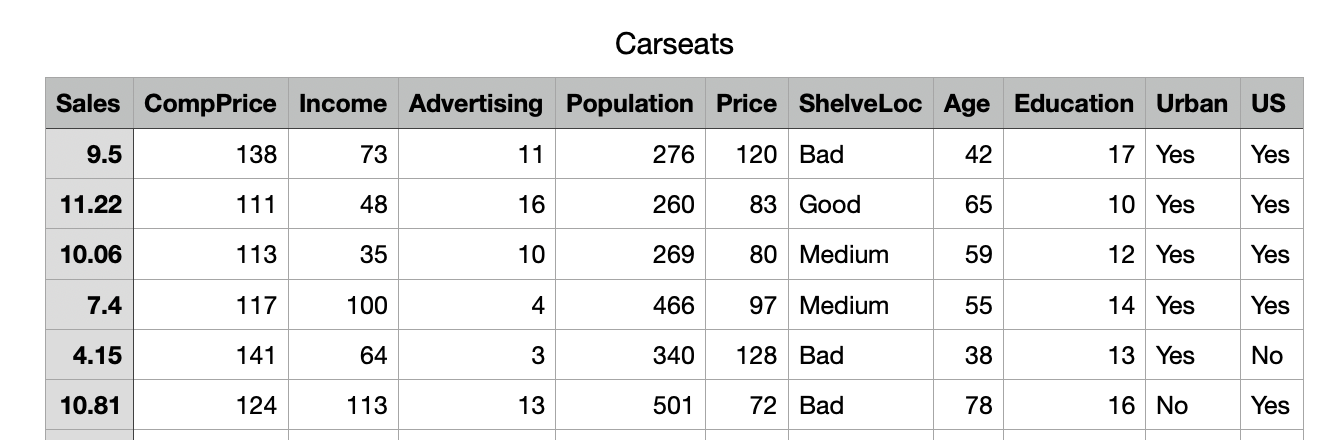
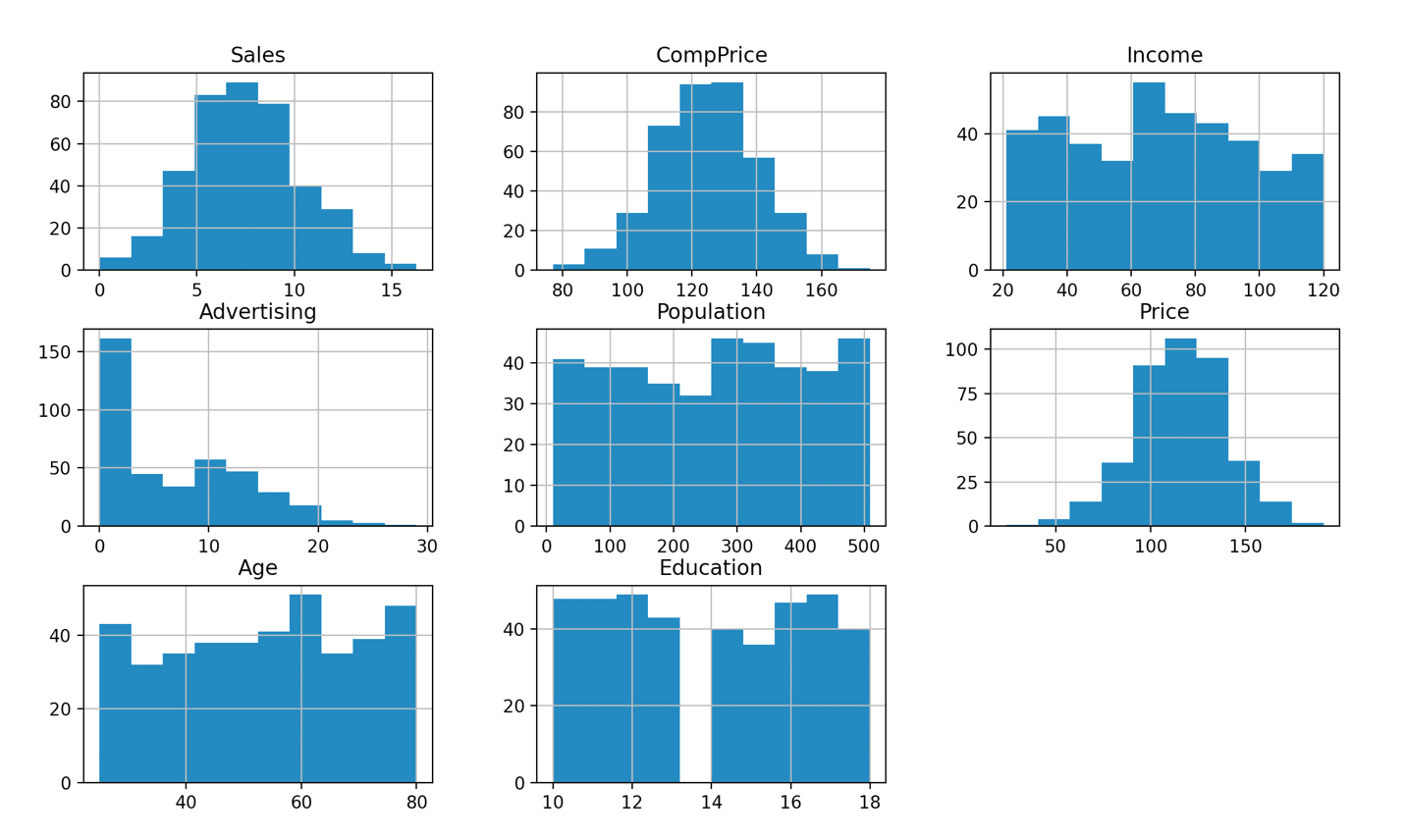
**Report:**

In Part 1, we analyze the Carseats dataset based on Decision Tree model and its variants. The dataset contains 400 data with 10 attributes and 1 predictive result (*Sales*). The data set is:



**Figure 1. Carseats Data Set**

**Data Visualization**: We first visualize each column by a histogram:



**Figure 2. Histogram Visualization of Columns with Numerical Data**

**Chart, bar chart

Description automatically generatedChart, bar chart

Description automatically generatedChart, bar chart

Description automatically generated**

**Figure 3. Histogram Visualization of Columns with Categorical Data**

From Figure 2 we observed that, among the 7 numerical attributes, *CompPrice* and *Price* are Normally distributed, *Advertising* is skewed, other attributes tend to be uniformly distributed. Our target *Sales* is also normally distributed. For categorical data in Figure 3, *ShelveLoc* has more Medium than Good or Bad, and the data tends to fall in *Urban* and *US*.

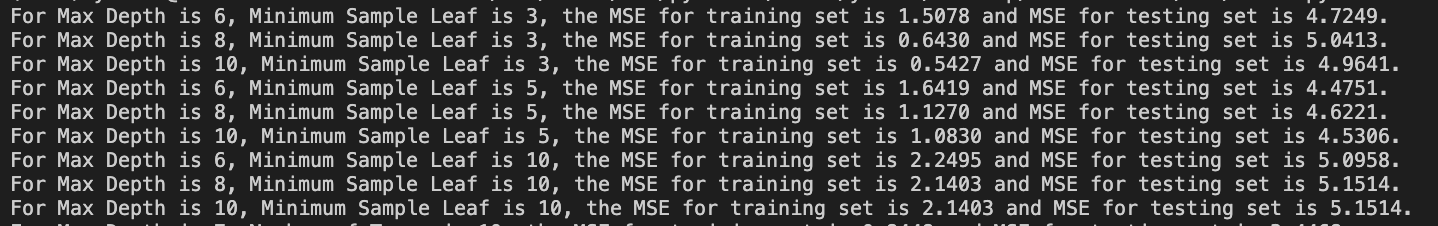
**Decision Tree**: We first apply Decision Tree to predict *Sales*, we iterate through **Least Node Size for 3, 5, 10**, and **Maximum Tree Depth for 6, 8, 10 to adjust the hyperparameters**, and built **9 trees in total**. The **first 300 data** in Carseats is used for training set and the **last 100 data** is used for testing set. For each tree built, we report the **Mean Squared Error (MSE)** for both the training and testing data sets and plot the resulting tree which is saved in PNG file in the same folder. Line 35 – 50 of “task1.py” are the codes adopted to realize the task:

Text

Description automatically generated

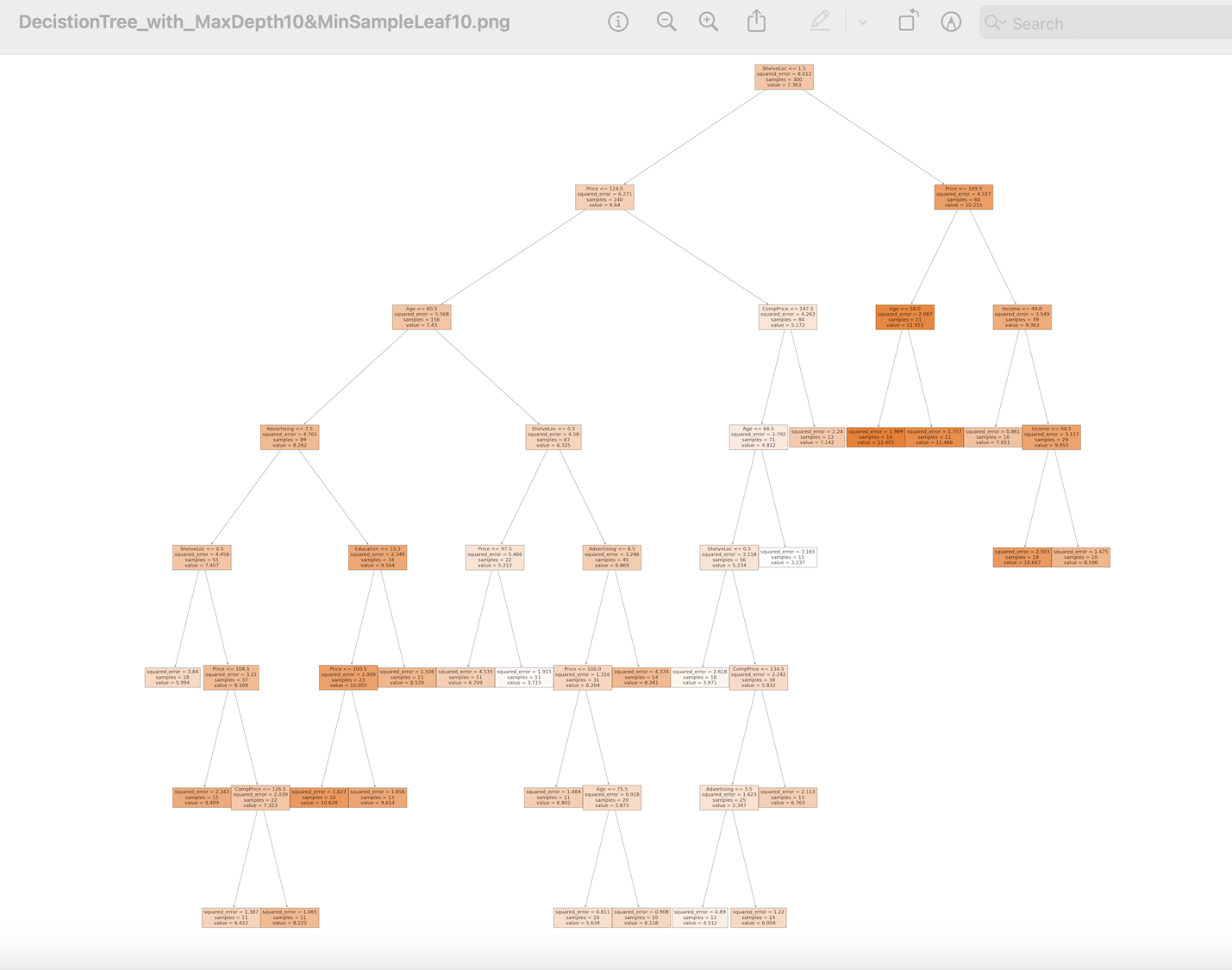
**Figure 4. Part of the Codes Used to Realize Decision Tree**

The output is:



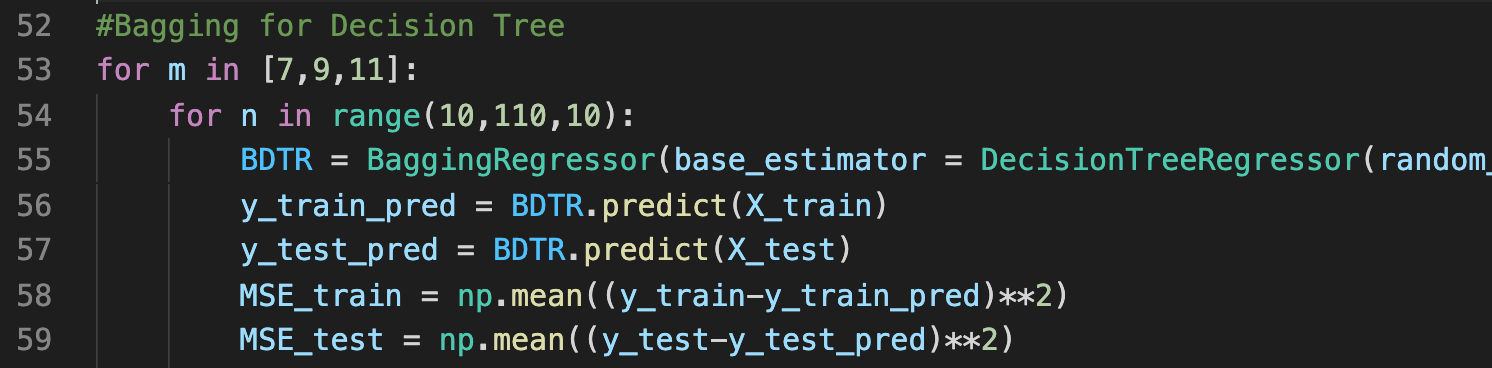
**Figure 5. Output for Decision Tree with Various Min\_Samples\_Leaf and Max\_Depth**

The overall MSE for testing sets tend to be **around 5**. The MSE for training set tends to decrease as Maximum Tree Depth increases, which gave a better fit for training set. The MSE for testing set remains the lowest for Least Node Size equals 5, and in such case, the error remains the lowest when Maximum Tree Depth equals 6, with a testing MSE 4.4751. One tree plot is shown as follows, all plots can be found in the same folder:



**Figure 6. Decision Tree (with Max\_Depth 10 and Min\_Samples\_Leaf 10) Plotting**

**Bagging for Decision Trees**: We now built the model using Bagging for Decision Trees, and iterate through a Maximum Tree Depth for 7, 9, 11, and Number or Trees for 10, 20, 30, … , 100 to adjust the hyperparameters, and get totally 30 models. We then report the MSE for training set and testing set for each model. Line 52 – 61 is used to realize the Bagging model:



**Figure 7. Part of the Codes Used to Realize Bagging for Decision Trees**

The output is:

Text

Description automatically generated

**Figure 8. Output for Training and Testing MSEs**

The overall MSE for testing sets tend to be **around 2.8**, which **improved by 2.2** from the Decision Tree model, thus we conclude that Bagging is a stronger model than the naïve Decision Tree. Apart from it, the MSE for training set tends to decrease when the Maximum Tree Depth is increased, which may lead to overfitting. Testing MSE decreases as the Number of Trees becomes larger, since more trees can reduce bias for the model.

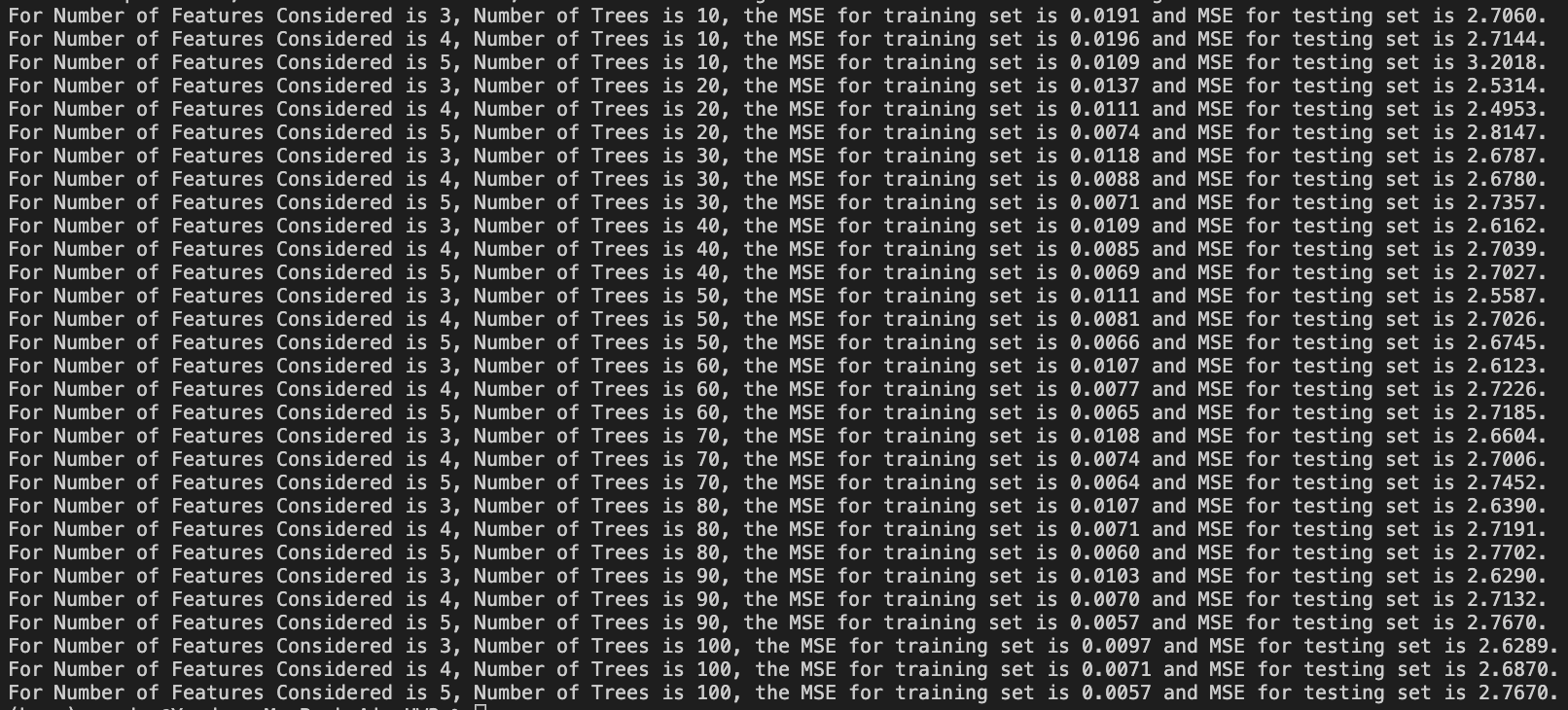
**Random Forest**: Lastly, we adopt Random Forest Regressor to predict *Sales*, and iterate through Number of Trees for 10, 20, … , 100, and Number of Features Considered in each split for 3 (approximately 10/3), 4, 5 to adjust the hyperparameters, resulting in 30 forests in total. We output the MSE for both training and testing sets. Line 63 - 73 is used to realize this part:

Text

Description automatically generated

**Figure 9. Part of the Codes Used to Realize Random Forest**

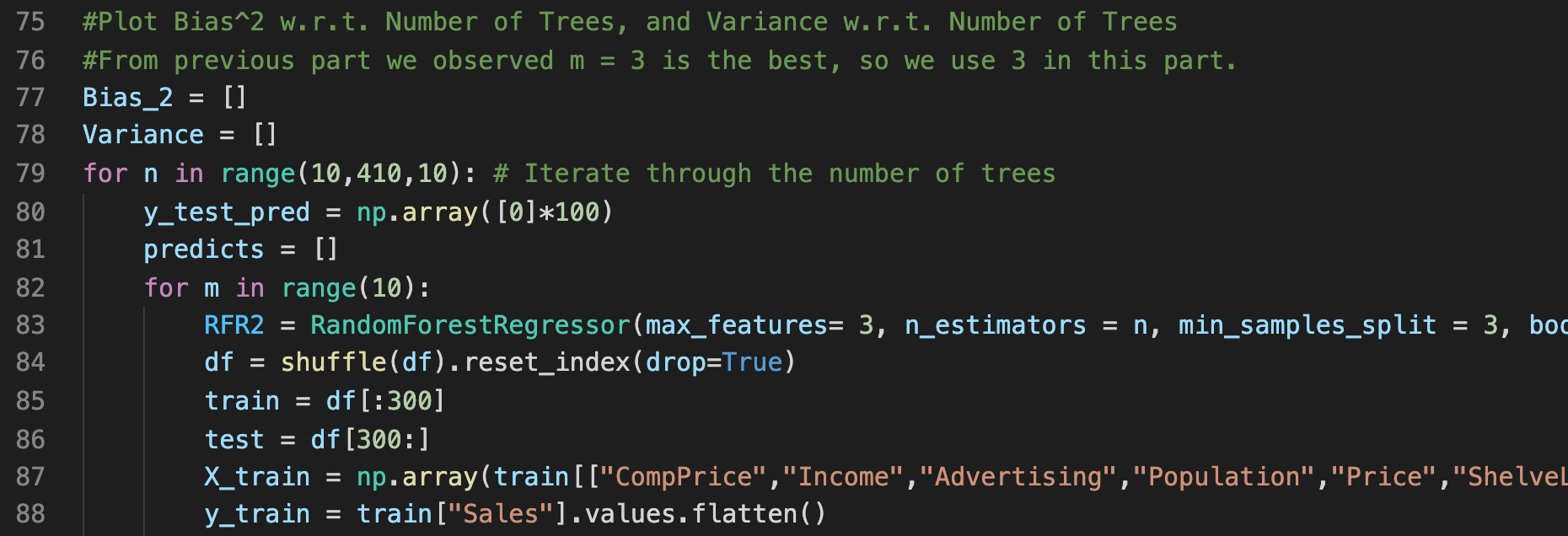
The output is:



**Figure 10. Output for Training and Testing MSEs in Random Forest Regressor**

The overall MSE for testing set is **around 2.7**, which **improved by 0.1** from Bagging for Decision Trees, thus Random Forest is slightly better than Bagging. The optimal Number of Features Considered is 3 in this case, since the testing MSEs tend to be the lowest among others. The training MSEs all stay low.

**Plotting Bias2 and Variance Versus Number of Trees in Random Forest:** We now want to study the relationship between Bias2 (Variance) and the number of trees in a Random Forest. From the previous part, we observe that m = 3 is the optimal number of features to be considered in each split. Hence, we set m = 3 in this part, and iterate through Number of Trees for 10, 20, …, 390, 400 in a forest, and under each fixed number of trees, we shuffle the data 10 times to process 10 different forests to calculate Bias2 and Variance. The codes are from Line 75 to the end:



**Figure 11. Part of the Codes Used to Calculate Bias2 and Variance for Different Number of Trees in a Random Forests**

The outputs are:

Chart, line chart, histogram

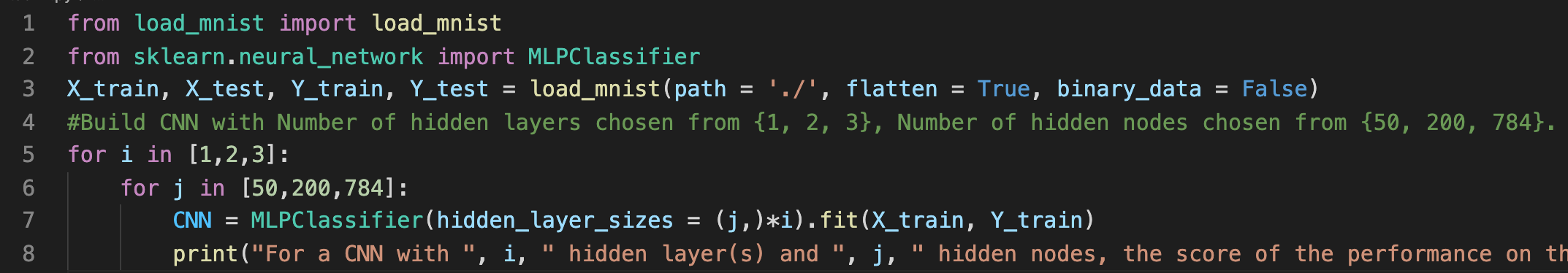
Description automatically generatedChart, line chart, histogram

Description automatically generated

**Figure 12. Bias2 and Variance vs Number of Trees in a Random Forest**

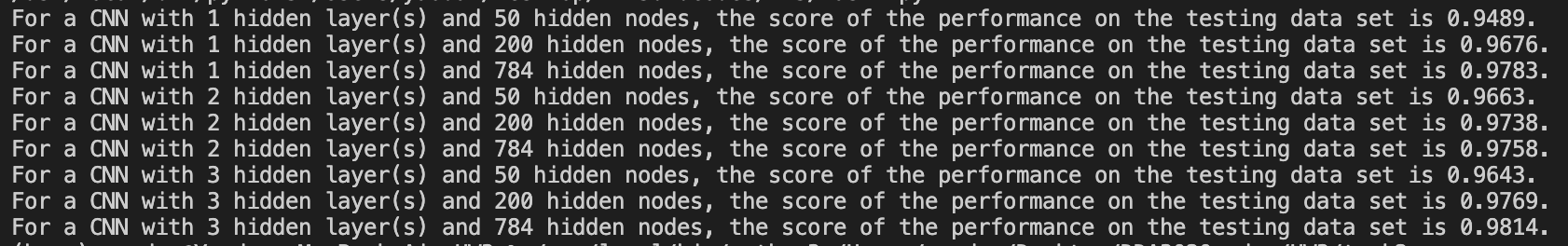
The Bias2 has no pattern when the number of trees varies, whereas the variance decreases as the number of trees increases, that is because when there are more trees in a forest, there are more similar trees, which makes the prediction results more stable.

In Part 2, we build a fully connected neural network to facilitate Handwritten Digit Recognition. The training and testing sets have been processed in the file “load\_mnist.py”. We adopt MLPClassifier of sk-learn for the neural network model, and iterate through number of hidden layers for 1, 2, 3, and number of hidden nodes for 50, 200, 784 to adjust for hyperparameters. Line 1 – 8 is used to realize the task:



**Figure 13. Codes Used to Realize Neural Network for Hand Written Digit Recognition**

The output contains the scores for training and testing sets in each convolutional neural network:



**Figure 14. Output Scores for Each CNN**

The neural network with 784 hidden nodes and 3 hidden layers achieves the highest score, 0.9814. This is expectable since the model is the most complex among all.