in a processed dataframe exceeding 1200 columns.

A severe disadvantage of this is that from a decision tree's point of view, all one-hot encoded variables are independent. With the large increase in sparsity of the data, the model will struggle to effectively split the samples as it potential gain in purity per node split is severely limited.

A better method is to find an encoding method that effectively represents sample features in a continuous rank-based format. This can for example be achieved by extracting each unique value per categorical feature, and calculating the difference between the mean target value when that value is and is not present across samples (see code example below).

Random Forests are effectively rank-based models, by virtue of how their decision trees split the data. For each node split, a cut-off point is decided based on which threshold will maximize the purity of the split. This means that the the numerical value mapping above can effectively be interpreted as assigning a relative ranking of the unique feature values. For example, a flight that has the highest mean load-factor in the dataset could be given a rank of 1, while an average flight given a rank of 231.

I note that as long as the relative ranking is kept for the unique feature values, their absolute value will not impact the decision tree splits. For this reason, I did not further order and convert the mapped values calculated across the unique values for each categorical feature to actual integer ranks.

```
# Iterate through categorical feature columns
for col in cat_cols:
    map_dict = {}

    # Extract unique categorical values for this feature
    # E.g. flight numbers 899, 903 and E21
    uniq = df_proc[col].unique()

    # For each unique value, calculate the mean target value when
                                    the categorical value is
                                    present and not present
    for value in uniq:
        # A mask allows us to select a subset or everything but the
                                        subset of the data
        mask = df_proc[col] == value

        # Calculate difference between the two groups
        delta = targets[mask].mean() - targets[~mask].mean()

        # Add value to a mapping dict
        map_dict[value] = delta

    # Use dict to map original feature values to the new numerical
                                    values
    df_proc[col] = df_proc[col].map(map_dict)
```

Figure 3: Python code for converting categorical values into continuous by mean target value

An overview of the engineered features used in the final Random Forest model is shown below.

Feature	Type	Range	Description
Time delta	Continuous	0.00-15.00	Fractional time in months since Jan 1st 2021
Time of day	Continuous	0.00-24.00	24-hour time in fractional format
Day of year	Continuous	1-365	Day of year
Day of week	Continuous	1-7	Day of week
Day of month	Continuous	1-31	Day of month
FlightCount	Continuous	(-)1.00-1.00	Weekly number of flights for given FlightNumber
FlightNumber	Continuous	(-)1.00-1.00	Delta target value of flight vs not flight
Destination	Continuous	(-)1.00-1.00	Delta target value of Destination vs not Destination
Airline	Continuous	(-)1.00-1.00	Delta target value of Airline vs not Airline
AircraftType	Continuous	(-)1.00-1.00	Delta target value of AircraftType vs not AircraftType
SeatCapacity	Continuous	(-)1.00-1.00	Flight seat capacity as provded

Feature table

Model validation

For model validation, I used a leave-one-month-out cross-validation method. I selected one month in the range January 2021 - February 2022 as the validation month, and trained the Random Forest model on the remaining months. This was successively performed for all months.

For each month, the mean accuracy per flight was calculated from the mean absolute error across all flights.

$$\text{Mean accuracy} = 1 - \frac{1}{N}\sum |y - \hat{y}|$$

The estimated accuracy for March 2022 was calculated based on the mean value for the leave-one-month out prediction for December 21 through January and February 22. These months are close in time to the test month of March and more importantly comprise the change of year to 2022 with a resetting of the feature values for day of the year and expected yearly trends. The extra month of December 21 was included to include an additional third data-point still recent in time to March 22.

Results

The best model uses the Scikit-Learn RandomForestRegressor with 0.50 max_features using the feature table described in factor handling.

The estimated accuracy for March 2022 is 0.84067, while the mean leave-one-month-out accuracy was calculated to be 0.8650. I observed

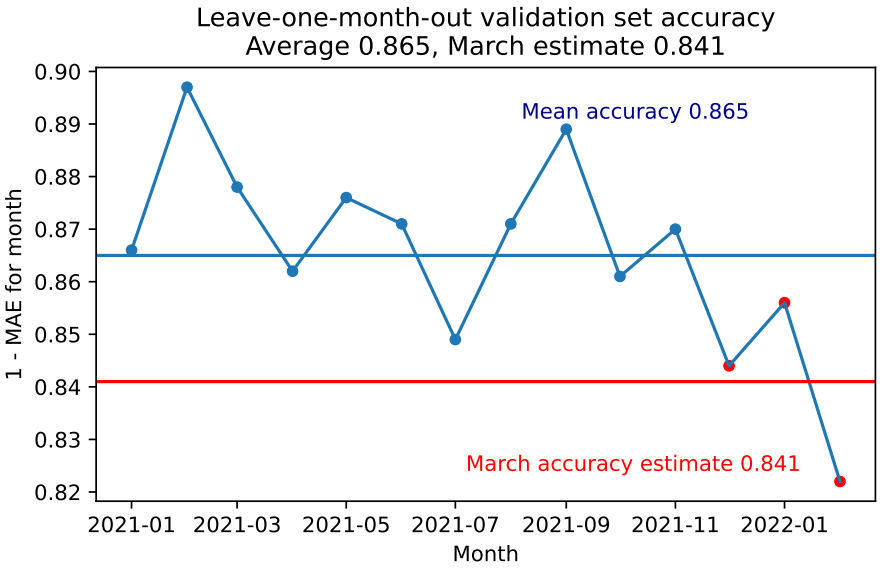

Figure 4: Leave-one-month out cross-validation accuracy, as 1 - mean absolute error per month. The estimated prediction error for March 2022 is estimated as **0.84068**.

that the leave-one-month-out accuracy plot trended towards a decrease in accuracy from November 21, and for that reason decided to only base the March estimate from the three most recent months to March 22.

I plotted the relative feature importance and observed that the Flight Number ranking value was used ca. 1/3 of the time in the decision tree splits across the random forest. This is expected as different routine flights will likely show highly specific trends in their load factor. The next most important features were the destination, day of the year, airline and a time delta feature representing how recent in time the flight is to now. I observed that the sector and flight type played only a very minor role in the Random Forest model predictions.

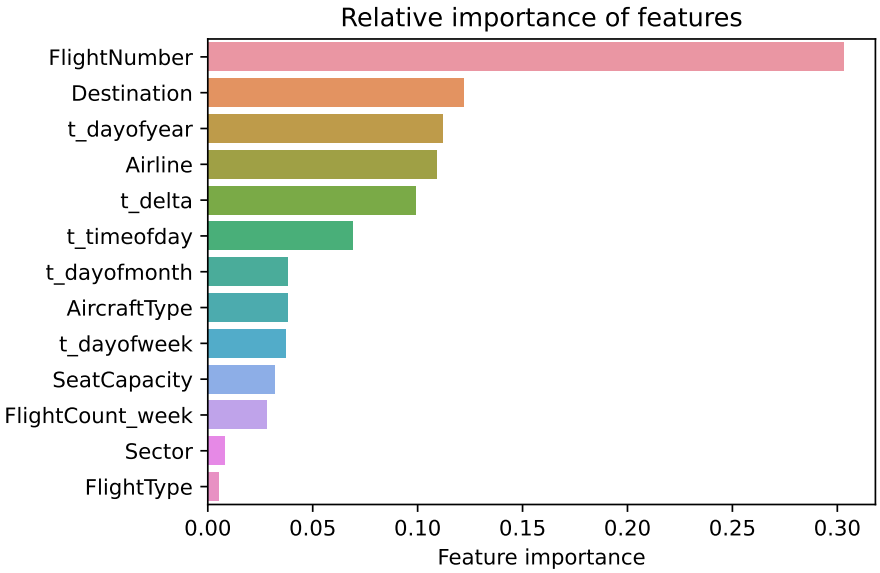

Figure 5: Random Forest feature importance when trained on all samples. Feature engineering are described in the Factor handling section.