Faculty of Engineering & Technology
Department of Computer Science

MACHINE LEARNING

Assignment 3

Bagging and Boosting

Prepared by:

Obada Tahayna 1191319

Instructor: Radi Jarrar

Section: 1

Date: 18/01/2022

Table of Contents

Contents

Table of Figures

Ensemble Learning Algorithms

The ensemble is a method used in the machine learning algorithm. In this method, multiple models or 'weak learners' are trained to rectify the same problem and integrated to gain desired results. Weak models combined rightly give accurate models. [1]

Bagging

Bagging, also known as Bootstrap Aggregation, is a parallel method used to decease the variance in the prediction model by separating the dataset randomly to make many decision trees and train them independently and simultaneously, the final decision established by making an average of N learners in regression or taking the voting rank done by most of them in classification. [1]

Random Forest

Random Forest is an extension of bagging that also randomly selects subsets of features used in each data sample. It used for regression and classification problems. Random Forest is more accurate than the decision tree and solves the issue of overfitting and increases precision. [2][3]

Besides using **Bootstrap** for splitting data, Random Forest uses **Feature Bagging** to split the dataset to multi set of data, this process make subsets of data by features randomly. The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the B trees, causing them to become correlated. [2][4][10]

In classification problems using Random Forest, the training data is fed to train various decision trees. This dataset consists of observations and features that will be selected randomly during the splitting of nodes (Bootstrap and Feature Bagging). In this case, the output chosen by the majority of the decision trees becomes the final output of the rain forest system. [2][3][4]

The figure below shows a sample that demonstrate how Random Forest Classifier works.

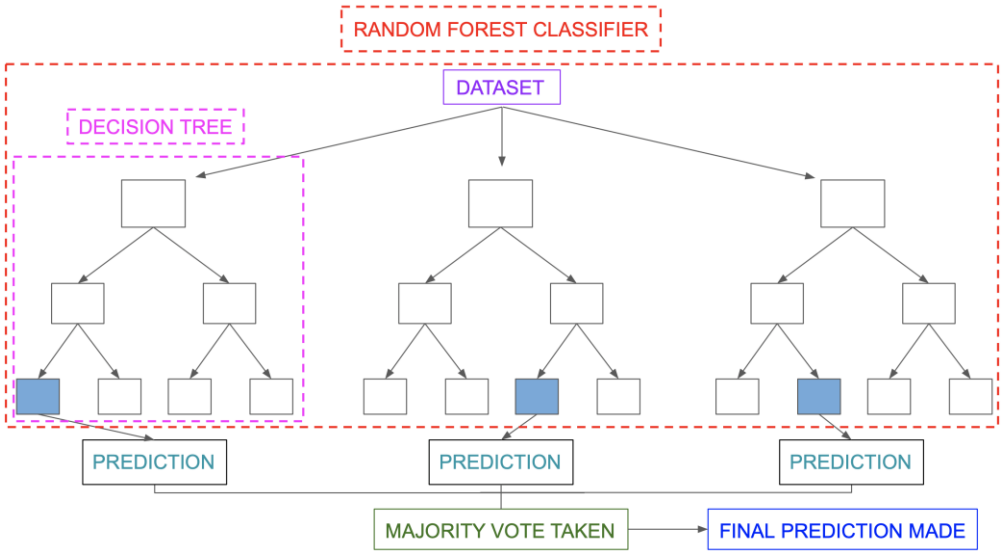

Figure 1: Sample of Random Forest Classifier [2]

Taking an example of training dataset of fruits classifier (example taken from section.io), The random forest classifier divides this dataset into subsets. These subsets are given to every decision tree in the random forest system. Each decision tree produces its specific output. For example, the prediction for trees 1 and 2 is apple and another decision tree (n) has predicted banana as the outcome. The random forest classifier collects the majority voting to provide the final prediction. The majority of the decision trees have chosen apple as their prediction. This makes the classifier choose apple as the final prediction. [2]

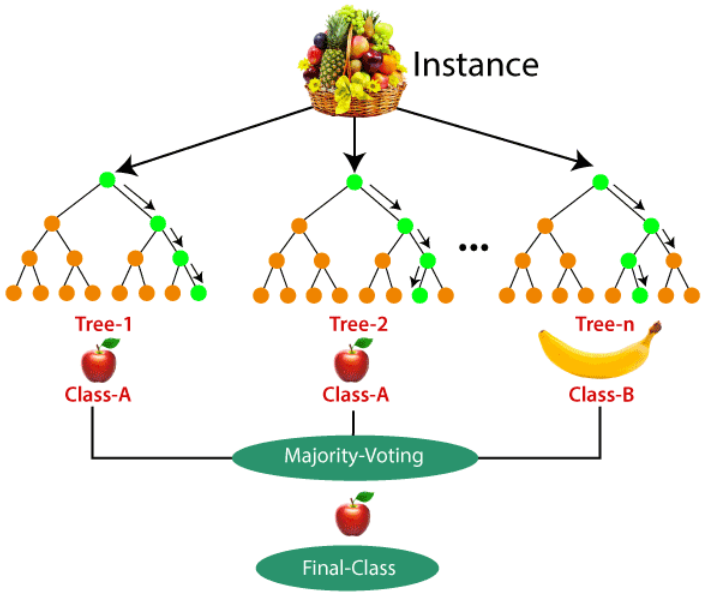

Figure 2: Example of Random Forest (Fruits Classifier) [2]

In regression models, Random Forest takes multiple decision trees, each tree predicts the output, then it calculates the average of all of the predictions to generate a great estimate of what the expected price for a real estate should be. [2][5][6]

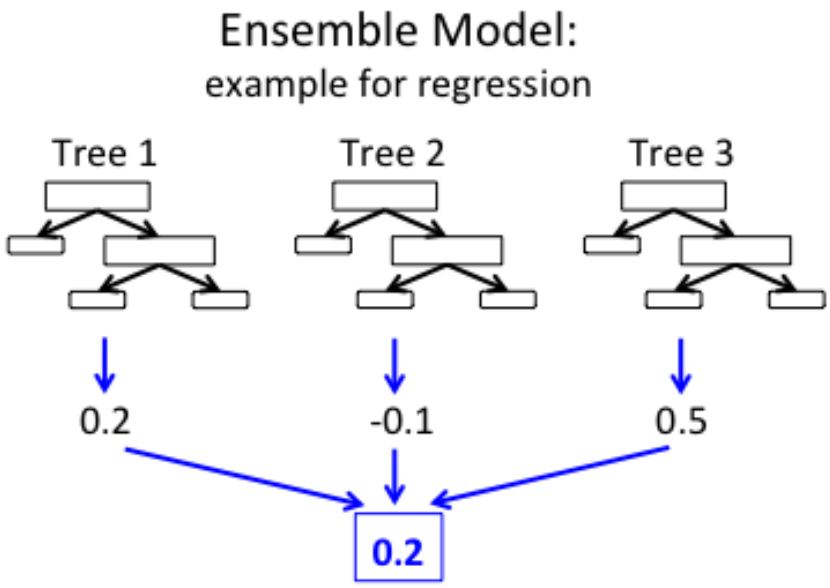

Figure 3: Regression Random Forest Example [7]

Boosting

Boosting is a sequential ensemble method that iteratively adjusts the weight of observation as per the last classification. If an observation is incorrectly classified, it increases the weight of that observation. Boosting used to build a strong and more accurate model, also it used to decrease the bias error. [1][8][11][19]

Boosting is a sequential algorithm that predicts the error in each training and make the new model to fit the errors in the previous model. [1][8][9]

Figure 4: Boosting Algorithm [9]

The process slowly learns from data and tries to improve its prediction in subsequence iteration. [1][11][12]

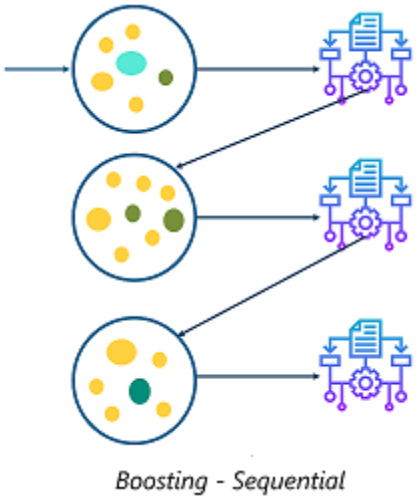

Figure 5: Boosting as Sequential Algorithm

There are many Boosting algorithms, the most two popular algorithms are Adaptive Boosting and Gradient Boosting. Adaptive Boosting automatically adjusts its parameters to the data based on the actual performance in the current iteration. Meaning, both the weights for re-weighting the data and the weights for the final aggregation are re-computed iteratively. While Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction, and it involves three elements: A loss function to be optimized, a weak learner to make predictions, an additive model to add weak learners to minimize the loss function. [1][8][12][14][15][18]

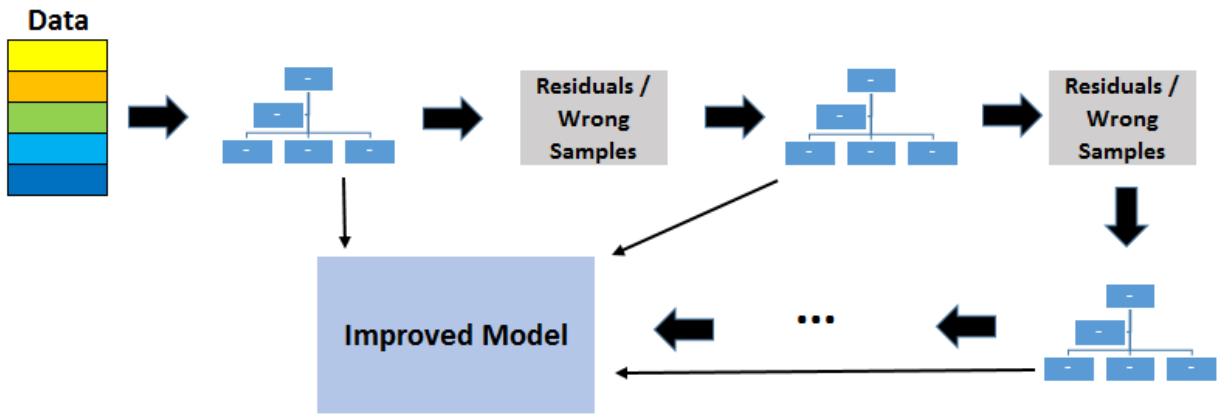

Figure 6: How Gradient Boosting Works [16]

XGBoost

XGBoost stands for e**X**treme **G**radient **B**oosting, it is an implementation of gradient boosted decision trees designed for speed and performance. The implementation of the algorithm was engineered for efficiency of compute time and memory resources. A design goal was to make the best use of available resources to train the model. Some key algorithm implementation features include:

- Sparse Aware implementation with automatic handling of missing data values.
- Block Structure to support the parallelization of tree construction.
- Continued Training so that you can further boost an already fitted model on new data. [17][18][20]

Differences between Random Forests and XGBoost.

- Random Forest Algorithm is based on Bagging Algorithm, which is a parallel method. XGBoost Algorithm is based on Boosting Algorithm which is a sequential method.
- Random Forest used mainly to decrease the variance, but XGBoost used basically to decrease the bias.
- While Random Forest is a Bagging Algorithm, then it solves the overfitting issue (that came from predict precisely in the training data but cannot predict with high accuracy in the testing data). XGBoost is a Boosting Algorithm then its suits with problem of underfitting (that came from inability to accurately predict).

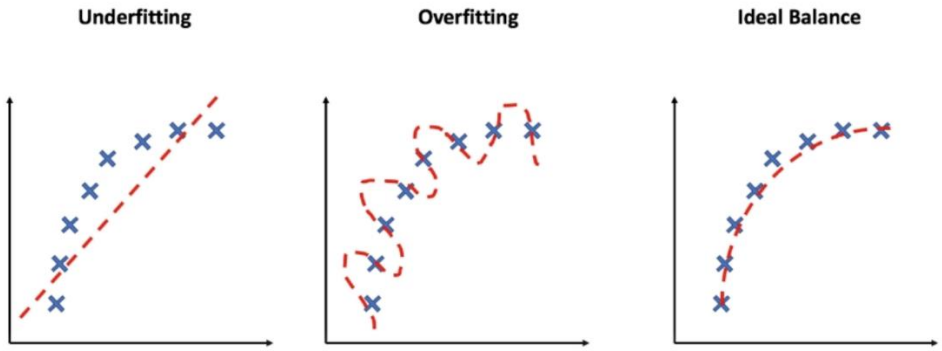

Figure 7: Overfitting and Underfitting [21]

- Random Forest is simpler than XGBoost.
- Random Forest has many trees with leaves of equal weight.

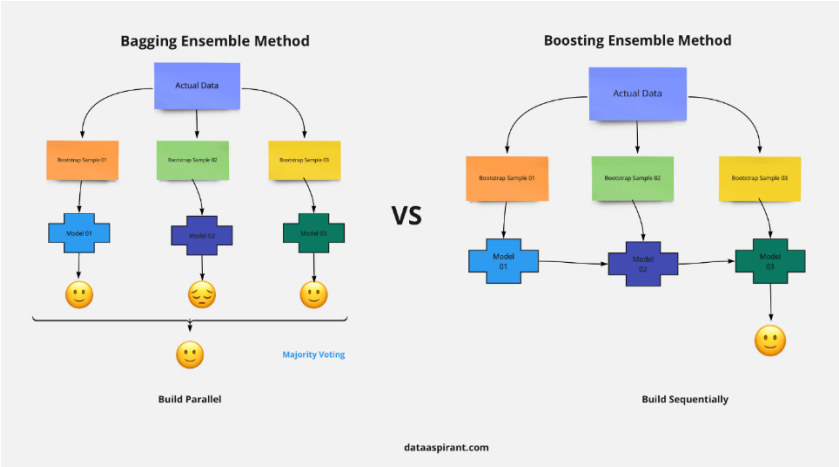

Figure 8: Bagging (Random Forest) VS Boosting (XGBoost) [22]

Part2 (Q2):

As it's clearly stated in the results, the random forest's model produced a bias that's not too low while also producing one with a low variance. These are the two characteristics that are expected from a good model with a decent fitting. However, the c4.5 decision tree didn't do as good of a job, as its bias was lower than the random forest's bias while the variance was higher, leading to the distribution being overfit on the training data, which means it performed extremely well on the training data, while not being able to predict the testing data properly.

Code

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from mlxtend.evaluate import bias_variance_decomp


def read_file(file_name):
    fileR = pd.read_csv(file_name)

    # Drop the first column (ID) as it's not required
    fileR = fileR.drop(['ID'], axis=1)

    return fileR


def reformat_data(data):
    labels = data['label'].values
    dataset = data.drop(['label'], axis=1)

    return labels, dataset


def split_data(data):
    # splitting data randomly to training and testing data
    train, test = train_test_split(data, test_size=0.2)
    return train, test


def c45_prediction(labels, dataset):
    clf = DecisionTreeClassifier()
    clf = clf.fit(dataset, labels)

    return clf


def rf_prediction(labels, dataset):
    clf = RandomForestClassifier()
    clf = clf.fit(dataset, labels)

    return clf
```

```
def main():
    file = read_file("Dataset.csv")

    training_data, testing_data = split_data(file)

    # label is the result column (to be predicted), dataset is the data
without the label column (data affects on the result)
    training_labels, training_dataset = reformat_data(training_data)
    testing_labels, testing_dataset = reformat_data(testing_data)

    # clf is the classification model
    c45_clf = c45_prediction(training_labels, training_dataset)
    rf_clf = rf_prediction(training_labels, training_dataset)

    # predicted labels for all testing data
    label_pred_c45 = c45_clf.predict(testing_dataset)
    label_pred_rf = rf_clf.predict(testing_dataset)

    # define reports that contain: precision, recall, f1-score, accuracy
    c45_report = classification_report(testing_labels, label_pred_c45)
    rf_report = classification_report(testing_labels, label_pred_rf)

    # finding mse, bias, var for C4.5 model and Random Forest model
    c45_mse, c45_bias, c45_var = bias_variance_decomp(c45_clf,
X_train=training_dataset.values, y_train=training_labels,

X_test=testing_dataset.values, y_test=testing_labels, loss='mse',
                                                      num_rounds=200,
random_seed=1)
    rf_mse, rf_bias, rf_var = bias_variance_decomp(rf_clf,
X_train=training_dataset.values, y_train=training_labels,

X_test=testing_dataset.values, y_test=testing_labels, loss='mse',
                                                    num_rounds=200,
random_seed=1)

    # printing all the results
    print("Results For Decision Tree C4.5 model: ")
    print(c45_report)
    print("MSE = " + str(c45_mse))
    print("Bias = " + str(c45_bias))
    print("Variance = " + str(c45_var))

    print("\n###############\n")

    print("Results For Random Forest model: ")
    print(rf_report)
    print("MSE = " + str(rf_mse))
    print("Bias = " + str(rf_bias))
    print("Variance = " + str(rf_var))


if __name__ == "__main__":
    main()
```

References

[1] https://www.upgrad.com/blog/bagging-vs-boosting/

[2] https://www.section.io/engineering-education/introduction-to-random-forest-in-machine-learning/

[3] https://machinelearningmastery.com/bagging-and-random-forest-for-imbalanced-classification/

[4] https://en.m.wikipedia.org/wiki/Random_forest

[5] https://neptune.ai/blog/random-forest-regression-when-does-it-fail-and-why

[6] https://www.keboola.com/blog/random-forest-regression

[7] https://databricks.com/blog/2015/01/21/random-forests-and-boosting-in-mllib.html

[8] https://en.wikipedia.org/wiki/Boosting_(machine_learning)

[9] https://towardsdatascience.com/what-is-boosting-in-machine-learning-2244aa196682

[10] https://www.youtube.com/watch?v=v6VJ2RO66Ag

[11] https://www.youtube.com/watch?v=sfVms30Ulxw

[12] https://www.youtube.com/watch?v=thR9ncsyMBE

[13] https://analyticsindiamag.com/adaboost-vs-gradient-boosting-a-comparison-of-leading-boosting-algorithms/

[14] https://en.wikipedia.org/wiki/Gradient_boosting

[15] https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

[16] https://towardsdatascience.com/gradient-boosted-decision-trees-explained-9259bd8205af

[17] https://en.wikipedia.org/wiki/XGBoost

[18] https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

[19] https://www.youtube.com/watch?v=wPqtzj5VZus

[20] https://www.youtube.com/watch?v=Vly8xGnNiWs

[21] https://medium.com/@rdhawan201455/overfitting-and-underfitting-bug-in-ml-models-97f2da56df30

[22] https://medium.com/geekculture/xgboost-versus-random-forest-898e42870f30