Meeting 6

10/14/2021

Chisom Sopuruchi

Deliverables

- Apply moving average to temp and HR plots
- Review and Implement Decision trees

Methodology and Learnings

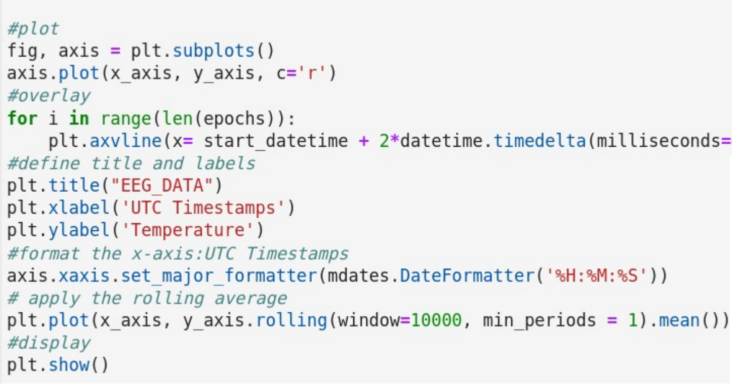
How did I do it?

- Apply rolling average with window frame to temp and HR plots
- Review decision tree and try sample decision tree examples

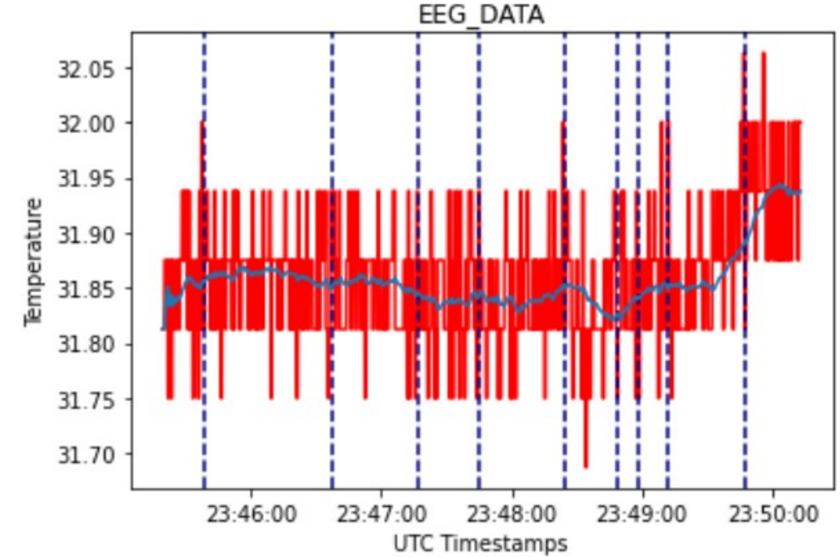
What did I learn on the way?

- Moving Average
- Decision Trees

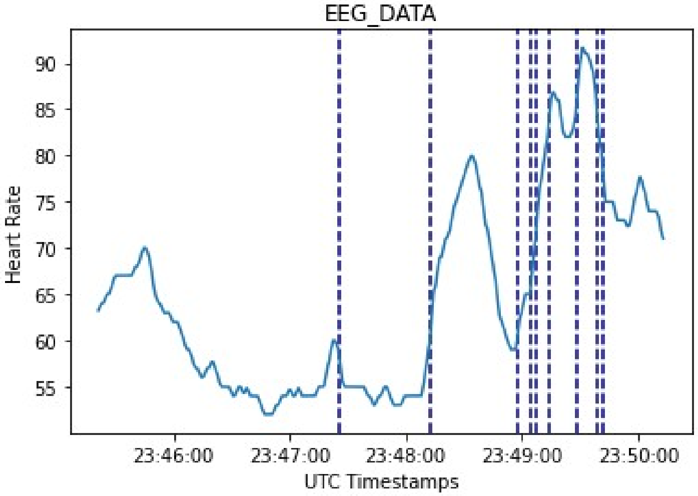
Results

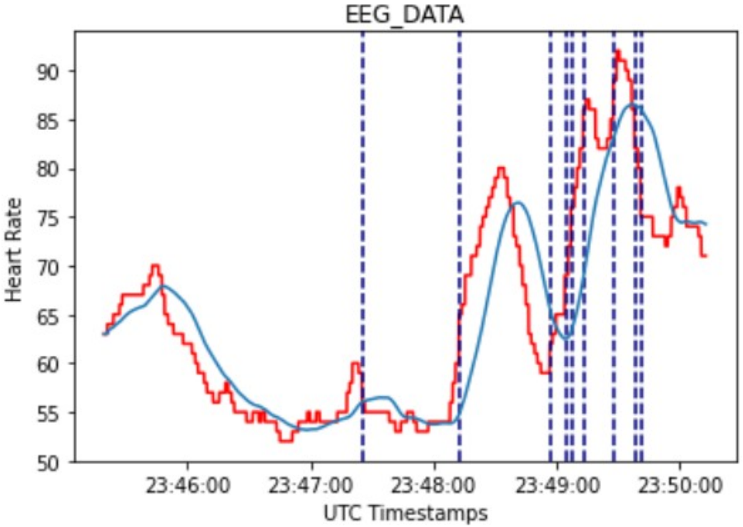


```
#plot
fig, axis = plt.subplots()
axis.plot(x_axis, y_axis, c='r')
#overlay
for i in range(len(epochs)):
    plt.axvline(x= start_datetime + 2*datetime.timedelta(milliseconds=epochs[i].item()), c='navy', linestyle='dashed')
#define title and labels
plt.title("EEG_DATA")
plt.xlabel('UTC Timestamps')
plt.ylabel('Temperature')
#format the x-axis:UTC Timestamps
axis.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M:%S'))
# apply the rolling average
plt.plot(x_axis, y_axis.rolling(window=10000, min_periods = 1).mean())
#display
plt.show()
```

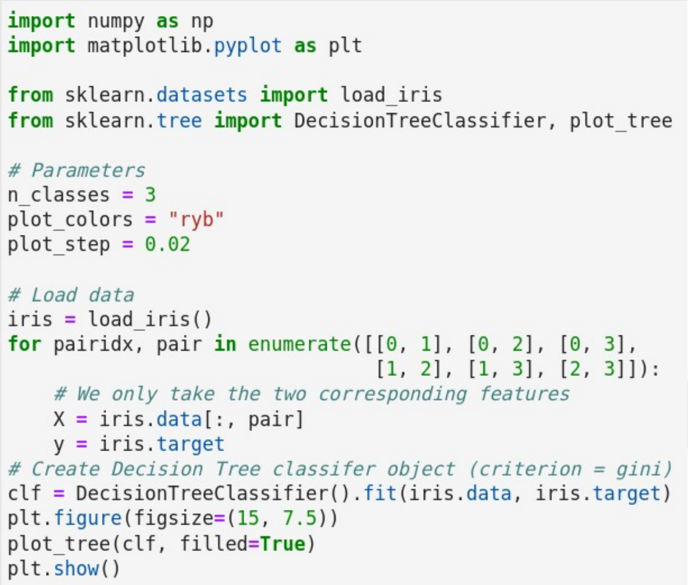


Results

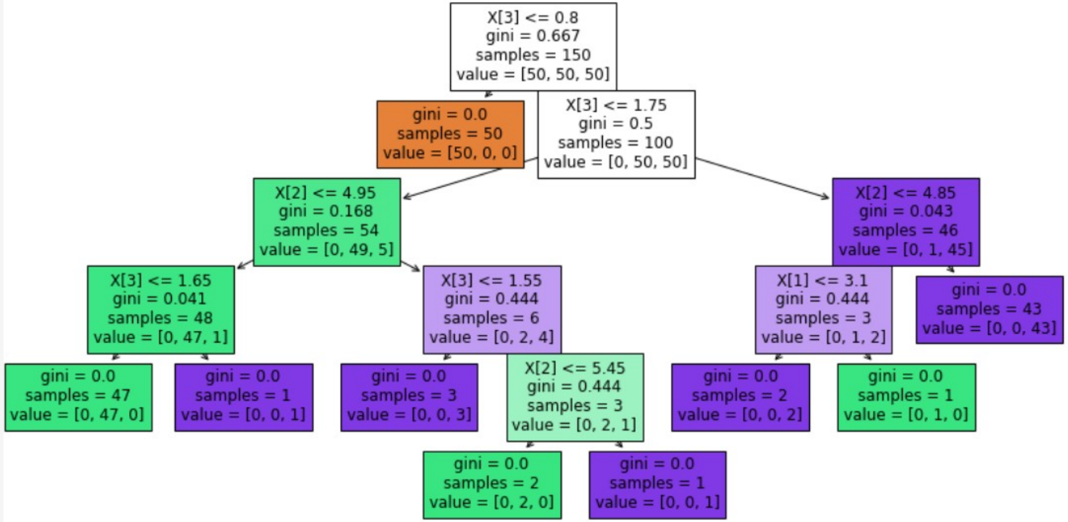

EEG_DATA
Heart Rate
90
85
80
75
70
65
60
55
23:46:00
23:47:00
23:48:00
23:49:00
23:50:00
UTC Timestamps


EEG_DATA
Heart Rate
90
85
80
75
70
65
60
55
50
23:46:00
23:47:00
23:48:00
23:49:00
23:50:00
UTC Timestamps

Results

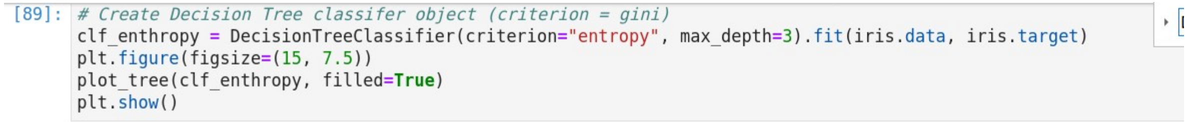


```
import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree

# Parameters
n_classes = 3
plot_colors = "ryb"
plot_step = 0.02

# Load data
iris = load_iris()
for pairidx, pair in enumerate([[0, 1], [0, 2], [0, 3],
                                [1, 2], [1, 3], [2, 3]]):
    # We only take the two corresponding features
    X = iris.data[:, pair]
    y = iris.target
# Create Decision Tree classifer object (criterion = gini)
clf = DecisionTreeClassifier().fit(iris.data, iris.target)
plt.figure(figsize=(15, 7.5))
plot_tree(clf, filled=True)
plt.show()
```

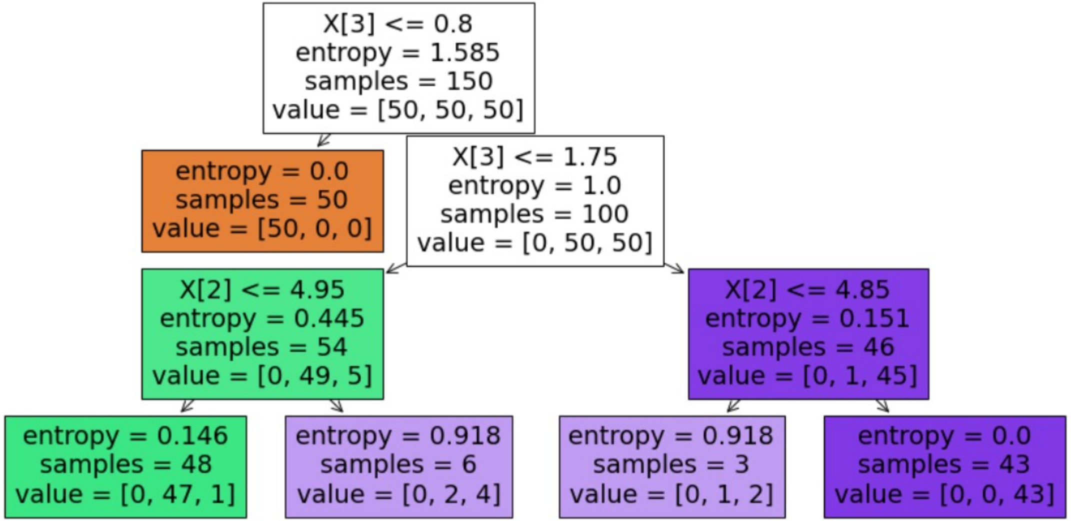


X[3] <= 0.8 gini = 0.667 samples = 150 value = [50, 50, 50]
gini = 0.0 samples = 50 value = [50, 0, 0]
X[3] <= 1.75 gini = 0.5 samples = 100 value = [0, 50, 50]
X[2] <= 4.95 gini = 0.168 samples = 54 value = [0, 49, 5]
X[2] <= 4.85 gini = 0.043 samples = 46 value = [0, 1, 45]
X[3] <= 1.65 gini = 0.041 samples = 48 value = [0, 47, 1]
X[3] <= 1.55 gini = 0.444 samples = 6 value = [0, 2, 4]
X[1] <= 3.1 gini = 0.444 samples = 3 value = [0, 1, 2]
gini = 0.0 samples = 43 value = [0, 0, 43]
gini = 0.0 samples = 47 value = [0, 47, 0]
gini = 0.0 samples = 1 value = [0, 0, 1]
gini = 0.0 samples = 3 value = [0, 0, 3]
X[2] <= 5.45 gini = 0.444 samples = 3 value = [0, 2, 1]
gini = 0.0 samples = 2 value = [0, 0, 2]
gini = 0.0 samples = 1 value = [0, 1, 0]
gini = 0.0 samples = 2 value = [0, 2, 0]
gini = 0.0 samples = 1 value = [0, 0, 1]

Results



```
[89]: # Create Decision Tree classifer object (criterion = gini)
      clf_enthropy = DecisionTreeClassifier(criterion="entropy", max_depth=3).fit(iris.data, iris.target)
      plt.figure(figsize=(15, 7.5))
      plot_tree(clf_enthropy, filled=True)
      plt.show()
```



X[3] <= 0.8
entropy = 1.585
samples = 150
value = [50, 50, 50]

entropy = 0.0
samples = 50
value = [50, 0, 0]

X[3] <= 1.75
entropy = 1.0
samples = 100
value = [0, 50, 50]

X[2] <= 4.95
entropy = 0.445
samples = 54
value = [0, 49, 5]

X[2] <= 4.85
entropy = 0.151
samples = 46
value = [0, 1, 45]

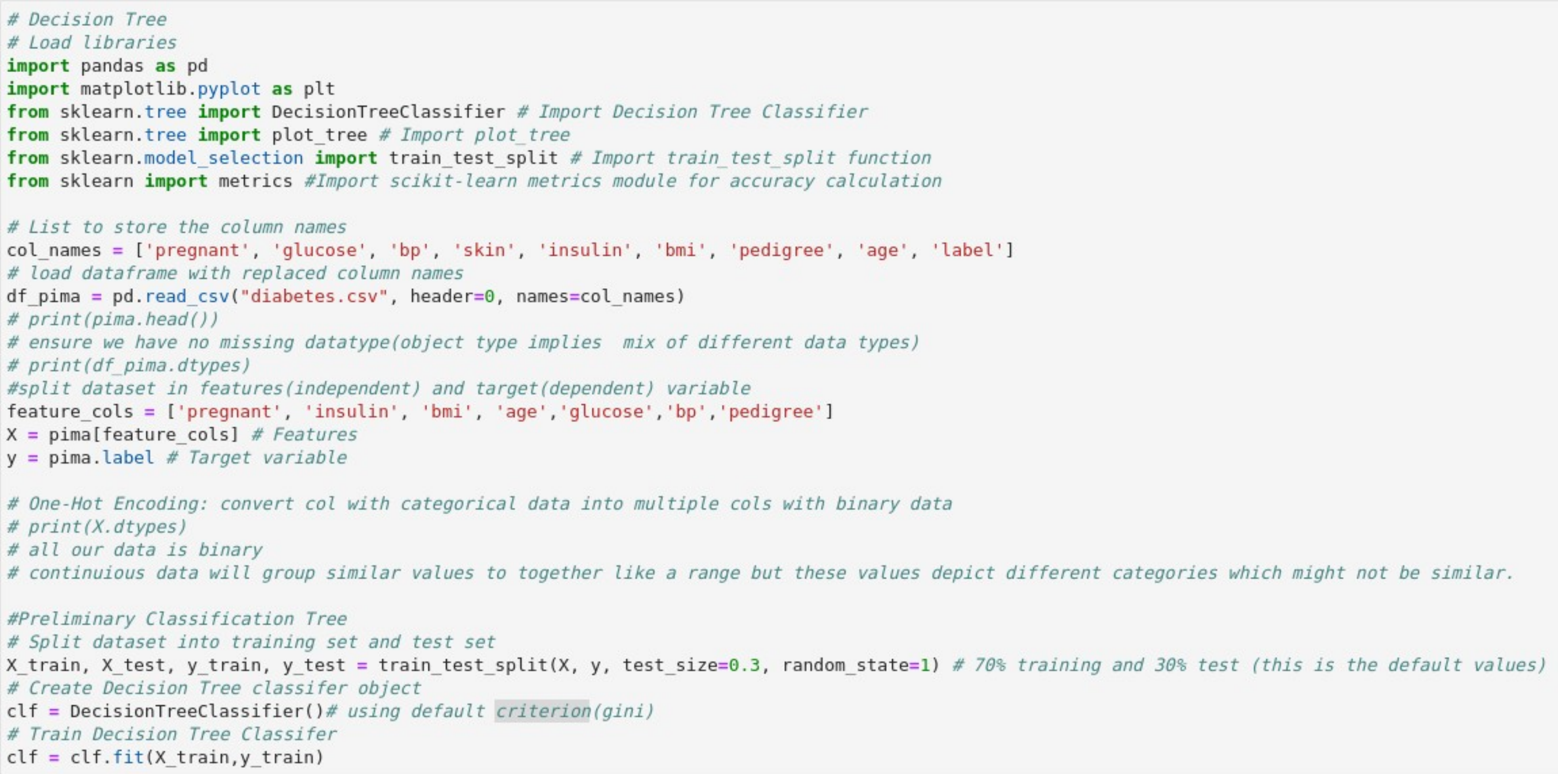
entropy = 0.146
samples = 48
value = [0, 47, 1]

entropy = 0.918
samples = 6
value = [0, 2, 4]

entropy = 0.918
samples = 3
value = [0, 1, 2]

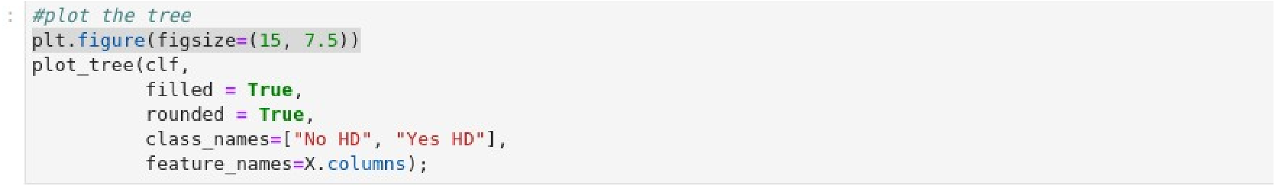
entropy = 0.0
samples = 43
value = [0, 0, 43]

Results

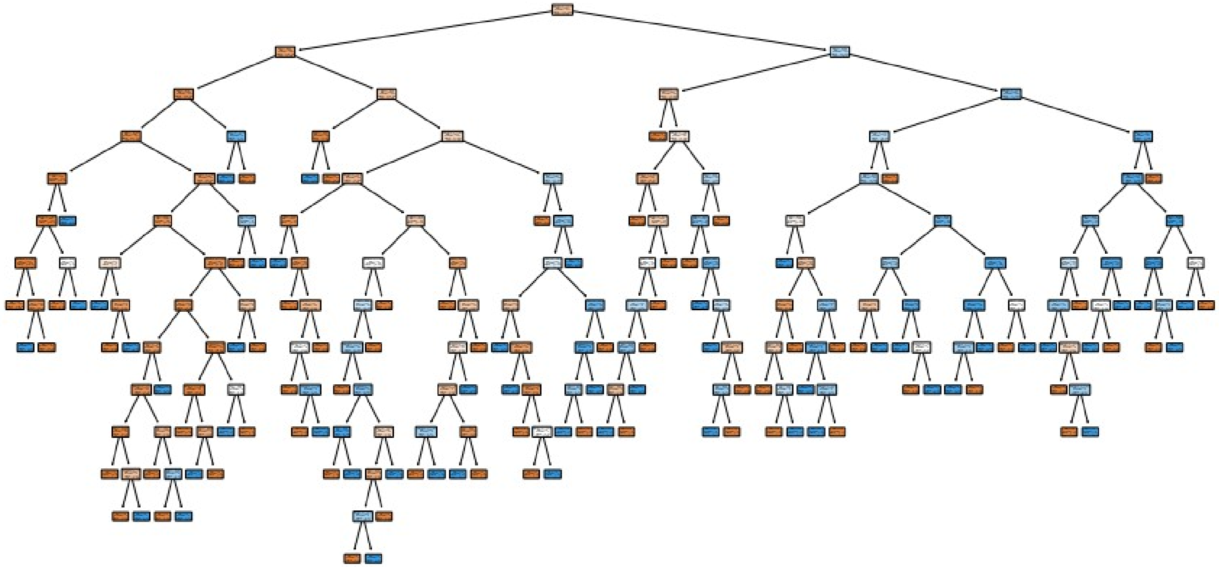


```
# Decision Tree
# Load libraries
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree Classifier
from sklearn.tree import plot_tree # Import plot_tree
from sklearn.model_selection import train_test_split # Import train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for accuracy calculation

# List to store the column names
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'age', 'label']
# load dataframe with replaced column names
df_pima = pd.read_csv("diabetes.csv", header=0, names=col_names)
# print(pima.head())
# ensure we have no missing datatype(object type implies  mix of different data types)
# print(df_pima.dtypes)
#split dataset in features(independent) and target(dependent) variable
feature_cols = ['pregnant', 'insulin', 'bmi', 'age','glucose','bp','pedigree']
X = pima[feature_cols] # Features
y = pima.label # Target variable

# One-Hot Encoding: convert col with categorical data into multiple cols with binary data
# print(X.dtypes)
# all our data is binary
# continuious data will group similar values to together like a range but these values depict different categories which might not be similar.

#Preliminary Classification Tree
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1) # 70% training and 30% test (this is the default values)
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()# using default criterion(gini)
# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)
```

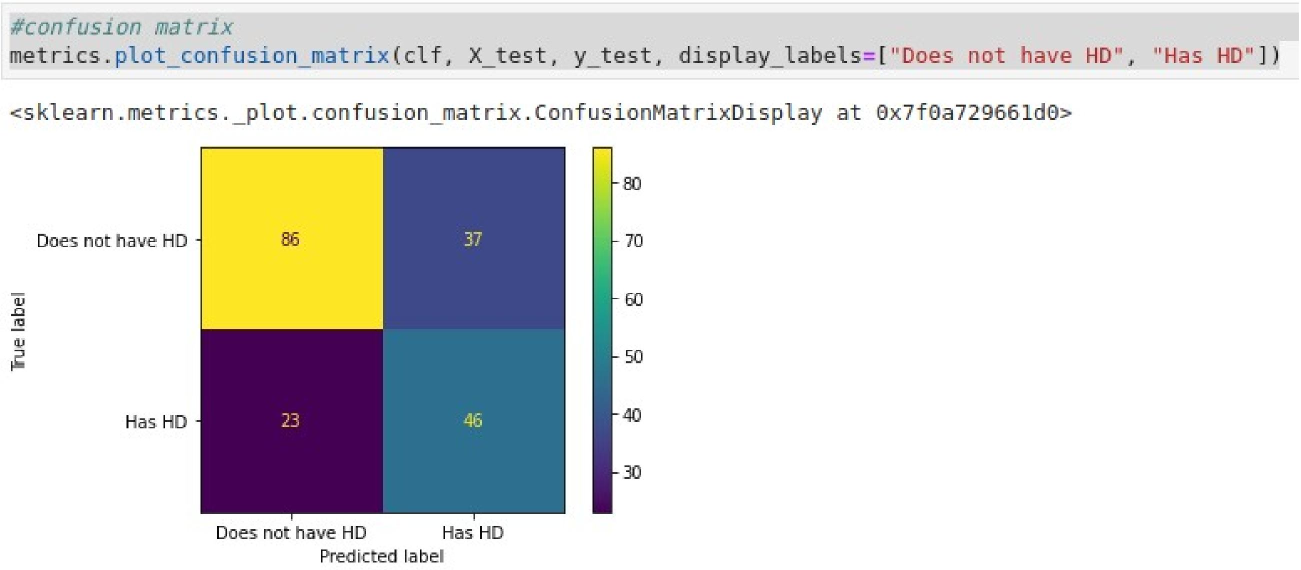
Results



```
#plot the tree
plt.figure(figsize=(15, 7.5))
plot_tree(clf,
          filled = True,
          rounded = True,
          class_names=["No HD", "Yes HD"],
          feature_names=X.columns);
```



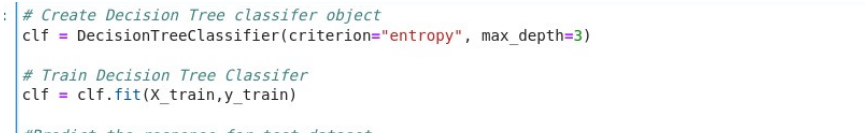
Results



```
#confusion matrix
metrics.plot_confusion_matrix(clf, X_test, y_test, display_labels=["Does not have HD", "Has HD"])
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f0a729661d0>
```

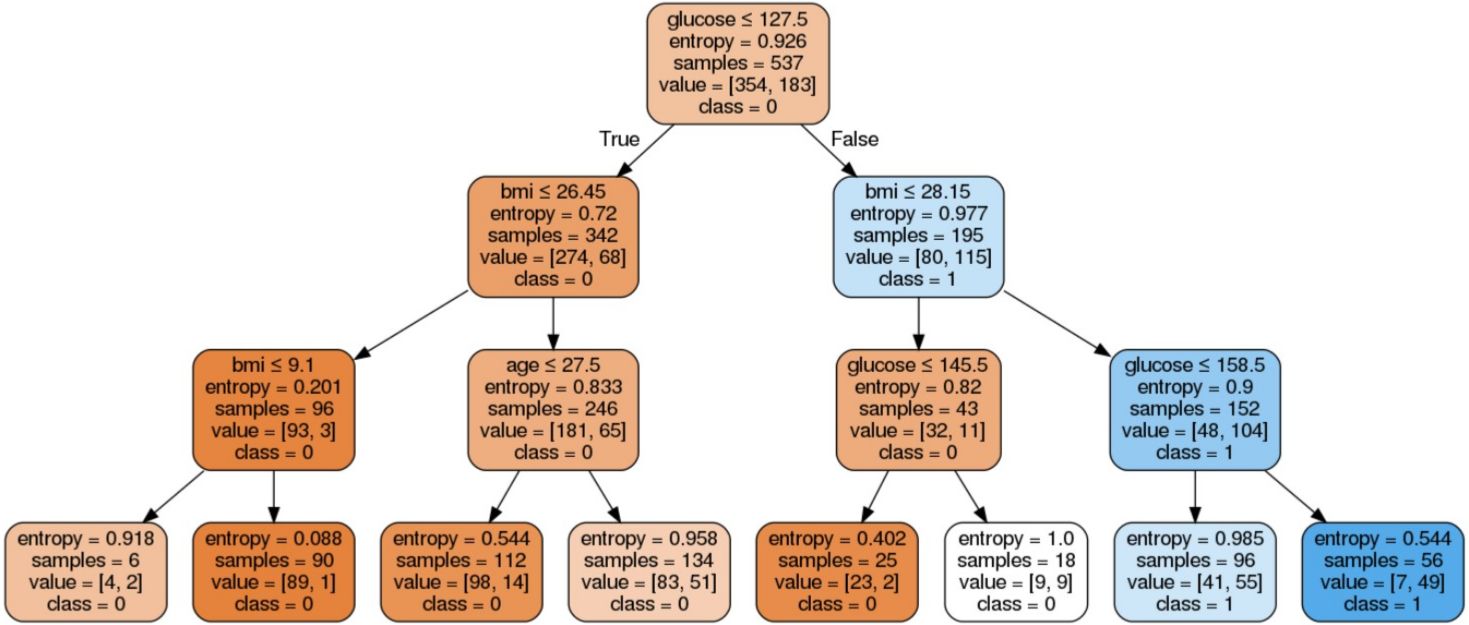
Results



```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the
y_pred = clf.

# Model Accur
print("Accura
```

```
Accuracy: 0.7
```

```
from sklearn.
from IPython.
from sklearn.
import pydotp
dot_data = St
export_graphv


graph = pydot
graph.write_p
Image(graph.c
```



Resources

Decision Trees:

- Data and code: https://www.datacamp.com/community/tutorials/decision-tree-classification-python
- In-depth explanation: https://www.youtube.com/watch?v=q90UDEgYqeI&t=91s
- Overview and Gini explained: https://www.youtube.com/watch?v=7VeUPuFGJHk&t=740s https://www.youtube.com/watch?v=_L39rN6gz7Y
- Scikit learn: https://scikit-learn.org/stable/modules/tree.html