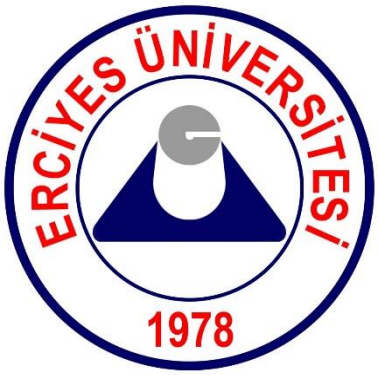


ERCIYES UNIVERSITY

FACULTY OF ENGINEERING

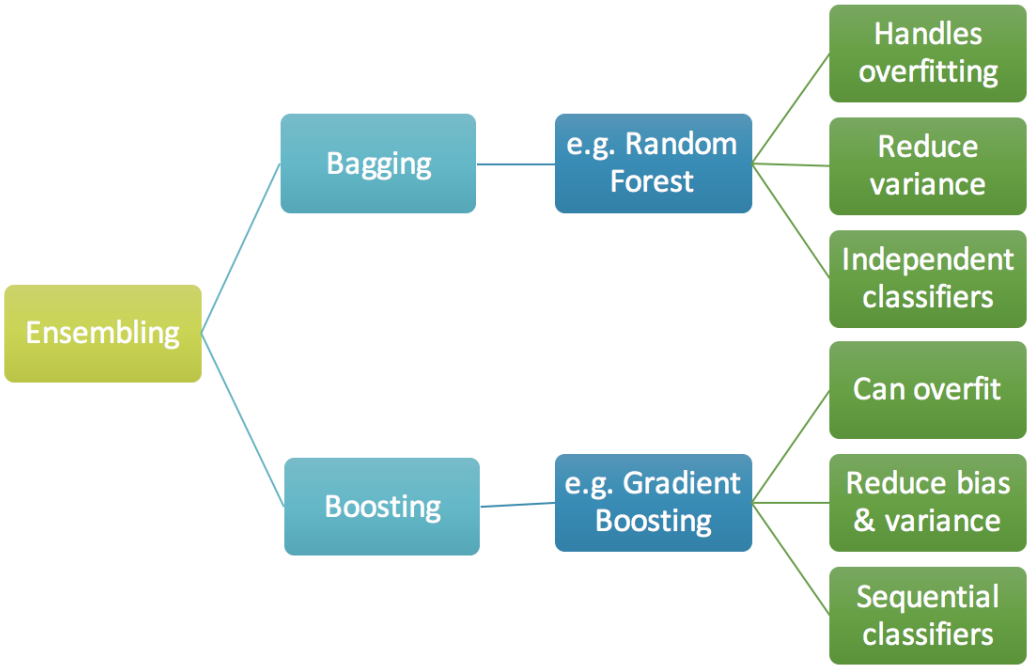
COMPUTER ENGINEERING DEPARTMENT

INTRODUCTION TO MACHINE LEARNING MIDTERM PROJECT REPORT

PREPARED BY:

ZEYNEP SERT 1030516727

CLASSIFICATION ALGORITHMS:



2- Gradient Boosted Trees Algorithm:

In gradient boosting, an ensemble of weak learners is used to improve the performance of a machine learning model. The weak learners are usually decision trees. Combined, their output results in better models.

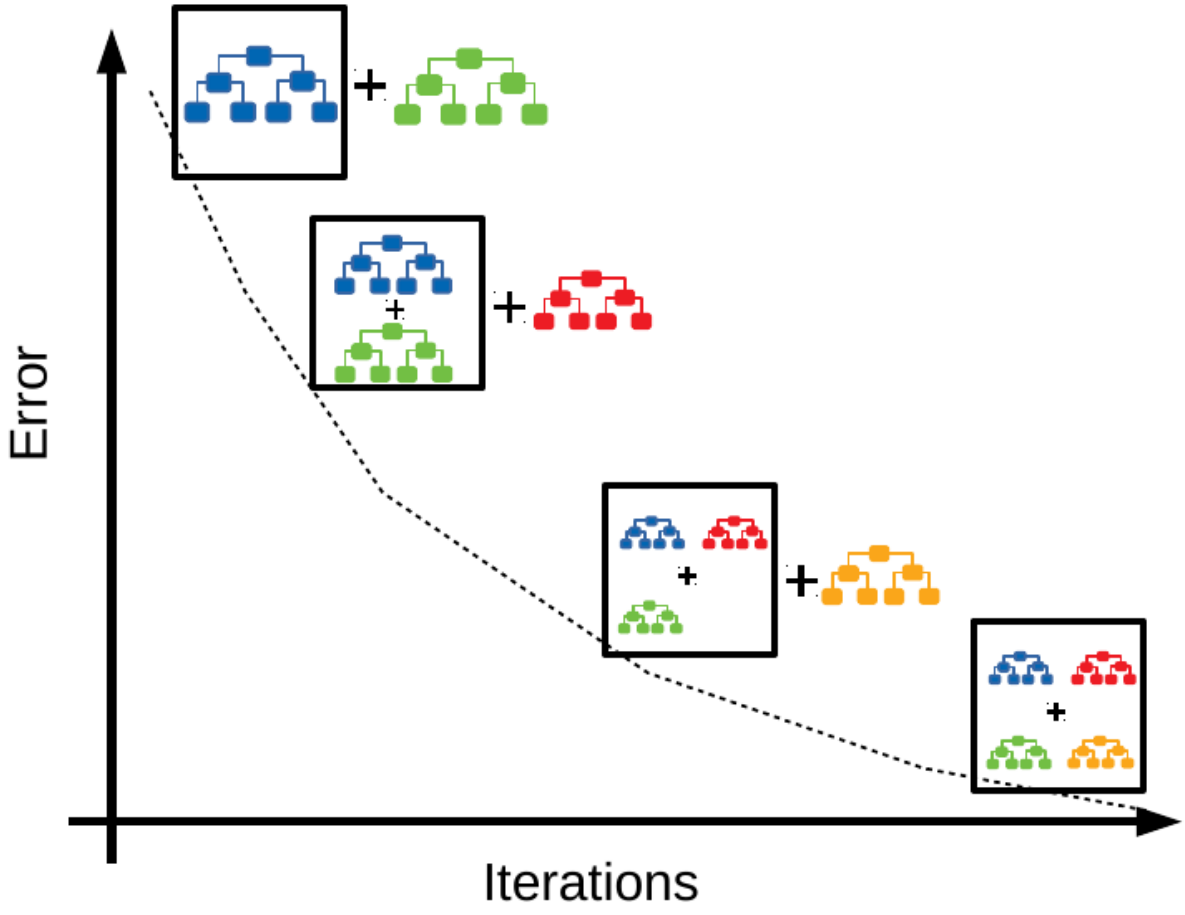
In case of regression, the final result is generated from the average of all weak learners. With classification, the final result can be computed as the class with the majority of votes from weak learners.

In gradient boosting, weak learners work sequentially. Each model tries to improve on the error from the previous model. This is different from the bagging technique, where several models are fitted on subsets of the data in a parallel manner. These subsets are usually drawn randomly with replacement. A great example of bagging is in Random Forests.

The boosting process looks like this:

- Build an initial model with the data,

- Run predictions on the whole data set,
- Calculate the error using the predictions and the actual values,
- Assign more weight to the incorrect predictions,
- Create another model that attempts to fix errors from the last model,
- Run predictions on the entire dataset with the new model,
- Create several models with each model aiming at correcting the errors generated by the previous one,
- Obtain the final model by weighting the mean of all the models.



CODE:

1- Download Libraries, Set Working Directory, Download Dataset

```
import csv
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import xgboost as xgb
veri=pd.read_csv('C:/Users/dekk/Desktop/ml/intubation.csv')
veriler= veri.fillna(0)
```

2-Separating the Data Set into Dependent and Independent Attributes

```
y=veriler.iloc[:,0]
x=veriler.iloc[:,1:]
```

3-Separating Data into Training and Testing

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test=train_test_split(x,y, test_size=0.20, random_state=0)
```

4-Standardization - Feature Scaling

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(x_train)
```

x_train=scaler.transform(x_train)

x_test=scaler.transform(x_test)

5-Creating and Training a Random Forest Model

from sklearn.ensemble import RandomForestClassifier

clf=RandomForestClassifier(n_estimators=10)

clf.fit(x_train,y_train)

6-Making Predictions with a Test Set

y_pred=clf.predict(x_test)

7-Creating the Error Matrix

from sklearn.metrics import classification_report, confusion_matrix

cmatrix=confusion_matrix(y_test,y_pred)

print(cmatrix)

print(classification_report(y_test,y_pred

In [8]:

```
from sklearn.metrics import classification_report, confusion_matrix
cmatrix=confusion_matrix(y_test,y_pred)
print(cmatrix)
print(classification_report(y_test,y_pred))
```

```
[[265   5]
 [ 14   4]]
              precision    recall  f1-score   support

          NO       0.95      0.98      0.97       270
         YES       0.44      0.22      0.30        18

    accuracy                           0.93       288
   macro avg       0.70      0.60      0.63       288
weighted avg       0.92      0.93      0.92       288
```

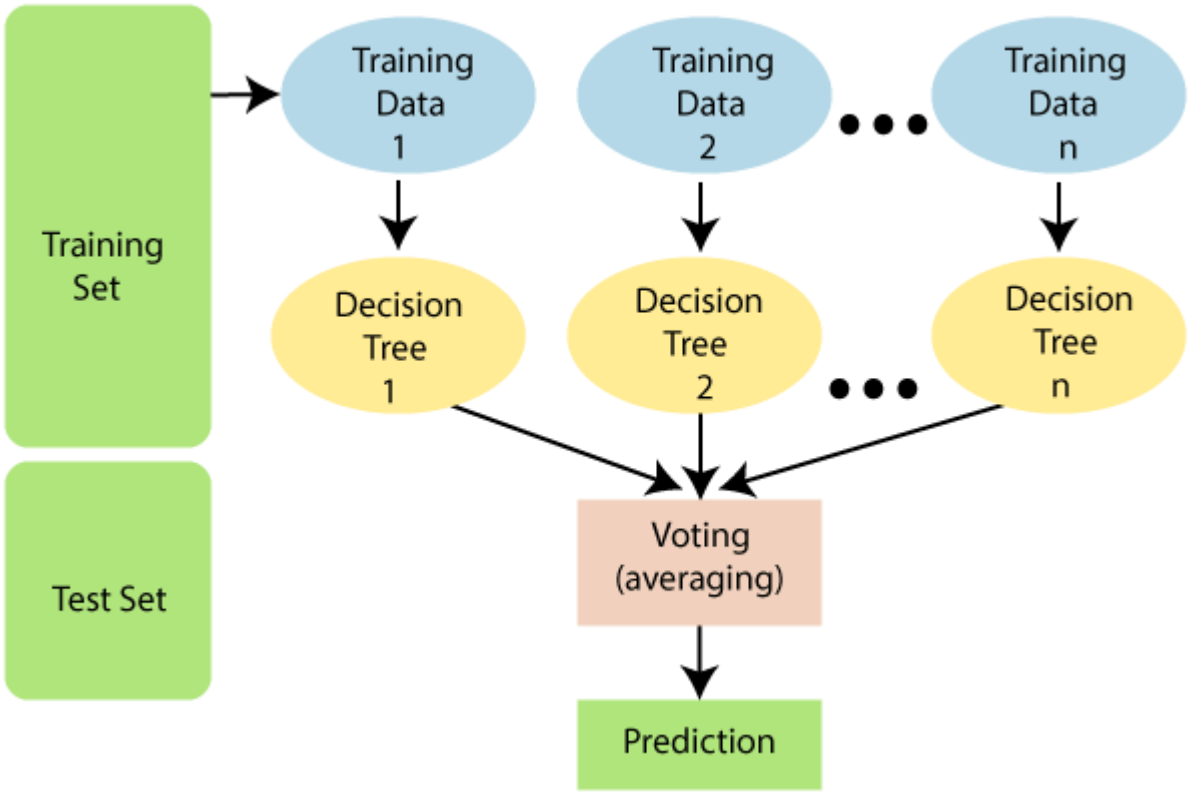
2-Random Forest Algorithm:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of **ensemble learning,** which is a process of *combining multiple classifiers to solve a complex problem and to improve the performance of the model.*

As the name suggests, ***"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."*** Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

The below diagram explains the working of the Random Forest algorithm:



CODE:

1- Import Python's data analysis library and CSV file data

```
import pandas as pd

data = pd.read_csv('C:/Users/dekk/Desktop/ml/intubation.csv')
```

2- Calculated patient height and weight means to missing places in CSV file

```
height_mean = data['HEIGHT'].mean()

weight_mean = data['WEIGHT'].mean()
```

3- Imputed numerical missing values with their means seperately

```
data['HEIGHT'] = data['HEIGHT'].fillna(height_mean)

data['WEIGHT'] = data['WEIGHT'].fillna(weight_mean)
```

4- Dropped null clinical values and drop IDs either

data.dropna(inplace=True)

data.drop(['ID'], axis=1 ,inplace=True)

5- Represented YES/NO values as 1/0

data["INTUBATION"] = data["INTUBATION"].map({'YES': 1, 'NO': 0})

6- Started to classify INTUBATION data

#Pandas drop columns using list of column names

X = data.drop(['INTUBATION', 'INTENSIVE CARE'], axis=1)

Y = data["INTUBATION"]

7-Printing out our model shapes

print(X.shape)

print(Y.shape)

```
In [112]: #Printing out our model shapes
          print(X.shape)
          print(Y.shape)
```

```
(1362, 34)
(1362,)
```

8- Omported scikit-learn's train_test_split library to split those each

from sklearn.model_selection import train_test_split

9- Assigned values in train_test_split function, Test Group is assigned as %25 percent and the rest is defined as Training Group

X_Train, X_Test, Y_Train, Y_Test = train_test_split(X ,Y, test_size = 0.25, random_state = 0)

10- Imported RandomForestClassifier and fitted our Train data

from sklearn.ensemble import RandomForestClassifier

classifier = RandomForestClassifier(n_estimators = 200, random_state = 0)

classifier.fit(X_Train,Y_Train)

In [117]:
```
#Printing out the Random Forest accuracy score
from sklearn.metrics import confusion_matrix, accuracy_score
accuracy_score_rf = accuracy_score(Y_Test, Y_Pred)
print(accuracy_score_rf)
```

```
0.9560117302052786
```

In [115]:
```
#We imported RandomForestClassifier and fitted our Train data
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 200, random_state = 0)
classifier.fit(X_Train,Y_Train)
```

Out[115]:
```
RandomForestClassifier(n_estimators=200, random_state=0)
```

11-Predicting Test data throughout classifier variable

Y_Pred = classifier.predict(X_Test)

12-Printing out the Random Forest accuracy score

from sklearn.metrics import confusion_matrix, accuracy_score

accuracy_score_rf = accuracy_score(Y_Test, Y_Pred)

print(accuracy_score_rf)

```
In [117]: #Printing out the Random Forest accuracy score
          from sklearn.metrics import confusion_matrix, accuracy_score
          accuracy_score_rf = accuracy_score(Y_Test, Y_Pred)
          print(accuracy_score_rf)

          0.9560117302052786
```

```
In [118]: print("RANDOM FOREST ALGORITHM ACCURACY SCORE")
          print(round(accuracy_score_rf,2,), "%")

          RANDOM FOREST ALGORITHM ACCURACY SCORE
          0.96 %
```

13-Receiving algorithm performance measure formulas as below

tn, fp, fn, tp = confusion_matrix(Y_Test, Y_Pred).ravel()

14- Imported Python's Numpy library and created two empty arrays

import numpy as np

sensitivity_rf = np.empty(0)

specificity_rf = np.empty(0)

15-Retrieving Specificity and Sensitivity formulas from 4 variable and assigned those arrays

sensitivity_rf = np.append(sensitivity_rf, np.array([tp / (tp + fn)]))

specificity_rf = np.append(specificity_rf, np.array([tn / (tn + fp)]))

16-Printing out the Specificity and Sensitivity variables

print(sensitivity_rf)

print(specificity_rf)

```
In [122]: #Printing out the Specificity and Sensitivity variables
          print(sensitivity_rf)
          print(specificity_rf)

          [0.25]
          [1.]
```

17- Imported GradientBoostingClassifier and fitted our Train data

from sklearn.ensemble import GradientBoostingClassifier

gradient_boosting = GradientBoostingClassifier(n_estimators = 100, max_depth = 3)

gradient_boosting.fit(X_Train, Y_Train)

```
In [104]:
          tn, fp, fn, tp = confusion_matrix(Y_Test, prediction_gb).ravel()
          sensitivity_gb = np.empty(0)
          specificity_gb = np.empty(0)
          sensitivity_gb = np.append(sensitivity_gb, np.array([tp / (tp + fn)]))
          specificity_gb = np.append(specificity_gb, np.array([tn / (tn + fp)]))
          print(sensitivity_gb)
          print(specificity_gb)

          [0.3]
          [0.97819315]
```

18-Printing out the Gradient Boosting accuracy score

acc_gradient = round(gradient_boosting.score(X_Train, Y_Train) * 100, 2)

print(round(acc_gradient,2,), "%")

```
In [124]: #Printing out the Gradient Boosting accuracy score
          acc_gradient = round(gradient_boosting.score(X_Train, Y_Train) * 100, 2)
          print(round(acc_gradient,2,), "%")

          99.8 %
```

19-Predicting Test data throughout classifier variable and process same operations as we did before

prediction_gb = gradient_boosting.predict(X_Test)

accuracy_score_gb = accuracy_score(prediction_gb, Y_Test)

print(accuracy_score_gb)

```
In [125]: #Predicting Test data throughout classifier variable and process same operations as we did before
          prediction_gb = gradient_boosting.predict(X_Test)
          accuracy_score_gb = accuracy_score(prediction_gb, Y_Test)
          print(accuracy_score_gb)

          0.9413489736070382
```

tn, fp, fn, tp = confusion_matrix(Y_Test, prediction_gb).ravel()

sensitivity_gb = np.empty(0)

specificity_gb = np.empty(0)

sensitivity_gb = np.append(sensitivity_gb, np.array([tp / (tp + fn)]))

specificity_gb = np.append(specificity_gb, np.array([tn / (tn + fp)]))

print(sensitivity_gb)

print(specificity_gb)

In [126]:
```
tn, fp, fn, tp = confusion_matrix(Y_Test, prediction_gb).ravel()
sensitivity_gb = np.empty(0)
specificity_gb = np.empty(0)
sensitivity_gb = np.append(sensitivity_gb, np.array([tp / (tp + fn)]))
specificity_gb = np.append(specificity_gb, np.array([tn / (tn + fp)]))
print(sensitivity_gb)
print(specificity_gb)
```

```
[0.25]
[0.98442368]
```

20- About to classify INTENSIVE CARE data

(Same operation proceeded like before)

data["INTENSIVE CARE"] = data["INTENSIVE CARE"].map({'YES': 1, 'NO': 0})

X = data.drop(['INTUBATION', 'INTENSIVE CARE'], axis=1)

Y = data["INTENSIVE CARE"]

21-Printing out our second model shapes

print(X.shape)

print(Y.shape)

In [128]:
```
#Printing out our second model shapes
print(X.shape)
print(Y.shape)
```

```
(1362, 34)
(1362,)
```

22- Fitted our Train data for our 2nd classification

classifier = RandomForestClassifier(n_estimators = 200, random_state = 0)

classifier.fit(X_Train,Y_Train)

```
In [129]:
#We fitted our Train data for our 2nd classification
classifier = RandomForestClassifier(n_estimators = 200, random_state = 0)
classifier.fit(X_Train,Y_Train)
```

```
Out[129]: RandomForestClassifier(n_estimators=200, random_state=0)
```

23-Predicting Test data throughout classifier variable

Y_Pred = classifier.predict(X_Test)

24-Printing out the Random Forest accuracy score

accuracy_score_rf = accuracy_score(Y_Test, Y_Pred)

print(accuracy_score_rf)

25-Receiving algorithm performance measure formulas as below

tn, fp, fn, tp = confusion_matrix(Y_Test, Y_Pred).ravel()

26- Imported Python's Numpy library and created two empty arrays

import numpy as np

sensitivity_rf = np.empty(0)

specificity_rf = np.empty(0)

27-Retrieving Specificity and Sensitivity formulas from 4 variable

sensitivity_rf = np.append(sensitivity_rf, np.array([tp / (tp + fn)]))

specificity_rf = np.append(specificity_rf, np.array([tn / (tn + fp)]))

28-Printing out the Specificity and Sensitivity variables

print(sensitivity_rf)

print(specificity_rf)

```
In [136]: #Printing out the Specificity and Sensitivity variables
          print(sensitivity_rf)
          print(specificity_rf)

          [0.25]
          [1.]
```

29- Imported GradientBoostingClassifier and fitted our Train data

from sklearn.ensemble import GradientBoostingClassifier

gradient_boosting = GradientBoostingClassifier(n_estimators = 100, max_depth = 3)

gradient_boosting.fit(X_Train, Y_Train)

```
In [137]: #We imported GradientBoostingClassifier and fitted our Train data
          from sklearn.ensemble import GradientBoostingClassifier
          gradient_boosting = GradientBoostingClassifier(n_estimators = 100, max_depth = 3)
          gradient_boosting.fit(X_Train, Y_Train)

Out[137]: GradientBoostingClassifier()
```

30-Printing out the Gradient Boosting accuracy score

acc_gradient = round(gradient_boosting.score(X_Train, Y_Train) * 100, 2)

print(round(acc_gradient,2,), "%")

```
In [138]: #Printing out the Gradient Boosting accuracy score
          acc_gradient = round(gradient_boosting.score(X_Train, Y_Train) * 100, 2)
          print(round(acc_gradient,2,), "%")
```

```
99.9 %
```

31-Predicting Test data throughout classifier variable and process same operations as we did before

prediction_gb = gradient_boosting.predict(X_Test)

accuracy_score_gb = accuracy_score(prediction_gb, Y_Test)

print(accuracy_score_gb)

```
In [139]: #Predicting Test data throughout classifier variable and process same operations as we did before
          prediction_gb = gradient_boosting.predict(X_Test)
          accuracy_score_gb = accuracy_score(prediction_gb, Y_Test)
          print(accuracy_score_gb)
```

```
0.9384164222873901
```

tn, fp, fn, tp = confusion_matrix(Y_Test, prediction_gb).ravel()

sensitivity_gb = np.empty(0)

specificity_gb = np.empty(0)

sensitivity_gb = np.append(sensitivity_gb, np.array([tp / (tp + fn)]))

specificity_gb = np.append(specificity_gb, np.array([tn / (tn + fp)]))

print(sensitivity_gb)

print(specificity_gb)

In [104]:

```
tn, fp, fn, tp = confusion_matrix(Y_Test, prediction_gb).ravel()
sensitivity_gb = np.empty(0)
specificity_gb = np.empty(0)
sensitivity_gb = np.append(sensitivity_gb, np.array([tp / (tp + fn)]))
specificity_gb = np.append(specificity_gb, np.array([tn / (tn + fp)]))
print(sensitivity_gb)
print(specificity_gb)
```

```
[0.3]
[0.97819315]
```

REFERENCES

1- https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d
2- https://towardsdatascience.com/machine-learning-part-18-boosting-algorithms-gradient-boosting-in-python-ef5ae6965be4
3- https://en.wikipedia.org/wiki/Gradient_boosting
4- https://www.veribilimiokulu.com/siniflandirma-notlari-18-random-forest-python-uygulama/
5- https://builtin.com/data-science/random-forest-algorithm
6- https://www.javatpoint.com/machine-learning-random-forest-algorithm
7- https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/
8- https://blog.mlreview.com/gradient-boosting-from-scratch-1e317ae4587d
9- https://docs.rapidminer.com/latest/studio/operators/modeling/predictive/trees/gradient_boosted_trees.html
10- https://neptune.ai/blog/gradient-boosted-decision-trees-guide