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:

Carseats

Sales	CompPrice	Income	Advertising	Population	Price	ShelveLoc	Age	Education	Urban	US
9.5	138	73	11	276	120	Bad	42	17	Yes	Yes
11.22	111	48	16	260	83	Good	65	10	Yes	Yes
10.06	113	35	10	269	80	Medium	59	12	Yes	Yes
7.4	117	100	4	466	97	Medium	55	14	Yes	Yes
4.15	141	64	3	340	128	Bad	38	13	Yes	No
10.81	124	113	13	501	72	Bad	78	16	No	Yes

Figure 1. Carseats Data Set

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

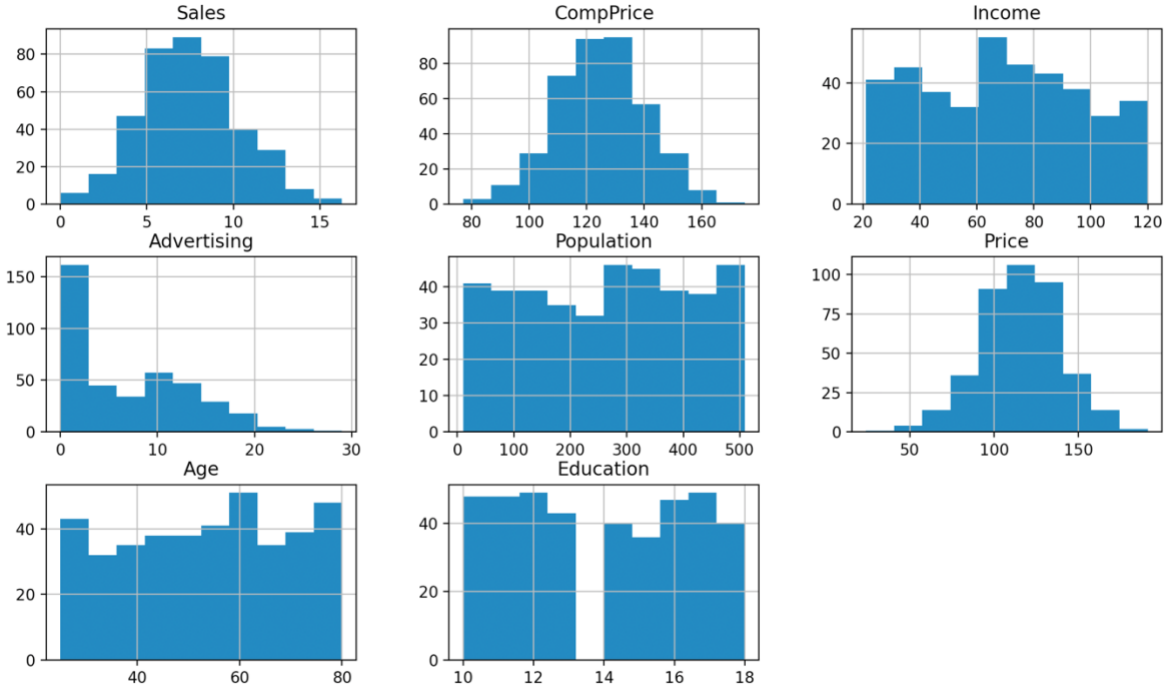

Figure 2. Histogram Visualization of Columns with Numerical Data

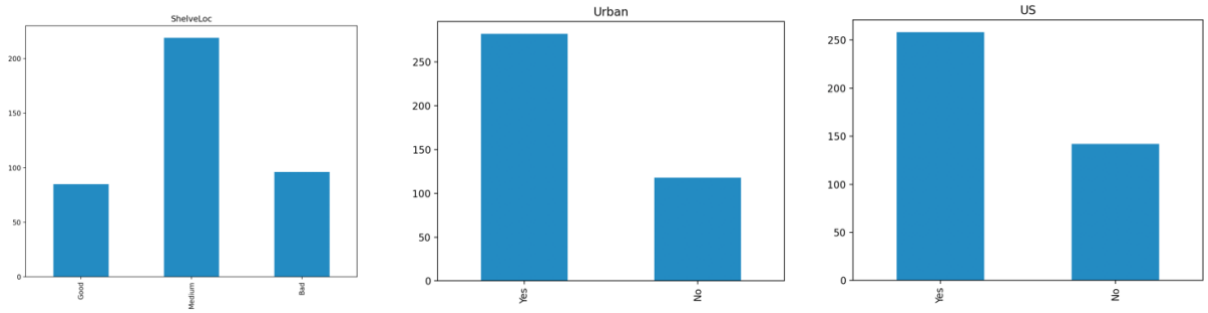

Figure 3. Histogram Visualization of Columns with Categorical Data

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:

```
35  #Decision Tree Regressor
36  #Repeat building the Tree for least node sizes = 3,5,10, maximum depths = 6,8,10.
37  for i in [3,5,10]:
38      for j in [6,8,10]:
39          DTR = DecisionTreeRegressor(random_state=0, max_depth=j, min_samples_leaf=i)
40          DTR.fit(X_train, y_train)
41          y_train_pred = DTR.predict(X_train)
42          y_test_pred = DTR.predict(X_test)
43          MSE_train = np.mean((y_train-y_train_pred)**2)
44          MSE_test = np.mean((y_test-y_test_pred)**2)
```

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

The output is:

```
For Max Depth is 6, Minimum Sample Leaf is 3, the MSE for training set is 1.5078 and MSE for testing set is 4.7249.
For Max Depth is 8, Minimum Sample Leaf is 3, the MSE for training set is 0.6430 and MSE for testing set is 5.0413.
For Max Depth is 10, Minimum Sample Leaf is 3, the MSE for training set is 0.5427 and MSE for testing set is 4.9641.
For Max Depth is 6, Minimum Sample Leaf is 5, the MSE for training set is 1.6419 and MSE for testing set is 4.4751.
For Max Depth is 8, Minimum Sample Leaf is 5, the MSE for training set is 1.1270 and MSE for testing set is 4.6221.
For Max Depth is 10, Minimum Sample Leaf is 5, the MSE for training set is 1.0830 and MSE for testing set is 4.5306.
For Max Depth is 6, Minimum Sample Leaf is 10, the MSE for training set is 2.2495 and MSE for testing set is 5.0958.
For Max Depth is 8, Minimum Sample Leaf is 10, the MSE for training set is 2.1403 and MSE for testing set is 5.1514.
For Max Depth is 10, Minimum Sample Leaf is 10, the MSE for training set is 2.1403 and MSE for testing set is 5.1514.
```

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:

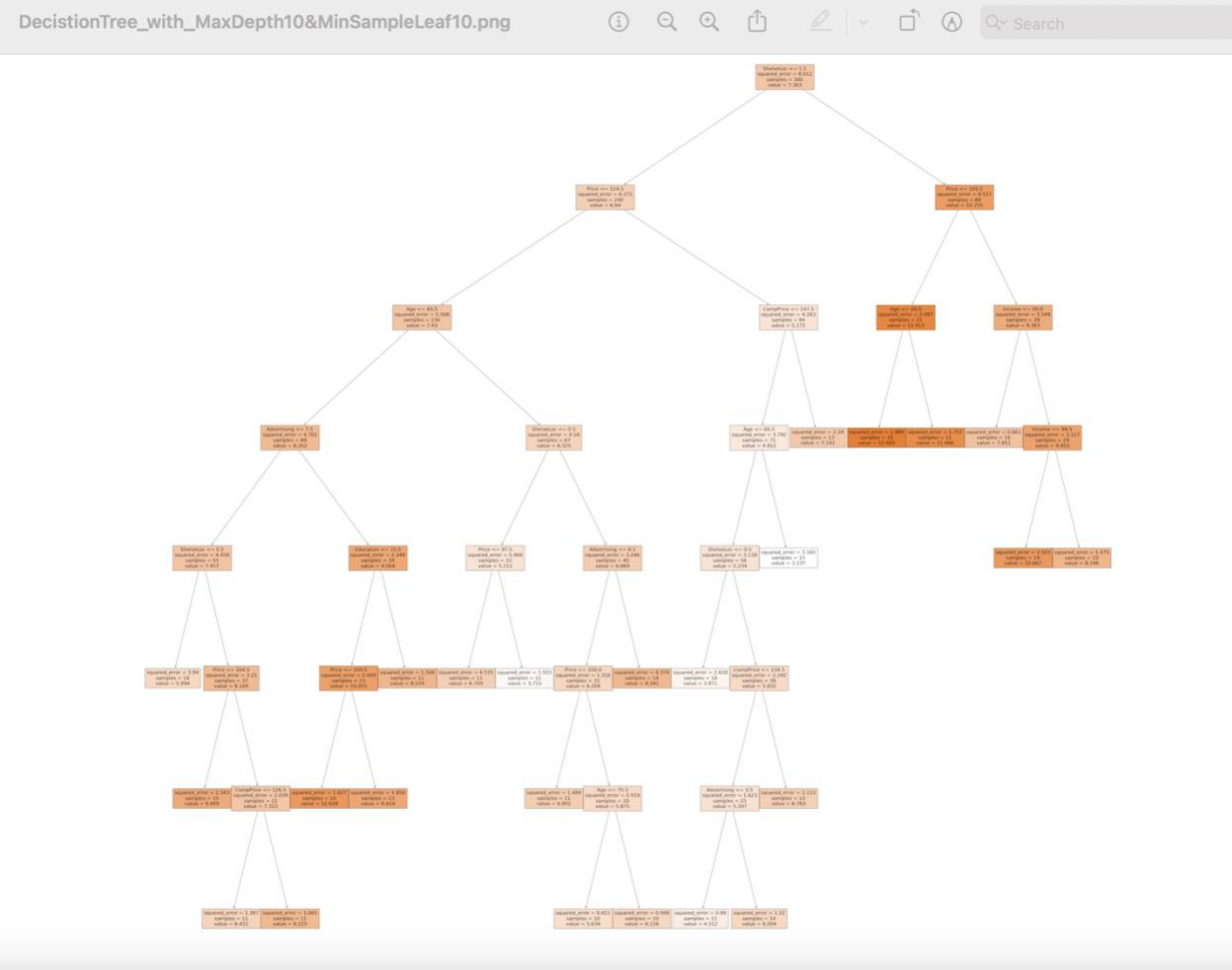

Figure 6. Decision Tree (with Max_Depth 10 and Min_Samples_Leaf 10) Plotting

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:

```
52  #Bagging for Decision Tree
53  for m in [7,9,11]:
54      for n in range(10,110,10):
55          BDTR = BaggingRegressor(base_estimator = DecisionTreeRegressor(random_
56          y_train_pred = BDTR.predict(X_train)
57          y_test_pred = BDTR.predict(X_test)
58          MSE_train = np.mean((y_train-y_train_pred)**2)
59          MSE_test = np.mean((y_test-y_test_pred)**2)
```

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

The output is:

```
For Max Depth is 7, Number of Trees is 10, the MSE for training set is 0.8448 and MSE for testing set is 3.4468.
For Max Depth is 7, Number of Trees is 20, the MSE for training set is 0.7209 and MSE for testing set is 3.2249.
For Max Depth is 7, Number of Trees is 30, the MSE for training set is 0.6822 and MSE for testing set is 2.9927.
For Max Depth is 7, Number of Trees is 40, the MSE for training set is 0.6553 and MSE for testing set is 2.8894.
For Max Depth is 7, Number of Trees is 50, the MSE for training set is 0.6514 and MSE for testing set is 2.9356.
For Max Depth is 7, Number of Trees is 60, the MSE for training set is 0.6576 and MSE for testing set is 2.8742.
For Max Depth is 7, Number of Trees is 70, the MSE for training set is 0.6505 and MSE for testing set is 2.8602.
For Max Depth is 7, Number of Trees is 80, the MSE for training set is 0.6515 and MSE for testing set is 2.8337.
For Max Depth is 7, Number of Trees is 90, the MSE for training set is 0.6636 and MSE for testing set is 2.8170.
For Max Depth is 7, Number of Trees is 100, the MSE for training set is 0.6705 and MSE for testing set is 2.8177.
For Max Depth is 9, Number of Trees is 10, the MSE for training set is 0.5990 and MSE for testing set is 3.2518.
For Max Depth is 9, Number of Trees is 20, the MSE for training set is 0.4711 and MSE for testing set is 3.1198.
For Max Depth is 9, Number of Trees is 30, the MSE for training set is 0.4331 and MSE for testing set is 2.9595.
For Max Depth is 9, Number of Trees is 40, the MSE for training set is 0.4152 and MSE for testing set is 2.8615.
For Max Depth is 9, Number of Trees is 50, the MSE for training set is 0.4129 and MSE for testing set is 2.8858.
For Max Depth is 9, Number of Trees is 60, the MSE for training set is 0.4159 and MSE for testing set is 2.8501.
For Max Depth is 9, Number of Trees is 70, the MSE for training set is 0.4102 and MSE for testing set is 2.8223.
For Max Depth is 9, Number of Trees is 80, the MSE for training set is 0.4113 and MSE for testing set is 2.8120.
For Max Depth is 9, Number of Trees is 90, the MSE for training set is 0.4255 and MSE for testing set is 2.7802.
For Max Depth is 9, Number of Trees is 100, the MSE for training set is 0.4267 and MSE for testing set is 2.7887.
For Max Depth is 11, Number of Trees is 10, the MSE for training set is 0.5388 and MSE for testing set is 3.1229.
For Max Depth is 11, Number of Trees is 20, the MSE for training set is 0.4167 and MSE for testing set is 3.1261.
For Max Depth is 11, Number of Trees is 30, the MSE for training set is 0.3795 and MSE for testing set is 2.9308.
For Max Depth is 11, Number of Trees is 40, the MSE for training set is 0.3562 and MSE for testing set is 2.8614.
For Max Depth is 11, Number of Trees is 50, the MSE for training set is 0.3509 and MSE for testing set is 2.8644.
For Max Depth is 11, Number of Trees is 60, the MSE for training set is 0.3539 and MSE for testing set is 2.8197.
For Max Depth is 11, Number of Trees is 70, the MSE for training set is 0.3457 and MSE for testing set is 2.8111.
For Max Depth is 11, Number of Trees is 80, the MSE for training set is 0.3474 and MSE for testing set is 2.7955.
For Max Depth is 11, Number of Trees is 90, the MSE for training set is 0.3577 and MSE for testing set is 2.7548.
For Max Depth is 11, Number of Trees is 100, the MSE for training set is 0.3563 and MSE for testing set is 2.7750.
```

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:

```
63  #Random Forests Regressor
64  for n in range(10,110,10):
65      for m in [3,4,5]:
66          RFR = RandomForestRegressor(max_features= m, n_estimators = n, min_samples
67          RFR.fit(X_train, y_train)
68          y_train_pred = RFR.predict(X_train)
69          y_test_pred = RFR.predict(X_test)
70          MSE_train = np.mean((y_train-y_train_pred)**2)
71          MSE_test = np.mean((y_test-y_test_pred)**2)
```

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

The output is:

```
For Number of Features Considered is 3, Number of Trees is 10, the MSE for training set is 0.0191 and MSE for testing set is 2.7060.
For Number of Features Considered is 4, Number of Trees is 10, the MSE for training set is 0.0196 and MSE for testing set is 2.7144.
For Number of Features Considered is 5, Number of Trees is 10, the MSE for training set is 0.0109 and MSE for testing set is 3.2018.
For Number of Features Considered is 3, Number of Trees is 20, the MSE for training set is 0.0137 and MSE for testing set is 2.5314.
For Number of Features Considered is 4, Number of Trees is 20, the MSE for training set is 0.0111 and MSE for testing set is 2.4953.
For Number of Features Considered is 5, Number of Trees is 20, the MSE for training set is 0.0074 and MSE for testing set is 2.8147.
For Number of Features Considered is 3, Number of Trees is 30, the MSE for training set is 0.0118 and MSE for testing set is 2.6787.
For Number of Features Considered is 4, Number of Trees is 30, the MSE for training set is 0.0088 and MSE for testing set is 2.6780.
For Number of Features Considered is 5, Number of Trees is 30, the MSE for training set is 0.0071 and MSE for testing set is 2.7357.
For Number of Features Considered is 3, Number of Trees is 40, the MSE for training set is 0.0109 and MSE for testing set is 2.6162.
For Number of Features Considered is 4, Number of Trees is 40, the MSE for training set is 0.0085 and MSE for testing set is 2.7039.
For Number of Features Considered is 5, Number of Trees is 40, the MSE for training set is 0.0069 and MSE for testing set is 2.7027.
For Number of Features Considered is 3, Number of Trees is 50, the MSE for training set is 0.0111 and MSE for testing set is 2.5587.
For Number of Features Considered is 4, Number of Trees is 50, the MSE for training set is 0.0081 and MSE for testing set is 2.7026.
For Number of Features Considered is 5, Number of Trees is 50, the MSE for training set is 0.0066 and MSE for testing set is 2.6745.
For Number of Features Considered is 3, Number of Trees is 60, the MSE for training set is 0.0107 and MSE for testing set is 2.6123.
For Number of Features Considered is 4, Number of Trees is 60, the MSE for training set is 0.0077 and MSE for testing set is 2.7226.
For Number of Features Considered is 5, Number of Trees is 60, the MSE for training set is 0.0065 and MSE for testing set is 2.7185.
For Number of Features Considered is 3, Number of Trees is 70, the MSE for training set is 0.0108 and MSE for testing set is 2.6604.
For Number of Features Considered is 4, Number of Trees is 70, the MSE for training set is 0.0074 and MSE for testing set is 2.7006.
For Number of Features Considered is 5, Number of Trees is 70, the MSE for training set is 0.0064 and MSE for testing set is 2.7452.
For Number of Features Considered is 3, Number of Trees is 80, the MSE for training set is 0.0107 and MSE for testing set is 2.6390.
For Number of Features Considered is 4, Number of Trees is 80, the MSE for training set is 0.0071 and MSE for testing set is 2.7191.
For Number of Features Considered is 5, Number of Trees is 80, the MSE for training set is 0.0060 and MSE for testing set is 2.7702.
For Number of Features Considered is 3, Number of Trees is 90, the MSE for training set is 0.0103 and MSE for testing set is 2.6290.
For Number of Features Considered is 4, Number of Trees is 90, the MSE for training set is 0.0070 and MSE for testing set is 2.7132.
For Number of Features Considered is 5, Number of Trees is 90, the MSE for training set is 0.0057 and MSE for testing set is 2.7670.
For Number of Features Considered is 3, Number of Trees is 100, the MSE for training set is 0.0097 and MSE for testing set is 2.6289.
For Number of Features Considered is 4, Number of Trees is 100, the MSE for training set is 0.0071 and MSE for testing set is 2.6870.
For Number of Features Considered is 5, Number of Trees is 100, the MSE for training set is 0.0057 and MSE for testing set is 2.7670.
```

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 Bias^2 and Variance Versus Number of Trees in Random Forest: We now want to study the relationship between Bias^2 (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 Bias^2 and Variance. The codes are from Line 75 to the end:

```
75  #Plot Bias^2 w.r.t. Number of Trees, and Variance w.r.t. Number of Trees
76  #From previous part we observed m = 3 is the best, so we use 3 in this part.
77  Bias_2 = []
78  Variance = []
79  for n in range(10,410,10): # Iterate through the number of trees
80      y_test_pred = np.array([0]*100)
81      predicts = []
82      for m in range(10):
83          RFR2 = RandomForestRegressor(max_features= 3, n_estimators = n, min_samples_split = 3, boo
84          df = shuffle(df).reset_index(drop=True)
85          train = df[:300]
86          test = df[300:]
87          X_train = np.array(train[["CompPrice","Income","Advertising","Population","Price","ShelveL
88          y_train = train["Sales"].values.flatten()
```

Figure 11. Part of the Codes Used to Calculate Bias^2 and Variance for Different Number of Trees in a Random Forests

The outputs are:

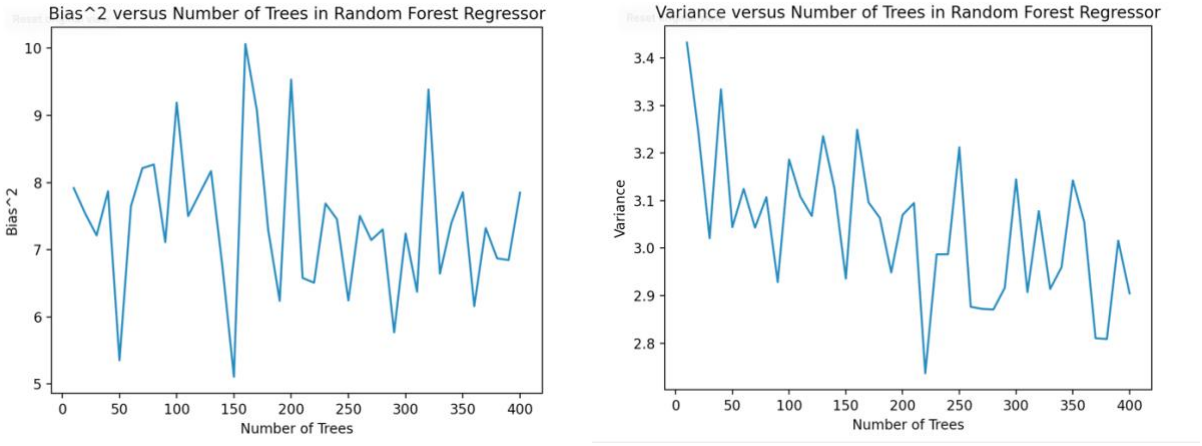

Figure 12. Bias^2 and Variance vs Number of Trees in a Random Forest

The Bias^2 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:

```
1  from load_mnist import load_mnist
2  from sklearn.neural_network import MLPClassifier
3  X_train, X_test, Y_train, Y_test = load_mnist(path = './', flatten = True, binary_data = False)
4  #Build CNN with Number of hidden layers chosen from {1, 2, 3}, Number of hidden nodes chosen from {50, 200, 784}.
5  for i in [1,2,3]:
6      for j in [50,200,784]:
7          CNN = MLPClassifier(hidden_layer_sizes = (j,)*i).fit(X_train, Y_train)
8          print("For a CNN with ", i, " hidden layer(s) and ", j, " hidden nodes, the score of the performance on th
```

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:

```
For a CNN with 1 hidden layer(s) and 50 hidden nodes, the score of the performance on the testing data set is 0.9489.
For a CNN with 1 hidden layer(s) and 200 hidden nodes, the score of the performance on the testing data set is 0.9676.
For a CNN with 1 hidden layer(s) and 784 hidden nodes, the score of the performance on the testing data set is 0.9783.
For a CNN with 2 hidden layer(s) and 50 hidden nodes, the score of the performance on the testing data set is 0.9663.
For a CNN with 2 hidden layer(s) and 200 hidden nodes, the score of the performance on the testing data set is 0.9738.
For a CNN with 2 hidden layer(s) and 784 hidden nodes, the score of the performance on the testing data set is 0.9758.
For a CNN with 3 hidden layer(s) and 50 hidden nodes, the score of the performance on the testing data set is 0.9643.
For a CNN with 3 hidden layer(s) and 200 hidden nodes, the score of the performance on the testing data set is 0.9769.
For a CNN with 3 hidden layer(s) and 784 hidden nodes, the score of the performance on the testing data set is 0.9814.
```

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.