BIRZEIT UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

COMP4388: Machine learning

Assignment#3

Prepared by:

Salah AlDin Dar AlDeek 1192404

Instructor: **Radi Jarrar**

17-jan-2022

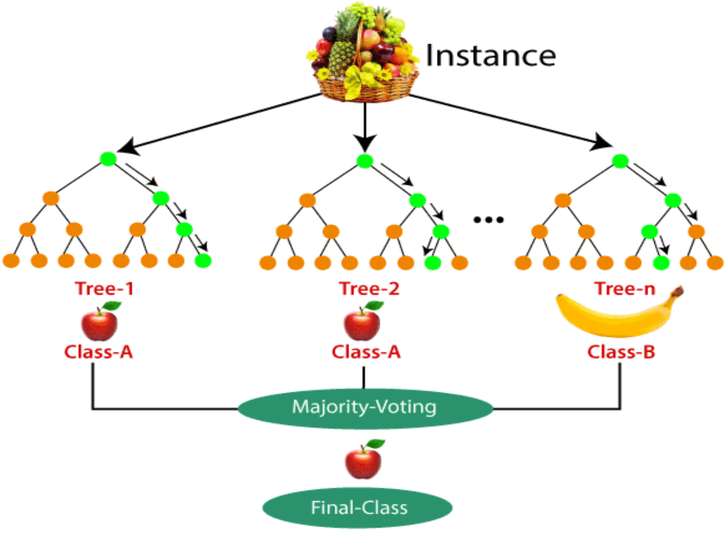
Question 1

Introduction

Ensemble is a machine learning algorithm to reduce overfitting and deal with missing data in which multiple models are trained using the same learning algorithm. Filling and reinforcement are two types of machine learning methods known as group learning, both methods build a set of models. and combine the predictions generated by those models to get better performance. Group learning is based on the principle that many predictors are better than one, and this is often the case.

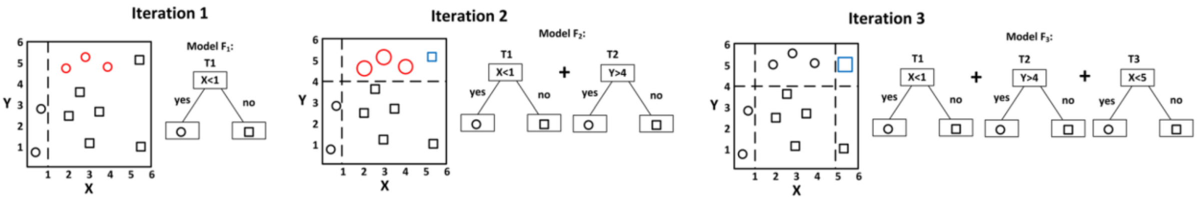
a.What is Random Forest?

Bagging is a method to decrease the variance(reduce the overfitting) in the prediction by generating a set of random samples, with replacement, from your training data.Then you train a model for each of these samples, which results in many models. Finally you integrate the expected results from all these models.If the target variable is categorical, you get a majority vote for the expected category, and that will be your prediction. If the target variable is numeric, you calculate the average of expectations from all models.A random forest is an example of a bagging algorithm. It consists of a collection of decision trees, thus it is referred to as a forest.
Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.



b.What is Boosting?

is an ensemble algorithm for primarily increasing accuracy, by dealing with missing data and minimising variance and bias in supervised learning, and a family of machine learning algorithms that convert weak learners to strong ones.
Boosting used to create a collection of predictors. In this technique, learners are learned sequentially with early learners fitting simple models to the data and then analysing data for errors. Consecutive trees (random sample) are fit and at every step, the goal is to improve the accuracy from the prior tree.



c.difference between Random Forests and XGBoost.

Random Forest: randomness comes from splitting the dataset into many sub datasets randomly, then we are going to build many decision trees for those sub datasets,the when we need to predict an input, we get the results from this sub datasets and finally return the average of the results or the most frequent one in classification tasks.
XGBoosting: we have a lot of decision trees sequentially,every tree depends on the previous tree,and the next tree is going to boost the attributes that led to miss classifications from the previous tree, so we have multiple trees are just build on top of each other to correct the errors of the previous tree.

- Random Forest is an example of baggining, and XGBoost is an example of Boosting.
- Splitting data in random forest is random, while in xgboost it has a higher vote to misclassified samples.
- The method used in the random forest is a random subspace and The method used in XGBoost is gradient boosting
- We combine models in Random forest by calculating the average, but in XGBoosting by Weighting majority vote.
- Random forest can be run in parallel,but XGBoosting runs in serial.

Question 2

Introduction

I use the decision tree library that exists in Scikit, and use its modified version of CART Algorithm, and it has a very similar performance to C4.5.

C4.5 is an algorithm used to generate a decision tree developed by Ross Quinlan,and it is an extension of ID3 algorithm.The decision trees generated by C4.5 can be used for classification, and for this reason, C4.5 is often referred to as a statistical classifier.C4.5 uses information entropy to generate the tree, for every iteration algorithm calculate the gain ratio for every class label to set the class label have highest entropy to next node of tree.

Improvements of C4.5

- Handling both continuous and discrete attributes
- Handling training data with missing attribute values
- Handling attributes with differing costs.
- Pruning trees after creation

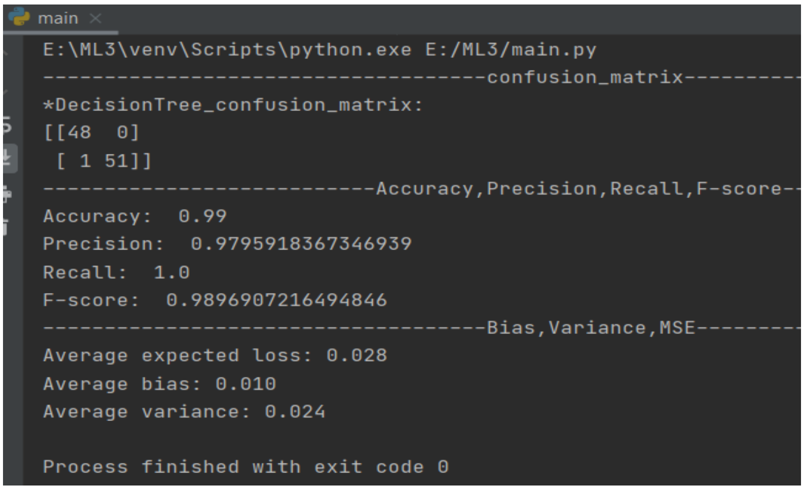
Explain code

First I read the data and drop the ID feature, because its dummy feature has the sequence number, then split the data set into x data set(input features), and **y** data set(target class).

Splitting the data 80:20 training:test , to calculate the accuracy measures of the model.

Then we generate the model by the training data set,and predict the target class of test data set and compare the predicted values with the real values to generate the confusion matrix to calculate Accuracy, Precision, Recall and F-score .

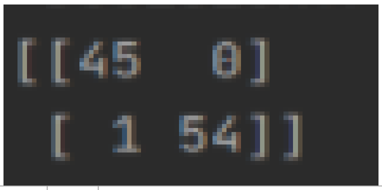
I use the bias_variance_decomp from mlxtend.evaluate library to calculate the Bias and variance.



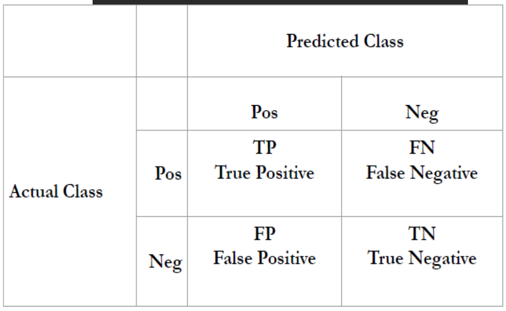
```
main
E:\ML3\venv\Scripts\python.exe E:/ML3/main.py
------------------------------------confusion_matrix---------
*DecisionTree_confusion_matrix:
[[48  0]
 [ 1 51]]
---------------------------Accuracy,Precision,Recall,F-score--
Accuracy:  0.99
Precision:  0.9795918367346939
Recall:  1.0
F-score:  0.9896907216494846
------------------------------------Bias,Variance,MSE--------
Average expected loss: 0.028
Average bias: 0.010
Average variance: 0.024

Process finished with exit code 0
```

Accuracy Measures

As shown in the confusion matrix of the output:
TP=45, FP:0, FN:1, TN:54
Accuracy =(TP+TN)/All=99/100=0.99
Precision=TP/(TP+FP)=45/46=~0.978
Recall=TP/(TP+FN)=45/45=1
F-Score=2*TP/(2*TP+FP+FN))=90/(90+1)=~0.989



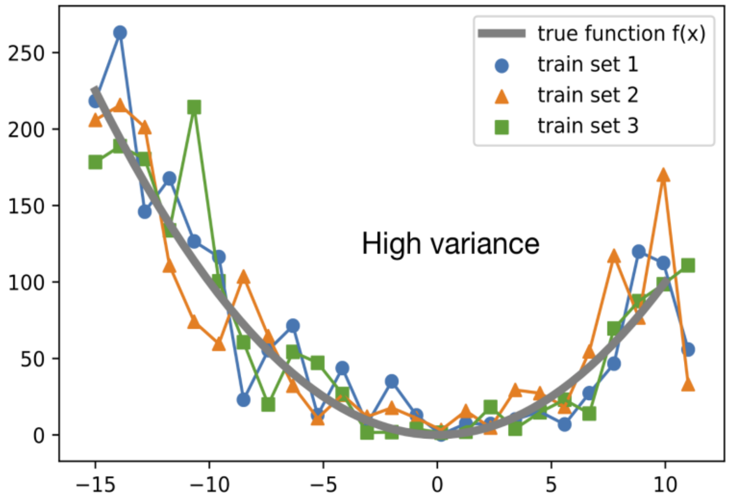
```
[[45  0]
 [ 1 54]]
```

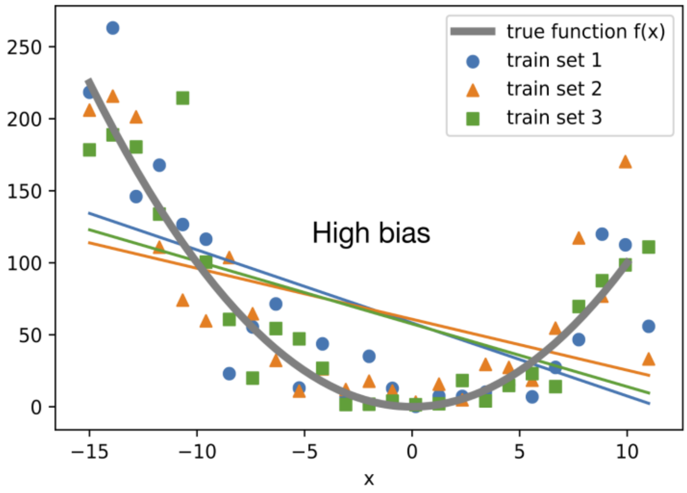


		Predicted Class	
		Pos	Neg
Actual Class	Pos	TP True Positive	FN False Negative
	Neg	FP False Positive	TN True Negative

Variance and Bias

The main idea of variance and bias is to check the balance between overfitting and underfitting by cross validation(split the data into subsets).





θ^ :point estimator

θ :parameter or function

$$\mathrm{Bias} = E[\hat{\theta}] - \theta.$$

$$\mathrm{Var}(\hat{\theta}) = E[\hat{\theta}^2] - \left(E[\hat{\theta}]\right)^2$$

I use the plot_tree from sklearn.tree library to blot the tree of the model.

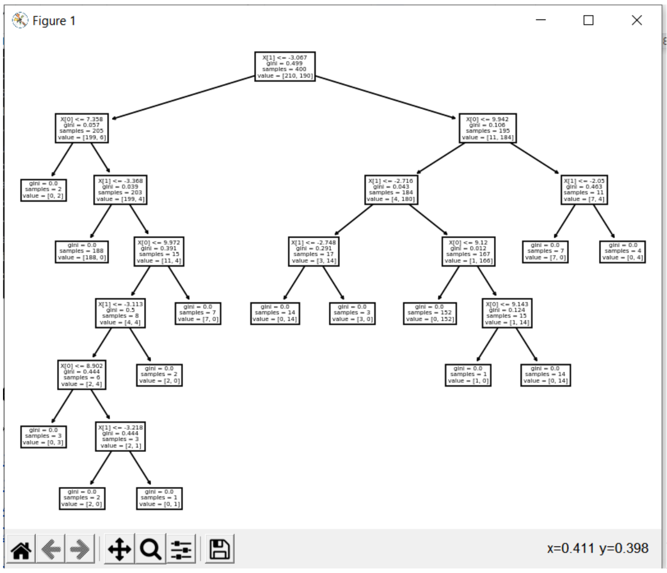


Figure 1
X[1] <= -3.067 gini = 0.499 samples = 400 value = [210, 190]
X[0] <= 7.358 gini = 0.057 samples = 205 value = [199, 6]
X[0] <= 9.942 gini = 0.106 samples = 195 value = [11, 184]
gini = 0.0 samples = 2 value = [0, 2]
X[1] <= -3.368 gini = 0.039 samples = 203 value = [199, 4]
X[1] <= -2.716 gini = 0.043 samples = 184 value = [4, 180]
X[1] <= -2.05 gini = 0.463 samples = 11 value = [7, 4]
gini = 0.0 samples = 188 value = [188, 0]
X[0] <= 9.972 gini = 0.391 samples = 15 value = [11, 4]
X[1] <= -2.748 gini = 0.291 samples = 17 value = [3, 14]
X[0] <= 9.12 gini = 0.012 samples = 167 value = [1, 166]
gini = 0.0 samples = 7 value = [7, 0]
gini = 0.0 samples = 4 value = [0, 4]
X[1] <= -3.113 gini = 0.5 samples = 8 value = [4, 4]
gini = 0.0 samples = 7 value = [7, 0]
gini = 0.0 samples = 14 value = [0, 14]
gini = 0.0 samples = 3 value = [3, 0]
gini = 0.0 samples = 152 value = [0, 152]
X[0] <= 9.143 gini = 0.124 samples = 15 value = [1, 14]
X[0] <= 8.902 gini = 0.444 samples = 6 value = [2, 4]
gini = 0.0 samples = 2 value = [2, 0]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 14 value = [0, 14]
gini = 0.0 samples = 3 value = [0, 3]
X[1] <= -3.218 gini = 0.444 samples = 3 value = [2, 1]
gini = 0.0 samples = 2 value = [2, 0]
gini = 0.0 samples = 1 value = [0, 1]
x=0.411 y=0.398

references and citations

https://analyticsindiamag.com/primer-ensemble-learning-bagging-boosting/
https://cnvrg.io/random-forest-regression/?gclid=CjwKCAiA0KmPBhBqEiwAJqKK48YERL62qp7CRIT5YRcSh0EVjB5BTj4L08GjxS0u6ZRryCvZO7nm6RoCT2EQAvD_BwE
https://www.youtube.com/watch?v=eTCehWQiLTs
https://en.wikipedia.org/wiki/C4.5_algorithm