Case 1

02582 Computational Data Analysis

Holger Christian Nyeland Ehlers (s182521) & Renjue Sun (s181294)

April 4, 2022

Introduction & Description of Data

The aim of the project is to predict the discrete feature 'Load Factor' using machine learning. The 'Load Factor' feature describes the number of passengers per flight as a share of total seats. Hence, if forecasting of this could be improved it would be interesting. The realized data set is from 2021-01-01 to 2022-02-28 and contains 9 features and 39449 data points with no missing values. The realized data set is provided by Copenhagen Optimization. The final model will be tested on data from the entire month of March 2022.

Table 1 describes features in regard of their statistic types, for example, whether they are nominal, ordinal, interval or ratio and whether they are either discrete or continuous. The mean and standard deviation of two ratio features are calculated and the standard deviation of both 'SeatCapacity' and 'LoadFactor' are quite large in reference with their mean.

Table 1: Basic statistic summary of features from realized data set.

-	Features	Type	Mean	Std.
1	ScheduleTime	Interval - Discrete	-	-
2	Airline	Nominal - Discrete	-	-
3	FlightNumber	Nominal - Discrete	-	-
4	Destination	Nominal - Discrete	-	-
5	AircraftType	Nominal - Discrete	-	-
6	FlightType	Nominal - Discrete	-	-
7	Sector	Nominal - Discrete	-	-
8	SeatCapacity	Ratio - Discrete	155.65	58.65
9	LoadFactor	Ratio - Discrete	0.54	0.27

Data pre-processing

The 'ScheduleTime' is split into four features. They are 'Year' – nominal and either 2021 or 2022, 'DaysPassed' – ratio denoting days from new years day, 'MinutesPassed' – ratio denoting minutes from beginning of the day and 'DayofWeek' – nominal from 0 (Monday) to 6 (Sunday).

Holiday indicator

A feature named 'HolidayIndicator' was also introduced in the light of effect of holidays. The assumption is that the load factor can be larger 9 days before and after the holiday and the effect can also relate to the length of holidays. For example, Easter holiday is a whole week of 7 days, therefore the value of feature is 7 on Easter day (04-04-2021). It changes to 6.3 one day before Easter (03-04-2021) and -6.3 one day after Easter (05-04-2021). Positive values denote days before Easter and negative values denote days after Easter. The feature values will return to 0 on the 10th day before and after Easter. The feature values around Easter day in 2021 are illustrated in Figure 1.

The same strategy is supposed to apply to all holidays for a general prediction. However our prediction target is March of 2022 rather than a whole year. Therefore only Easter is considered since it is the only holiday near March.

After the split of 'ScheduleTime' feature and the introduction of 'HolidayIndicator' feature, there are 13 features in total.

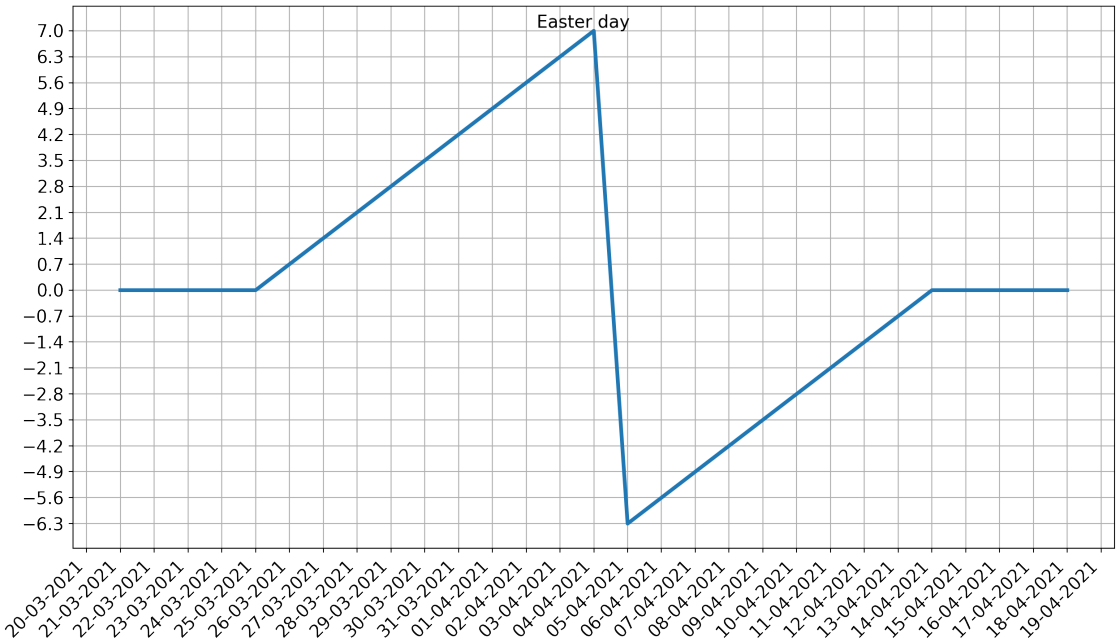

Figure 1: Holiday indicator for Easter holiday of 2021.

Factor handling

Label encoders are applied to transform nominal features to categorical integers. For example, available observations are from both 2021 and 2022, therefore the 'Year' attribute is categorized to 0 (2021) and 1 (2022). All features except the target are then standardized by subtracting the mean and dividing by the standard deviation.

$$\hat{\mu}_j = \frac{1}{N}\sum_{i=1}^{N} X_{ij} \quad , \quad \hat{\sigma}_j = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(X_{ij} - u_j)^2} \tag{1}$$

Then obtaining the new matrix when subtracting the mean divided by the standard deviation:

$$\tilde{\mathbf{X}} = \begin{bmatrix} \cdots & (X_{1j} - \hat{\mu}_j)/\hat{\sigma}_j & \cdots \\ \cdots & (X_{2j} - \hat{\mu}_j)/\hat{\sigma}_j & \cdots \\ & \vdots & \\ \cdots & (X_{Nj} - \hat{\mu}_j)/\hat{\sigma}_j & \cdots \end{bmatrix} \tag{2}$$

The standardization is done within folds when using cross validation in order to avoid leaking of information from training set to test set due to different splits.

Models and methods

The models are implemented in Python utilizing several libraries including SciPy, Sklearn, NumPy and Pandas. Data, code and auxiliary documentation is available at GitHub, *https://github.com/HCEhlers/02582_MLComp*. Ordinary Least Square regression is used as a baseline model.

Figure 2 describes pairwise correlation for each feature using Pearson correlation coefficient to assess the correlation between the features. The absolute value of the β's can be used as a crude measure of feature importance as illustrated in Figure 3.

Significant correlations discovered in Figure 2 are between 'Year' and 'Days passed' (this is because the realized data set contains all months of 2021 but only the first two months of 2022), 'Destination' and 'Sector', 'Aircraft type' and 'Seat capacity', and 'Aircraft type' and 'Sector'. Combining the two figures, we can draw a crude conclusion that 'Airline', 'Days passed', 'Aircraft type' and 'Flight type' are quite important for predicting the load factor. 'Day of week', 'Year' and 'Seat capacity' are more important than 'Holiday indicator', 'Flight number' and 'Destination'. 'Minutes passed' and 'Sector' are the least important features to predict the target.

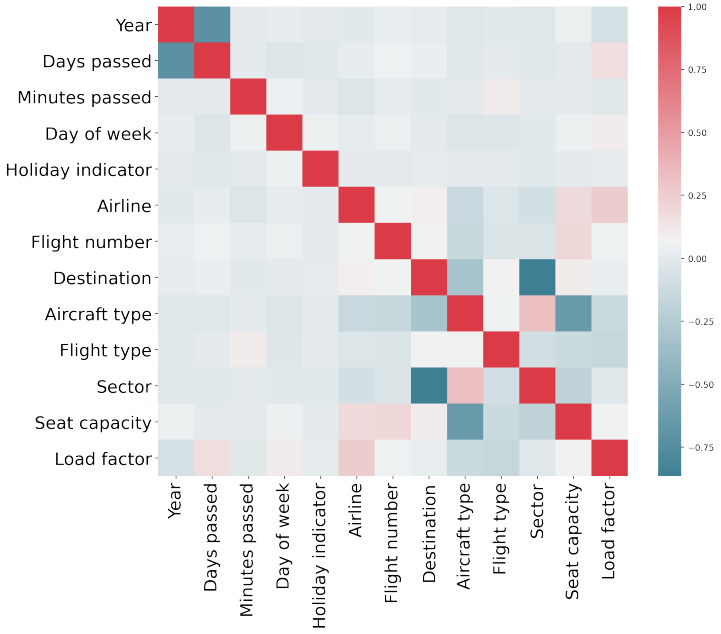

Figure 2: Pairwise correlation for each feature.

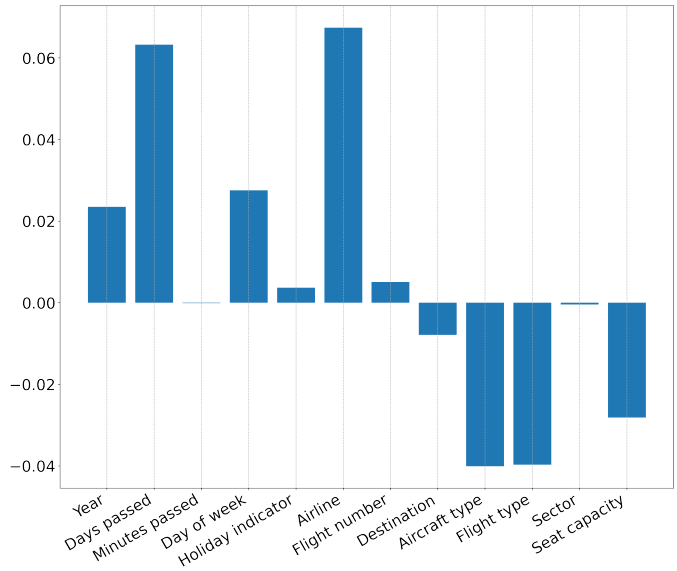

Figure 3: OLS coefficients as a measure of feature importance.

Model description

Ordinary Least Square (OLS) regression

OLS is an unbiased model without any hyper-parameter and β_{OLS} can be simply calculated by Equation 3.

$$\begin{aligned} \beta_{OLS} &= \underset{\beta}{\operatorname{argmin}} \|y - X\beta\|^2 \\ \beta_{OLS} &= (X^T X)^{-1} X^T y \end{aligned} \tag{3}$$

Ridge regression

Ridge regression adds L_2-penalty to the unbiased OLS, which increases the linearity of the model by introducing a shrinkage to all coefficents of variables to predict the target. The larger the λ, the more linear the model. β_{ridge} can be calculated by Equation 4.

$$\begin{aligned} \hat{\beta}_{ridge} &= \underset{\beta}{\operatorname{argmin}} \|y - X\beta\|^2 + \lambda\|\beta\|^2 \\ \beta_{ridge} &= (X^T X + \lambda I)^{-1} X^T y \end{aligned} \tag{4}$$

Lasso regression

Lasso regularization adds the absolute value of the coefficients to the loss function. Least Angle Regression (LARS) can be used in Lasso for feature selection which picks important features first and can discard redundant features ($\beta_j = 0$).

$$\hat{\beta}_{lasso} = \underset{\beta}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\} \tag{5}$$

AdaBoost regression

The core of AdaBoost regression is to continuously build new regression trees to minimize residuals of all observations with focus on observations with large residuals in previously built regression tree. The 'score' of each tree (Amount of Say) is large when the total residual is small and this also gives observations with large residuals larger weights when building the next tree.

$$\text{Amount of Say} = \frac{1}{2} \log \frac{1 - \text{Total Error}}{\text{Total Error}} \tag{6}$$

KNN regression

We used another supervised learning algorithm, K-nearest neighbor. The prediction of a response can be more accurate when referring to observations that share similar values of features, for example, the same period of a year, the same airline and destination. This is our motivation of investigating KNN regression.

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i, \tag{7}$$

where $N_k(x)$ is the neighborhood of x defined by the k closest points x_i in the training set. Closeness can be measured using the Euclidean distance.

Model selection

10-fold cross validation (CV) is the tool we used for selecting optimal parameters and comparing the test accuracy of proposed models. The test accuracy is calculated by Equation 8 with $K = 10$ where Y and $\hat{r}(X)$ denote observed and predicted load factors in test splits respectively, which follows the 'Accuracy per flight' equation provided in the case description. The model selection is trying to maximize the test accuracy.

$$\mathbb{E}[\text{TestAccuracy}(\hat{r})] = \frac{1}{K} \sum_{i=1}^{K} \mathbb{E}_i[1 - \mid 1 - \frac{\hat{r}(X)}{Y} \mid] \tag{8}$$

OLS, Ridge and Lasso

We used CV with 10 folds to select optimal λ of Ridge and Lasso. We also tried information criteria AIC and BIC. BIC selected the simplest model with largest λ and CV selected the most complex model with smallest λ. AIC selected larger λ than CV but they perform similarly. (see Figure 5 and Figure 6 in Appendix)

Ridge is the best model among OLS, Ridge and Lasso. The main reason is that the shrinkage applied to all coefficients makes the model more flexible and prevents the model from overfitting the training set as rigid OLS does. As Lasso can have coefficients with values of exact 0 ($\beta_j = 0$), we thought that the model performance might be improved when discarding a few less important features and we applied LARS for feature selection. However, the result turns out to be worse than OLS and Ridge since the regularization is too strong.

AdaBoost regression

Parameters to tune are `maximum tree depth` - from 1 to 20, `loss function` - AdaBoostRegressor provides 'linear', 'square' and 'exponential', `the number of growing trees` - from 10 to 150, `learning rate` - from 0.0025 to 1. We used CV with 10 folds to select optimal parameters. We firstly applied CV to calculate the average test accuracy combining 20 maximum tree depths with 3 loss functions with fixed number of growing trees 25 and fixed learning rate 0.25. The result reveals that maximum tree depth 13 with square loss function performs the best. We later applied CV again with fixed maximum tree depth 13 and square loss function to select best combination of the number of growing trees and learning rate. The optimal number of growing trees 130 and learning rate 0.66 were acquired. (see Figure 7, Figure 8 and Figure 9 in Appendix)

To sum up, the selected model is composed of 130 trees with maximum tree depth 13, learning rate at 0.66 and square loss function.

KNN regression

We used 10 fold CV to select the optimal number of neighbors as well as selecting the best weighting scheme, either distance based or uniform. The model is trained using the Euclidean distance as distance metric and to maximize the mean of accuracies per flight. The average test accuracy was calculated for the combination of neighbors and weighting scheme. According to Figure 10 and Figure 11 in Appendix, the selected model should use $K = 12$ with distance based weighting scheme. For $K < 12$, the model is supposed to be underfitting and for $K > 12$, the model starts to overfit the training set.

Compared to other models KNN seems to perform best with highest test accuracy according to Table 3 in the Results section.

Model validation

We used 10 fold CV for model selection with different parameters for each model and found that distance weighted KNN regression gives the best performance with $K = 12$. We now applied 2-layer cross-validation to combine both model selection and model validation for KNN regression.

Figure 4 illustrates how nested CV works with the number of outer folds $K_1 = 10$ and the number of inner folds $K_2 = 10$. We used inner folds to select optimal parameters and then fit the optimal models with the training sets of outer folds. We then validated selected models using test sets of outer folds. The test sets of outer folds are therefore also validation sets.

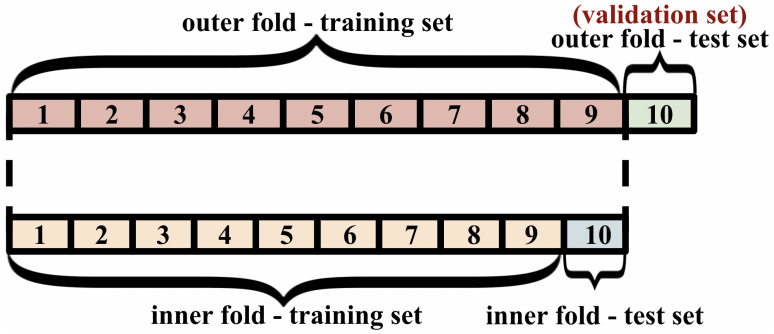

Figure 4: Nested cross validation with 10 outer folds and 10 inner folds.

Expected Prediction Error (EPE) & Expected Total Accuracy (ETA)

EPE is composed of 3 parts – the noise of observed response, the deviation between observed and predicted response and the noise of predicted response. Here EPE is calculated by mean square error of each validation set, where Y and $\hat{r}(X)$ denote actual and forecasted load factors in validation data respectively:

$$\text{Expected Prediction Error (EPE)} = \mathbb{E}[\text{ValidationErr}(\hat{r})] = \mathbb{E}[(Y - \hat{r}(X))^2] \tag{9}$$

ETA is calculated by the mean of accuracies per flight of each validation set:

$$\text{Expected Total Accuracy (ETA)} = \mathbb{E}[\text{ValidationAccuracy}(\hat{r})] = \mathbb{E}[1 - \mid 1 - \frac{\hat{r}(X)}{Y} \mid] \tag{10}$$

Table 2 illustrates EPE and ETA based on 10 validation sets with different selection of parameters according to inner folds of nested CV. 'dis' and 'uni' denote distance weighting and uniform weighting respectively. We picked the 8^{th} validation set with highest ETA and the same K (12) and weighting scheme (distance) as we discovered in model selection of KNN regression.

Table 2: Summary of model validation based on 10 test sets of outer folds (validation sets).

Outer fold	1	2	3	4	5	6	7	**8**	9	10
K	11	15	15	15	15	15	15	**12**	15	15
Weighting	uni	dis	dis	uni	uni	uni	uni	**dis**	dis	uni
EPE	0.05	0.05	0.05	0.05	0.05	0.05	0.05	**0.05**	0.05	0.05
ETA	0.35	0.29	0.26	0.374	0.3	0.33	0.33	**0.375**	0.27	0.33

Results

Table 3 describes the test accuracy calculated following Equation 8 with optimal parameters of each proposed models. The values are therefore supposed to be the highest test accuracy that each model achieved in model selection. KNN regression stands out and is selected to be our final model to make the prediction for March of 2022. Our `Expected Prediction Error is 0.05` calculated by Equation 9 and our suggested `Expected Total Accuracy is 0.3747 (37.47% in percentage)` calculated by Equation 10. They are both based on distance weighted KNN with 12 nearest neighbors using validation data.

Table 3: Summary of test accuracies using 10 fold Cross Validation for model selection.

Model	OLS	Lasso	Ridge	AdaBoost	**KNN**
Test accuracy using CV	-0.0024	-0.0700	0.0792	0.0665	**0.3213**

Appendix

Figures for Ridge regression

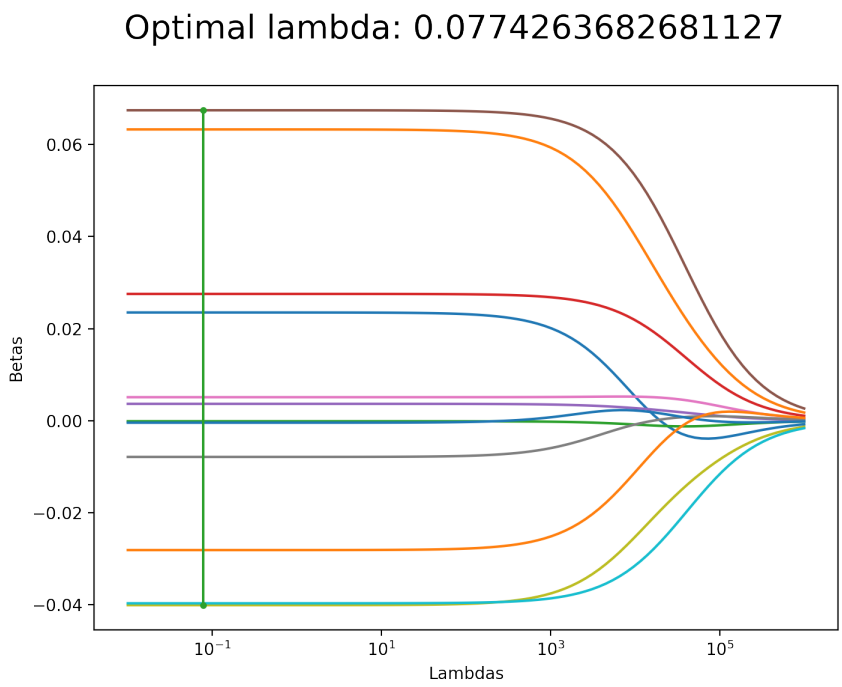

Figure 5: Model selection (the selection of λ) with 10 fold cross validation for Ridge regression.

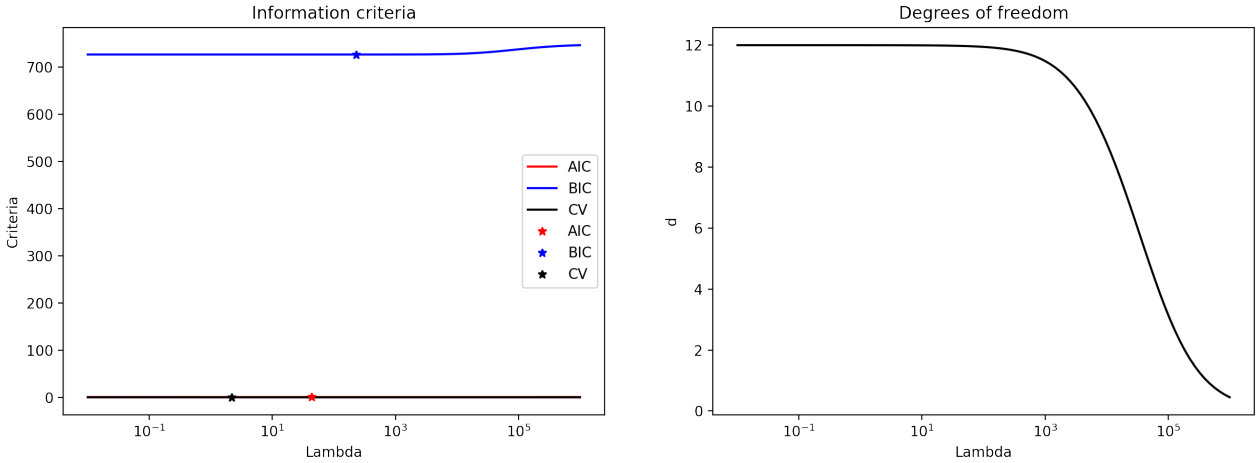

Figure 6: Information criteria for Ridge regression.

Figures for AdaBoost regression

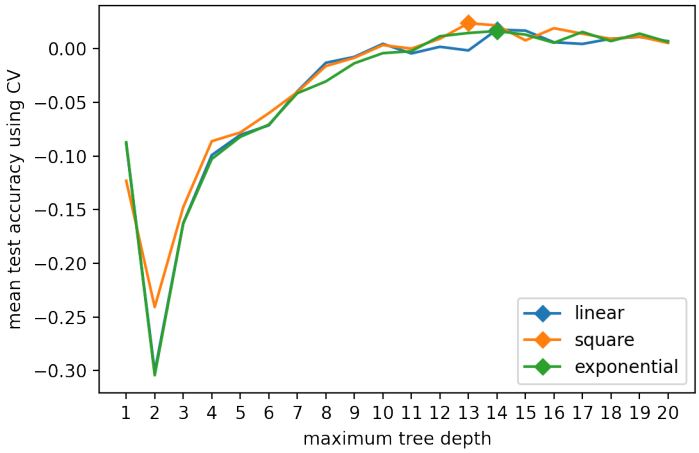

Figure 7: Results from 10 fold cross validation using different maximum tree depths and loss functions.

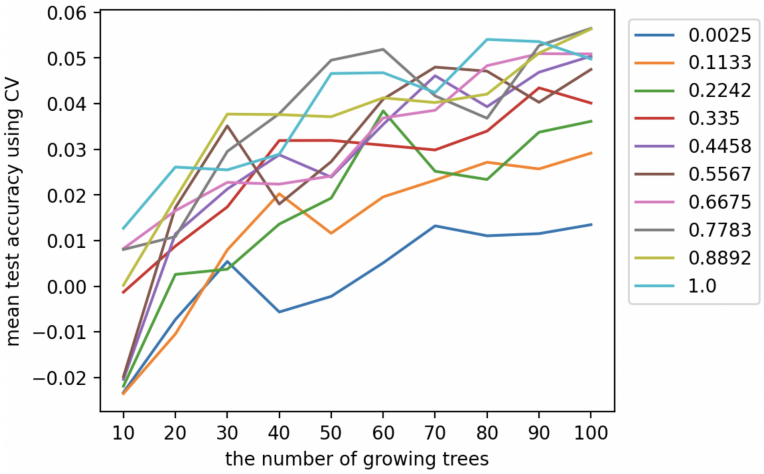

Figure 8: Results from 10 fold cross validation for crude selection of the number of growing trees and learning rates.

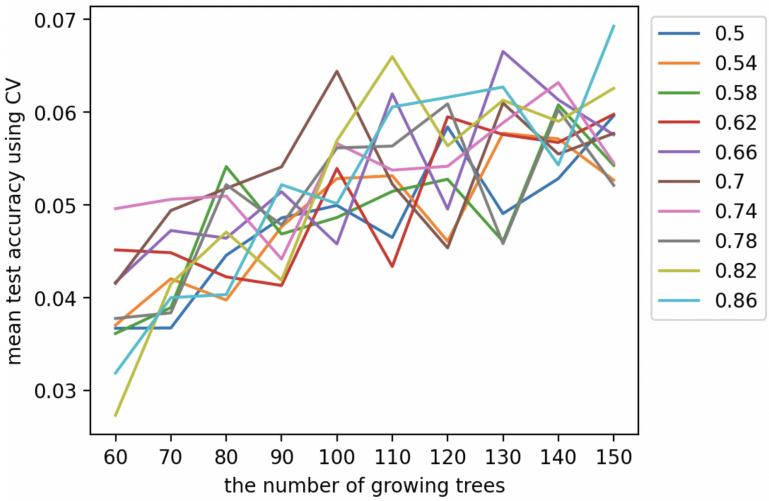

Figure 9: Results from 10 fold cross validation for precise selection of the number of growing trees and learning rates.

Figures for KNN regression

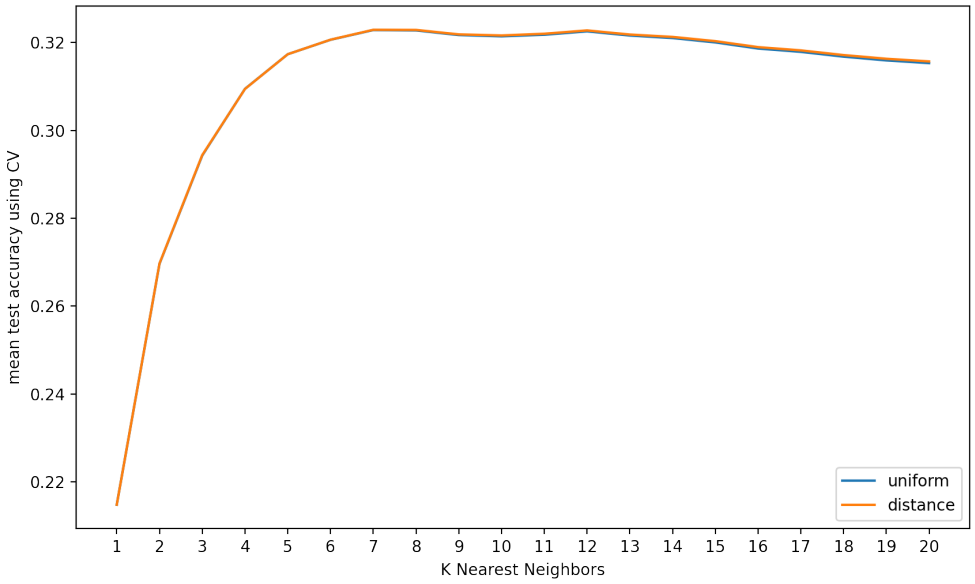

Figure 10: Results from 10 fold CV using uniform weighting or distance based weighting for a crude selection of K.

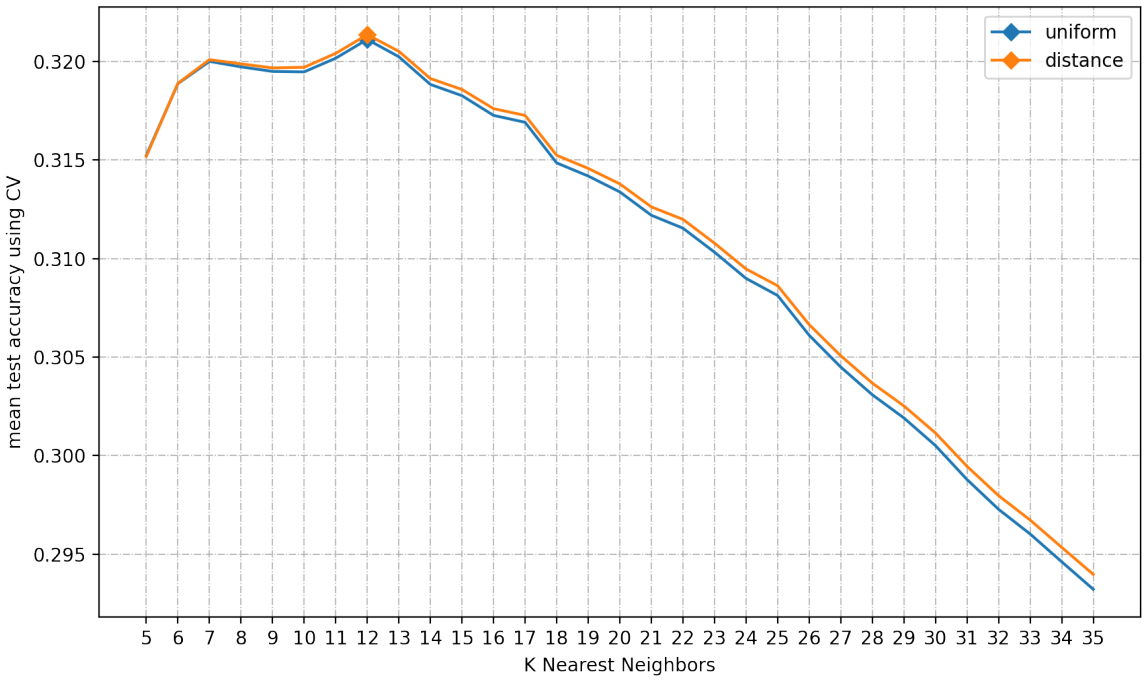

Figure 11: Results from 10 fold CV using uniform weighting or distance based weighting for a precise selection of K.