Graded Assignment 2 - Forecasting using Regression and Ensembles
Jon-Fredrik Hopland

Task 1
The task is to develop a forecasting system for daily demand using regression models. By applying a sliding window technique on the 2022 data, I will create and train decision tree regressors, a custom random forest, and an ensemble of multilayer perceptron regressors. The best performing model will be selected to generate predictions for the rest of 2023's demands.

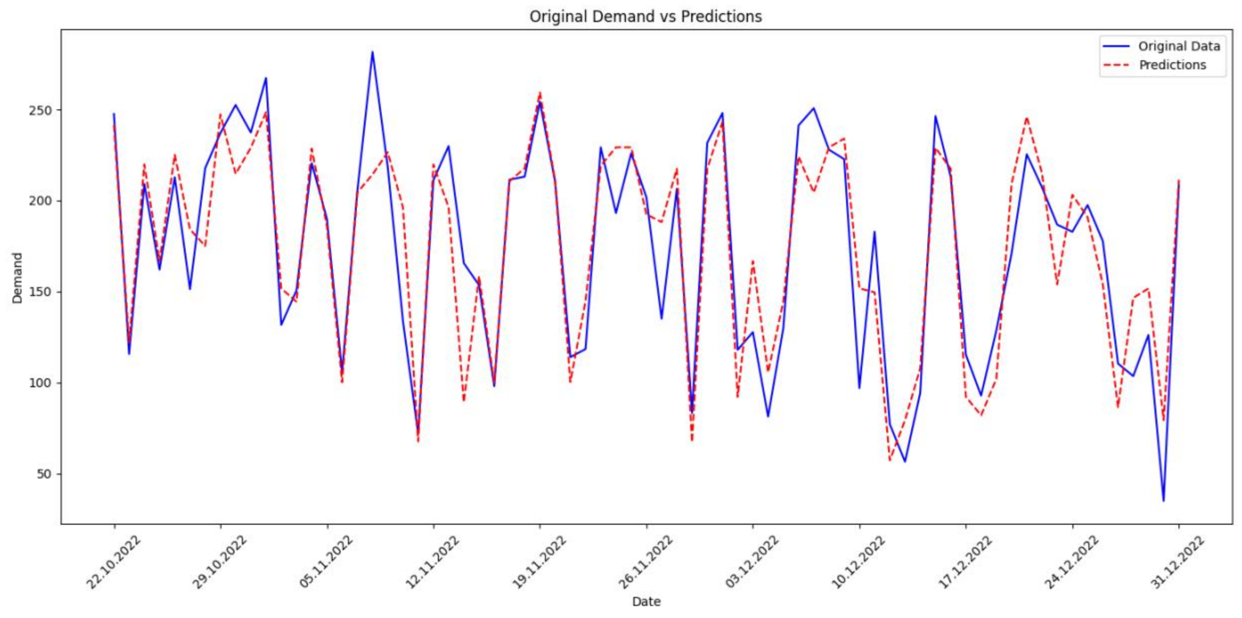
Task 2
In task 2, the goal was to implement a decision tree regressor to forcast the demand.

First we started preparing the data by reading the data_2022.csv file and using the sliding window approach. To do this I created a function "load_and_prepare_data". The function reads the demand data and creates a dataset suitable for regression analysis using the sliding window approach. I set the window size to 10, meaning that each prediction is based on the demand of the previous 10 days.

I then implemented the train_decision_tree function. Here, we initialize and the regression tree model. The DecisionTreeRegressor is used with a fixed random state. This is a seed for the random generator, so by setting it to 0 we should get the same result every time we train the model.

I realized that I had to split the data into a training set and a test set. For this I used train_test_split from sickit-learn. I used 80% of the data for training and 20% for testing. I split the data to make sure it could generalize well and to validate the model's performance. I trained the decision tree on the training set.

Then, for the last step, we make predictions on the test set and plot the actual data and the predictions. The predictions were generated by passing the test set inputs through the trained decision tree model. We can see that the model can capture the general trend in the data.



Task 3

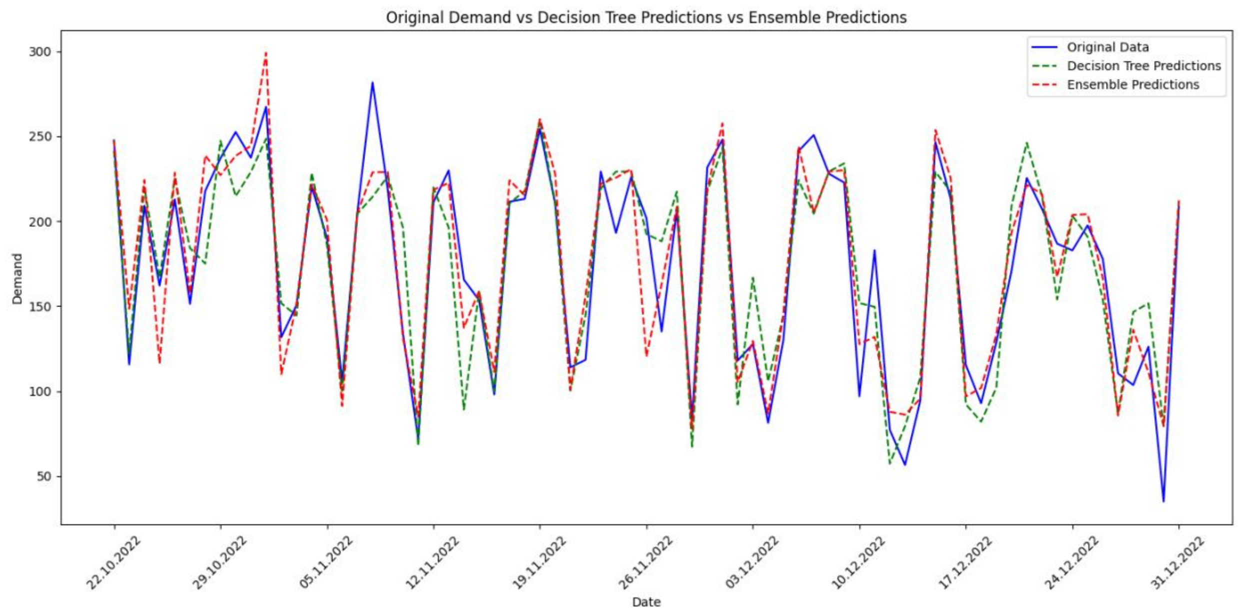
In Task 3, I developed a custom implementation of a random forest to improve the accuracy of demand forecasting for Angstrom Fruitcake Co. Now I'm going to briefly explain each step.

I created a total of 100 trees, where I randomly generated hyperparameters for each tree. For the ensemble creation, I created a function create_ensemble that initializes an empty list to hold individual trees. For each of the n_estimators specified, a bootstrapped sample of the training dataset was created using the bootstrap_sample function.

For training each tree i used bootstrapping. I did this to provide each tree with a slightly different perspective of the data, creating a better and more robust ensemble. I used the bootstrap_sample for this.

To aggregate the predictions from all individual trees, I created the predict_ensemble function. This takes the predictions from each tree in the ensemble to get the average of these predictions. This is effective in reducing the variance of the model's predictions and usually gives us a more accurate and stable model. It also smoothens the predictions and helps to prevent overfitting.

For the last part, I visualize the predictions from the ensembles as well as the predictions from task 2 and compare them to each other and the original data. We can see that compared to the single decision tree, the ensemble's predictions are smoother and appear to be a better fit to the original demand curve. This probably means that the ensemble is more effective in capturing the general trend and is likely to generalize better to unseen data.



Task 4

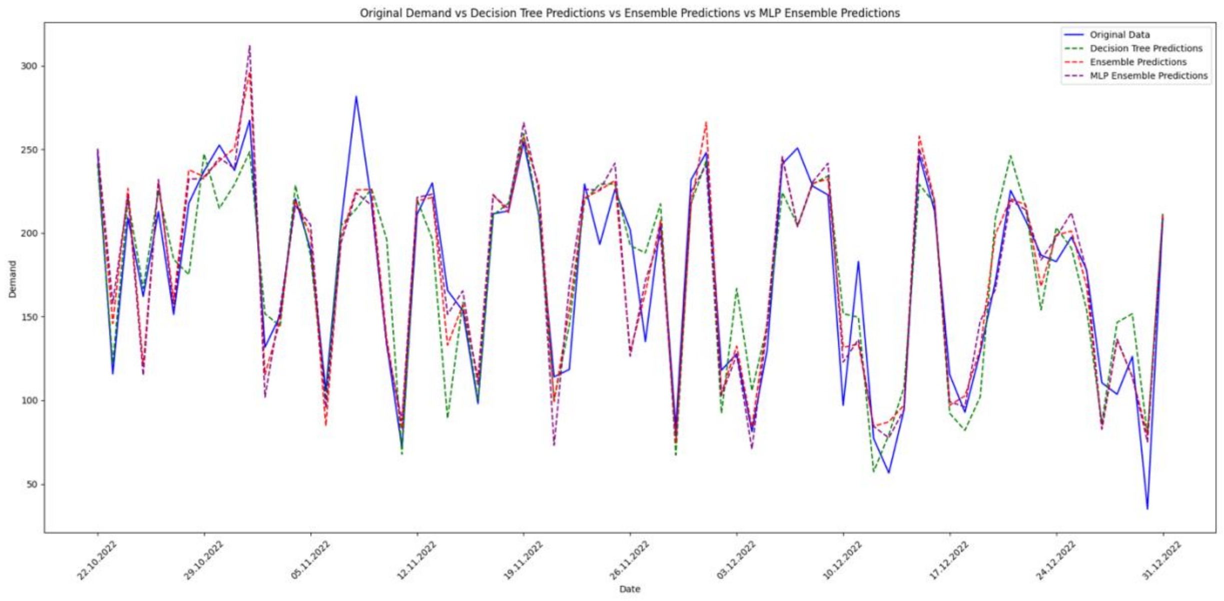
Moving on to Task 4, the goal was to create an ensemble of Multi-Layer Perceptrons (MLP).

I initiated the process by constructing an ensemble of MLPRegressor instances — create_mlp_ensemble. I kept each MLP consistent with a single hidden layer of 100 neurons, while making sure that the results were reproducable by setting random_state=0, like I've done throughout the whole assignment. I created an ensemble of 100 estimators.

For training I used bagging. I individually trained each MLP on unique bootstrap samples to get some diversity and randomness among the models. This step was computationally intensive.

For the predictions, I implemented the predict_with_mlp_ensemble function, aggregating forecasts from each MLP to average out the noise and individual errors.

When comparing all the models it seems that the MLP ensemble I quite balanced and captures the data's complexity without overfitting. But we can see that all the models are relatively similar many places.

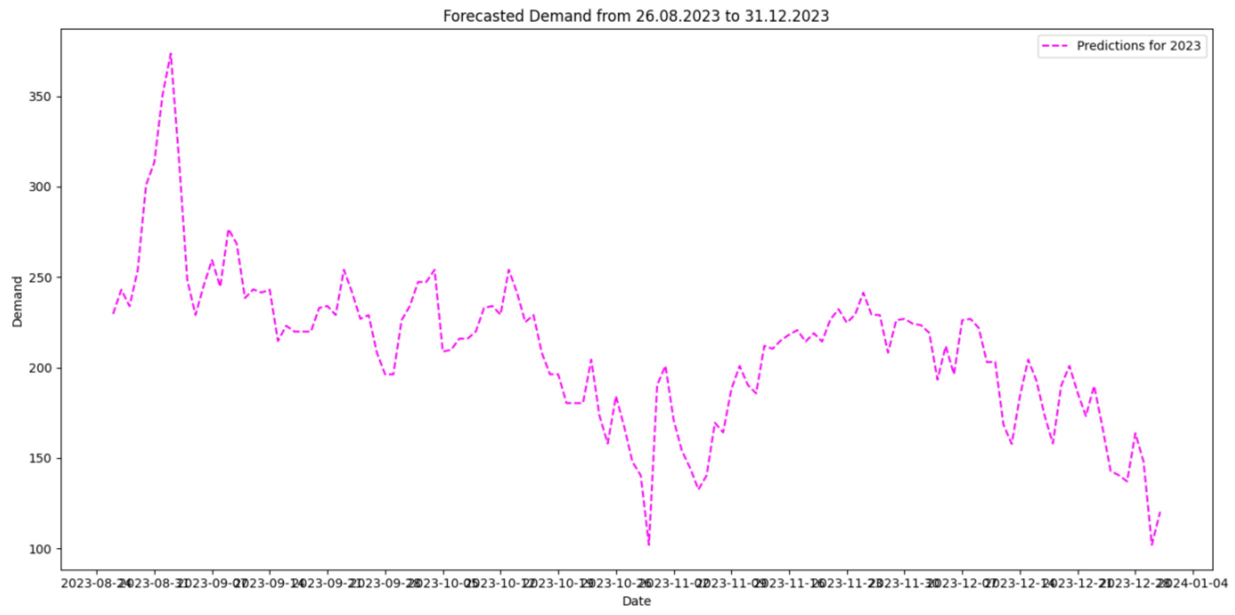


Task 5

In task 5 I first loaded the "data_2023.csv" file. I then prepared the data up to 25.08.2023 using the familiar sliding window technique with a window size of 10, consistent with our previous approach. This was to create a foundation for our predictions from 26.08.2023 to the end of the year.

Then I selected the decision tree model trained on 2022 data for this year's predictions. I implemented a function predict_demand to generate daily demand forecasts from 26.08.2023 to 31.12.2023. I was surprised that the decision tree model from task 2 was the best model, so I'm suspecting that something is not quite right in my code for task 5. However, I was unable to resolve the potential issue.

Next I plotted the prediction for the given date range.


Forecasted Demand from 26.08.2023 to 31.12.2023
Predictions for 2023
Demand
Date