Implementation of Classification and DSS To Diagnose Obesity in a Person

Prepared by : Bishal Adhikari

Date : June, 2022

- **Introduction**

In this project, we are analyzing a dataset which classifies obesity levels in a patient based on different attributes. The dataset is taken from UCI Machine Learning Repository and contains 2111 instances and 17 attributes. The original data classifies a broad category of obesity , overweight levels, underweight condition and normal weight condition. For convenience we have reduced the classes to three categories : ‘Obesity’, ‘Overweight’ and ‘No Obesity.’

- **Goal of the project**

1. To conduct classification on obesity dataset using DecisionTreeClassifier

2. To understand how the different hyperparameters affect the accurracy of the classifier mmodel

3. To use the result of classification to derive a rule-based program(DSS) that diagnose obesity levels

4. To use DSS to predict the obesity levels in a person based on certain attributes

DATASET

- This dataset include data for the estimation of obesity levels in individuals from the countries of Mexico, Peru and Colombia, based on their eating habits and physical condition.

```
data = pd.read_csv("ObesityDataSet_raw.csv")
data.head()
```

	Gender	Age	Height	Weight	family_history_with_overweight	FAVC	FCVC	NCP	CAEC	SMOKE	CH2O	SCC	FAF	TUE	CALC	MTRAI
0	Female	21.0	1.62	64.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	0.0	1.0	no	Public_Transportati
1	Female	21.0	1.52	56.0	yes	no	3.0	3.0	Sometimes	yes	3.0	yes	3.0	0.0	Sometimes	Public_Transportati
2	Male	23.0	1.80	77.0	yes	no	2.0	3.0	Sometimes	no	2.0	no	2.0	1.0	Frequently	Public_Transportati
3	Male	27.0	1.80	87.0	no	no	3.0	3.0	Sometimes	no	2.0	no	2.0	0.0	Frequently	Walki
4	Male	22.0	1.78	89.8	no	no	2.0	1.0	Sometimes	no	2.0	no	0.0	0.0	Sometimes	Public_Transportati

- Attributes
  - Frequent consumption of high caloric food (FAVC)
  - Frequency of consumption of vegetables (FCVC)
  - Number of main meals (NCP)
  - Consumption of food between meals (CAEC)
  - Consumption of water daily (CH20)
  - Consumption of alcohol (CALC)
  - Calories consumption monitoring (SCC)
  - Physical activity frequency (FAF)
  - Time using technology devices (TUE)
  - Transportation used (MTRANS)
  - Gender
  - Age
  - Height
  - Weight.

- Classification Targets
  - Original Targets(7 classes)

```
data['NObeyesdad'].unique()
```

```
array(['Normal_Weight', 'Overweight_Level_I', 'Overweight_Level_II',
       'Obesity_Type_I', 'Insufficient_Weight', 'Obesity_Type_II',
       'Obesity_Type_III'], dtype=object)
```

  - Operation

```
replace_dict = {'Obesity_Type_I' : 'Obesity',
                'Obesity_Type_III' : 'Obesity',
                'Obesity_Type_II': 'Obesity',
                'Overweight_Level_I': 'Overweight',
                'Overweight_Level_II':'Overweight',
                'Normal_Weight': 'No Obesity',
                'Insufficient_Weight':'No Obesity'
               }
data['NObeyesdad'] = data['NObeyesdad'].replace(replace_dict)
```

  - Modified Targets(3 classes)

```
data['NObeyesdad'].unique()
```

```
array(['No Obesity', 'Overweight', 'Obesity'], dtype=object)
```

- Data Preparations

- All categorical columns were encoded using labelencoder.

- Numeric columns were not scaled.

```
for cols in cat_cols:
    obesity[cols].astype('category')
```

```
cat_df = obesity[[cols for cols in cat_cols]]
num_df = obesity.drop(cat_df,axis = 1)
nums = list(num_df.columns)
cats = list(cat_df.columns)
```

```
class MultiColumnLabelEncoder:
    def __init__(self,columns = None):
        self.columns = columns
    def fit(self,X,y=None):
        return self
    def transform(self,X):
        output = X.copy()
        if self.columns is not None:
            for col in self.columns:
                output[col] = LabelEncoder().fit_transform(output[col])
        else:
            for colname,col in output.iteritems():
                output[colname] = LabelEncoder().fit_transform(col)
        return output

    def fit_transform(self,X,y=None):
        return self.fit(X,y).transform(X)
```

```
obesity_final = MultiColumnLabelEncoder(cat_cols).fit_transform(obesity)
test_obesity_final = MultiColumnLabelEncoder(cat_cols).fit_transform(test_obesity)
```

Running Classifier

- Run 1(criterion = gini, ccp_alpha = 0.01) ccp-alpha is responsible for pruning trees

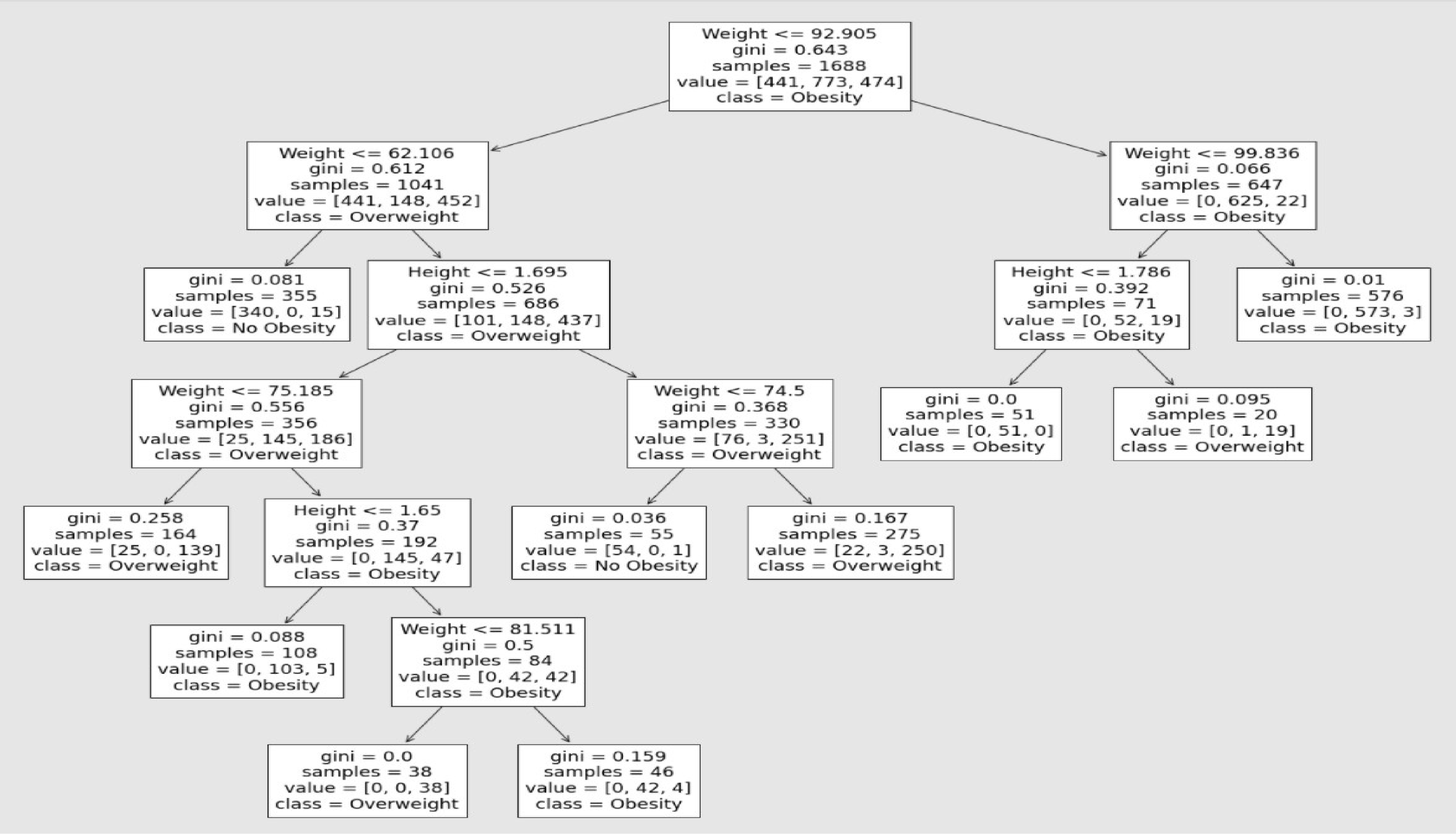
```
model = DecisionTreeClassifier( criterion = 'gini', ccp_alpha = 0.01)
model.fit(obesity_final, obesity_labels)
fig = plt.figure(figsize=(25,20))
_ = plot_tree(model,class_names = obesity_labels.unique(),feature_names = obesity.columns)
cross_val_score(model,obesity_final, obesity_labels,cv = 10)
```

```
array([0.94674556, 0.95857988, 0.92899408, 0.96449704, 0.95857988,
       0.91715976, 0.92307692, 0.94674556, 0.95238095, 0.94642857])
```

```
model.feature_importances_
```

```
array([0.        , 0.        , 0.09197612, 0.90802388, 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        ])
```

- Decision Tree from Run #1



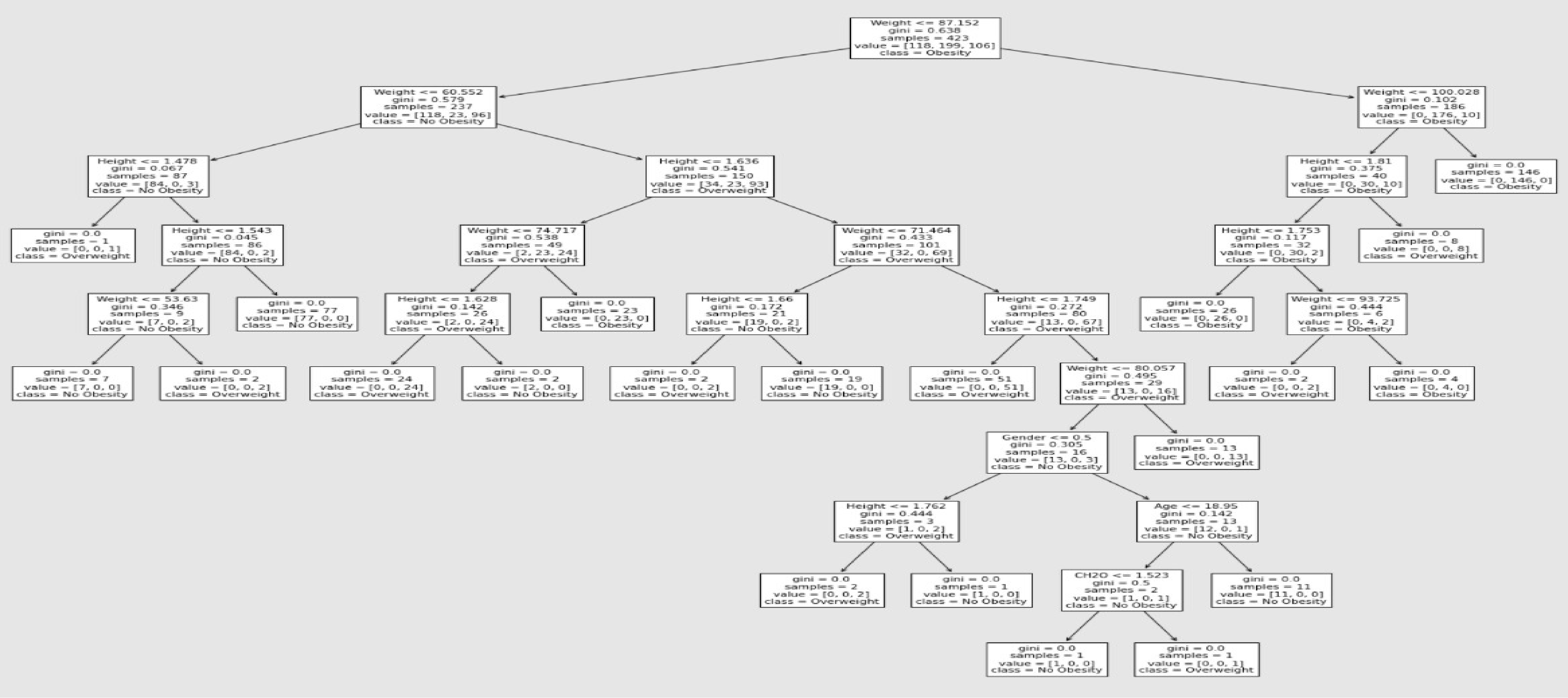
Run #2(max_depth = 10) no pruning

```
model = DecisionTreeClassifier( max_depth = 10)
model.fit(test_obesity_final, test_labels)
fig = plt.figure(figsize=(25,20))
_ = plot_tree(model,class_names = obesity_labels.unique(),feature_names = obesity.columns)
cross_val_score(model,test_obesity_final, test_labels,cv = 10)
```

```
array([0.95348837, 0.97674419, 0.88372093, 0.97619048, 0.92857143,
       0.92857143, 0.97619048, 0.97619048, 0.92857143, 0.95238095])
```

With higher max_depth, the accurracy improved but the tree became complex.

Decision Tree from Run#2



- As we saw in Run #1 and Run #2 , with no pruning in Run #2 the tree was very complex and was dependent on more attributes.

- But in Run #1 effective pruning allowed the tree to remain small , depend on less/best attributes and still gave the similar accurracy.

- Run 3 (criterion = entropy, ccp_alpha = 0.01, max_leaf_nodes = 8)

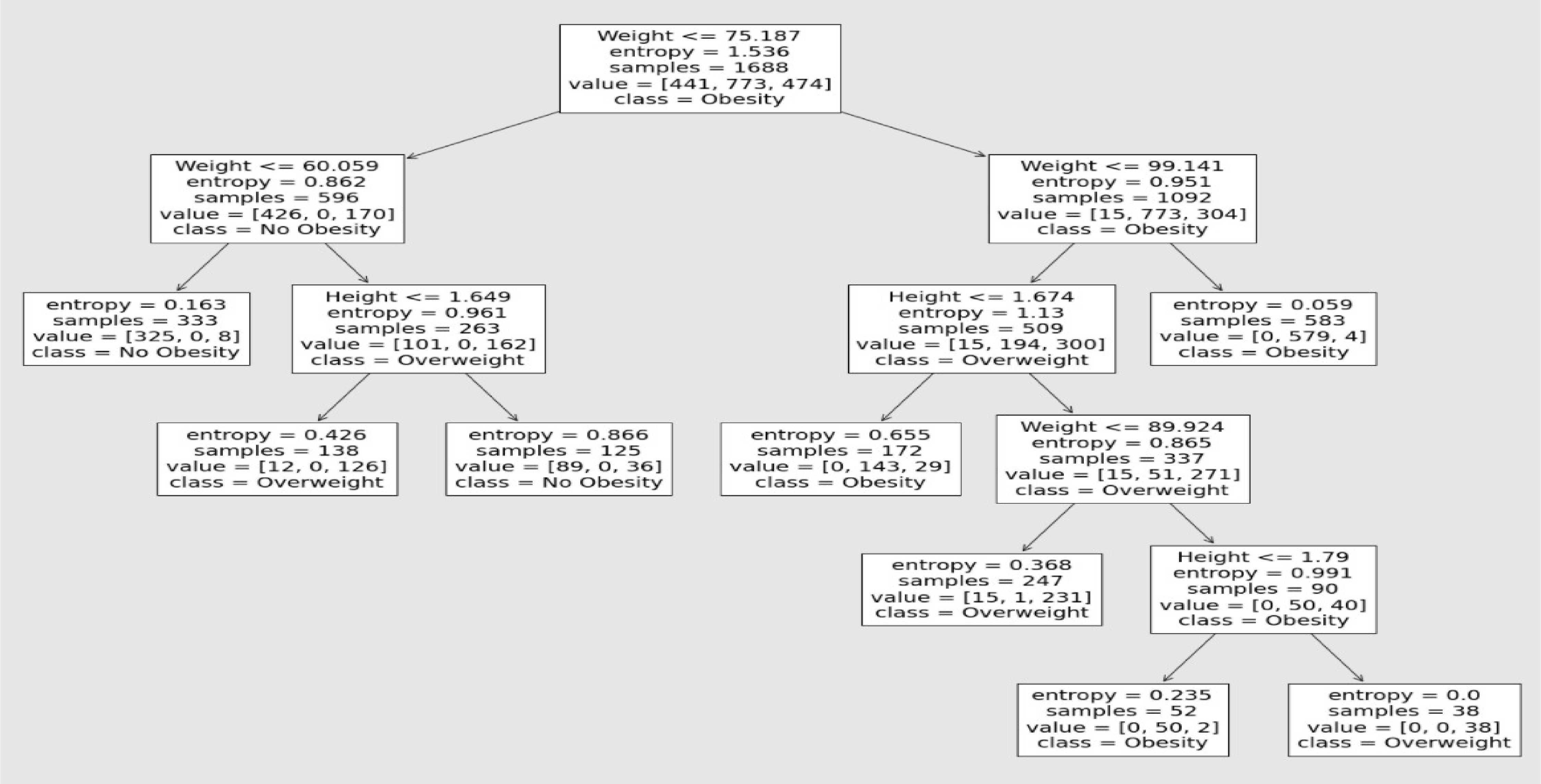
```
model = DecisionTreeClassifier( criterion = 'entropy', ccp_alpha = 0.01, max_leaf_nodes = 8)
model.fit(obesity_final, obesity_labels)
fig = plt.figure(figsize=(25,20))
_ = plot_tree(model,class_names = obesity_labels.unique(),feature_names = obesity.columns)
cross_val_score(model,obesity_final, obesity_labels,cv = 10)
```

```
array([0.94674556, 0.9112426 , 0.9408284 , 0.95857988, 0.95857988,
       0.93491124, 0.9112426 , 0.92307692, 0.92261905, 0.88095238])
```

```
model.feature_importances_
```

```
array([0.        , 0.        , 0.15720822, 0.84279178, 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        ])
```

- Decision Tree from Run #3

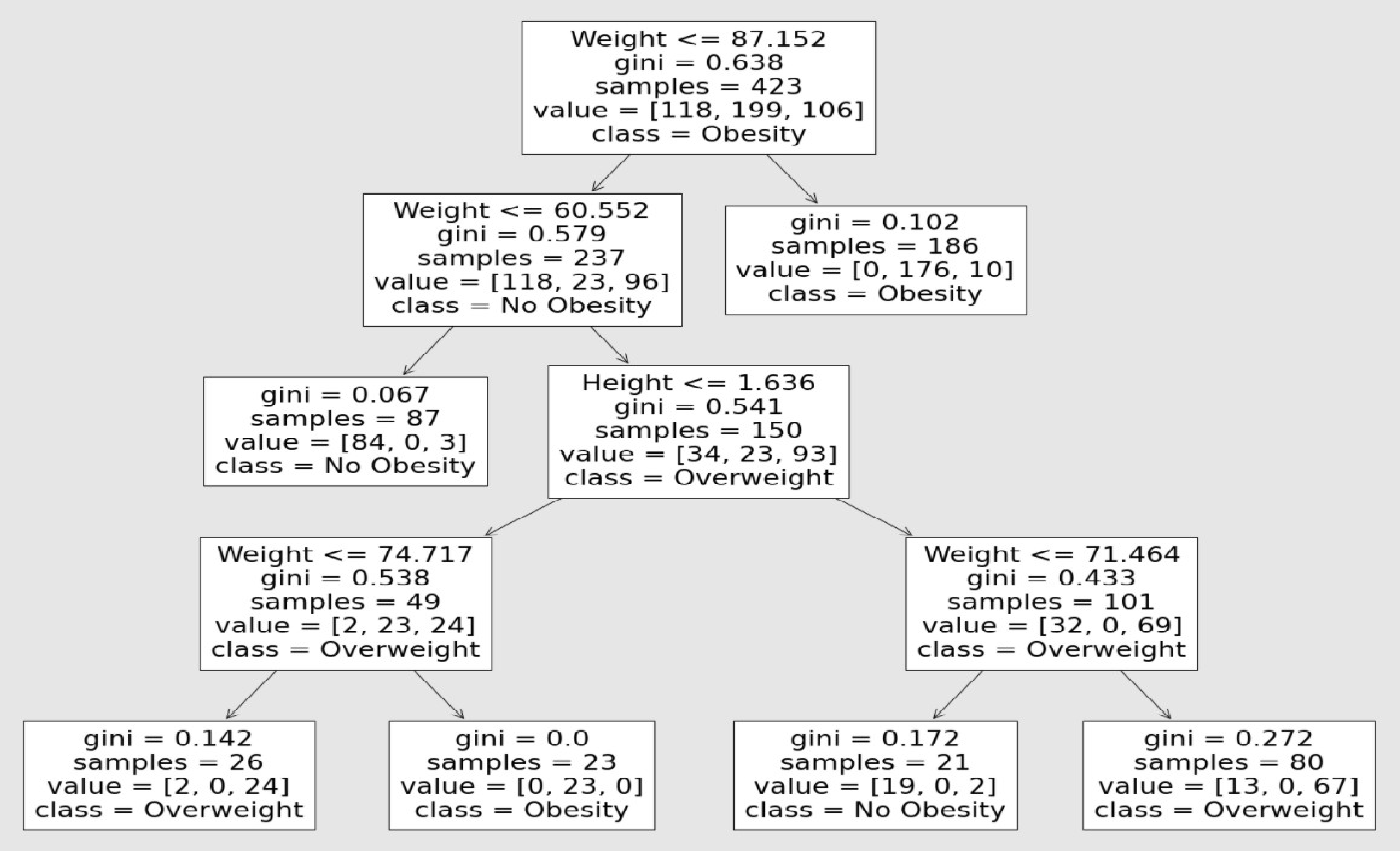


Run #4(max_depth = 4, max_leaf_nodes = 6)

```
model = DecisionTreeClassifier( max_depth = 4, max_leaf_nodes = 6)
model.fit(test_obesity_final, test_labels)
fig = plt.figure(figsize=(25,20))
_ = plot_tree(model,class_names = obesity_labels.unique(),feature_names = obesity.columns)
cross_val_score(model,test_obesity_final, test_labels,cv = 10)
```

```
array([0.95348837, 0.88372093, 0.86046512, 0.9047619 , 0.88095238,
       0.92857143, 0.85714286, 0.95238095, 0.95238095, 0.9047619 ])
```

Decision Tree From Run #4



Decision Support System

- **Rules manually derived from a selected decision tree**

In [30]:
```python
def obesity_diagnosis(height, weight, age, gender, CH2O):
    if weight <= 87.15 :
        if weight <= 60.55:
            return'No Obesity'
        elif weight > 60.55 :
            if height <= 1.64:
                if weight <= 74.71:
                    return'Overweight'
                elif weight > 74.71:
                    return 'Obesity'
            elif height > 1.64:
                if weight <= 71.46:
                    return 'No Obesity'
                elif weight > 71.46:
                    if height <= 1.75:
                        return 'Overweight'
                    elif height > 1.75:
                        if weight <= 80.06:
                            return 'No Obesity'
                        elif weight > 80.06:
                            return 'Overweight'
    elif weight > 87.15:
        if weight <= 100.03:
            if height <= 1.81:
                return 'Obesity'
            elif height > 1.81:
                return 'Overweight'
        elif weight > 100.03:
            return 'Obesity'
```

Test Run DSS

```
obesity_diagnosis(1.4, 65, 32, 'Male',3)
```

'Overweight'

```
obesity_diagnosis(1.6, 55, 32, 'Female',2)
```

'No Obesity'

```
obesity_diagnosis(1.3, 85, 32, 'Male',3)
```

'Obesity'

```
obesity_diagnosis(1.5, 58, 32, 'Male',3)
```

'No Obesity'

CONCLUSIONS

Hence, we successfully implemented Decision Tree Classifier to the obesity dataset. We saw that among all the attributes, most of the information was contained by only weight and height.

We got accurracy as high as 97% with the decision trees and it was surprising to see that with only height and weight, obesity levels of a person coold be identified or diagnosed.

Regarding hyperparameters:

- other parameters constant, increasing max_depth increased accurracy of the model
- max_leaf_nodes did the same as max_depth in contributing to accurracy
- pruning of trees by specifying ccp_alpha was very helpful in getting high accurracy with less complicated trees
- calculation of information gain using either gini or entropy didn't affect the accurracy of our model in big way

- We hereby conclude, DecisionTree Classifier is a powerful tool . Using Decision Trees it is possible to derive simple rules in predicting classes (obesity levels) in our case.

!! THANK YOU!!