# COMPASAnalysis

November 4, 2023

# 1 Analysis of COMPAS Score, Detecting Inaccuracies

## 1.1 Jack DeGesero

### 1.1.1 The data regarded predicts whether or not criminal defendants are likely to be reoffenders based on multiple attributes, in this report we will be examining sex, age (binned in three categories: <25, 25-45, >45), decile score, and if they did re-offend or not within a two year time frame (is a recidivist). All the data examined is sourced from Broward County, FL.

**This data is sourced from the Pro Publica, who initially led the report with all variables.** https://www.propublica.org/article/how-we-analyzed-the-compas-recidivism-algorithm

[1]:
```
import pandas as pd
import numpy as py
import statistics

import matplotlib.pyplot as plt #graphing

from sklearn import tree #for tree
from sklearn import model_selection #for partition into test and training data
from sklearn import preprocessing #to change attributes
from sklearn.metrics import accuracy_score #for checking model accuracy
from sklearn.metrics import classification_report, confusion_matrix #to show␣
↪confusion matrix
```

[2]:
```
#load the data
df1 = pd.read_csv('compas_small.csv')
df2 = pd.read_csv('compas_small_Ca.csv')
df3 = pd.read_csv('compas_small_AfAm.csv')

#get all relevant columns, and class attribute (is_recid)
df1 = df1[['sex', 'age_cat', 'decile_score', 'is_recid']]
df2 = df2[['sex', 'age_cat', 'decile_score', 'is_recid']]
df3 = df3[['sex', 'age_cat', 'decile_score', 'is_recid']]
```

[3]:
```
df1
```

[3]:
```
         sex           age_cat  decile_score is_recid
0       Male  Greater than 45             1       no
1       Male           25 - 45             3      yes
2       Male      Less than 25             4      yes
3       Male      Less than 25             8       no
4       Male           25 - 45             1       no
…       …              …          …       …
7209    Male      Less than 25             7       no
7210    Male      Less than 25             3       no
7211    Male  Greater than 45             1       no
7212  Female           25 - 45             2       no
7213  Female      Less than 25             4      yes

[7214 rows x 4 columns]
```

[4]:
```
df2
```

[4]:
```
         sex       age_cat  decile_score is_recid
0       Male       25 - 45             6      yes
1     Female       25 - 45             1       no
2       Male  Less than 25             3      yes
3       Male       25 - 45             4       no
4     Female       25 - 45             1       no
…       …          …          …       …
2449    Male       25 - 45             2       no
2450  Female       25 - 45             1      yes
2451    Male  Less than 25             8       no
2452    Male  Less than 25            10      yes
2453    Male  Less than 25             6      yes

[2454 rows x 4 columns]
```

[5]:
```
df3
```

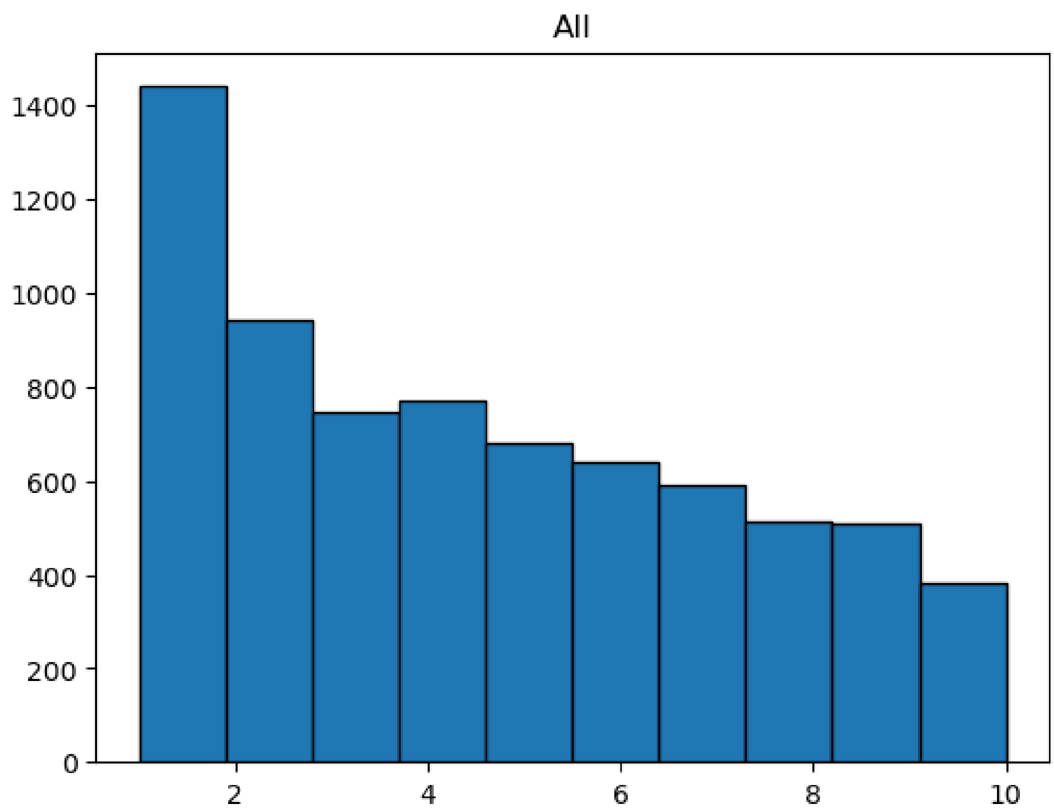
[5]:
```
         sex       age_cat  decile_score is_recid
0       Male       25 - 45             3      yes
1       Male  Less than 25             4      yes
2       Male  Less than 25             8       no
3       Male  Less than 25             6      yes
4       Male       25 - 45             4       no
…       …          …          …       …
3691    Male       25 - 45             2      yes
3692    Male  Less than 25             9       no
3693    Male  Less than 25             7       no
3694    Male  Less than 25             3       no
3695  Female       25 - 45             2       no
```

```
[3696 rows x 4 columns]
```

[6]:
```
#Check if any na values are present in each data set
any([df1['decile_score'].isna().any(),df2['decile_score'].isna().
 ↪any(),df3['decile_score'].isna().any()])
```

[6]:
```
False
```

[7]:
```
plt.hist(df1['decile_score'], ec='black')
plt.title("All")
plt.show()
```

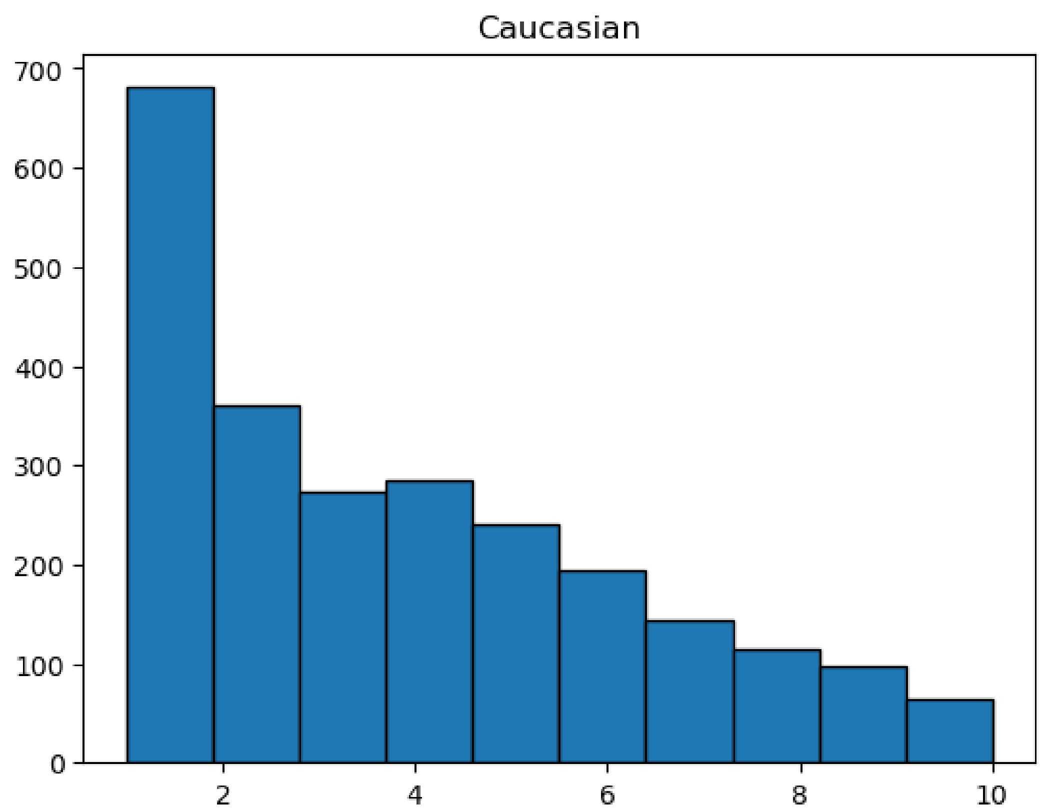



[8]:
```
print("Median of All: " + str(df1['decile_score'].median()) + ", Mode of All: "␣
 ↪+str(df1['decile_score'].mode().values[0]))
```

```
Median of All: 4.0, Mode of All: 1
```

### 1.1.2 From the figure above, we can see the median for decile score is greater than the mean indicating it is positively skewed

[9]:
```
plt.hist(df2['decile_score'], ec='black')
plt.title("Caucasian")
plt.show()
```

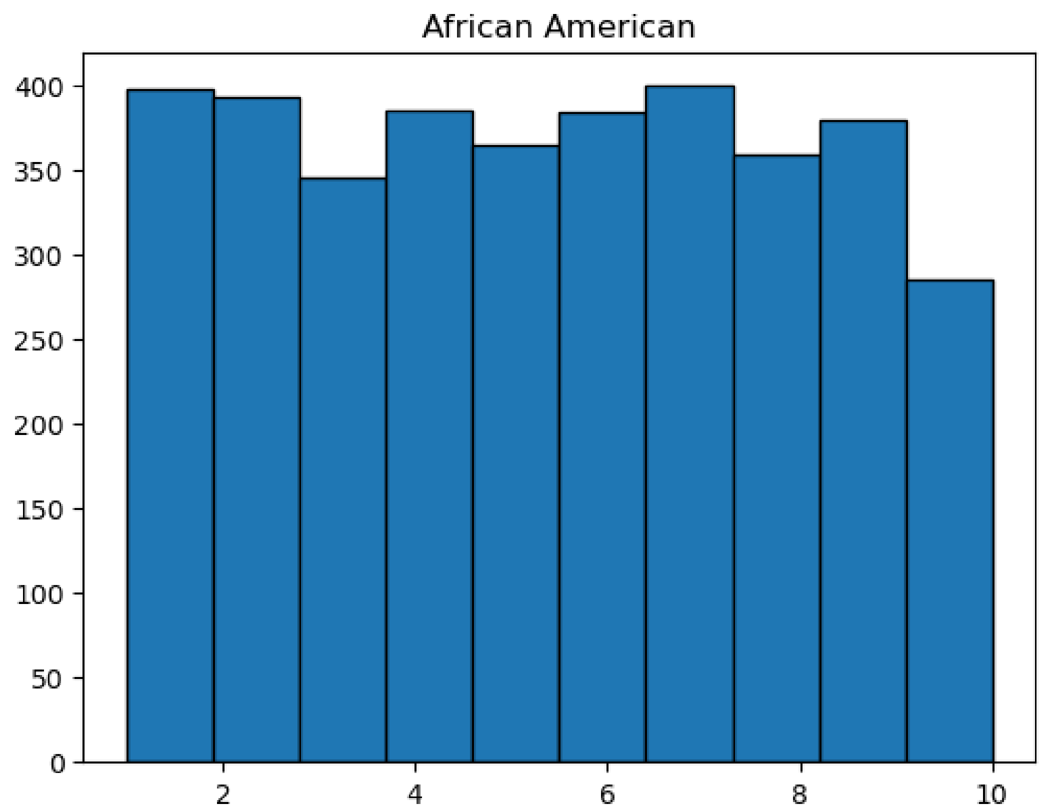



[10]:
```
print("Median of Caucasians: " + str(df2['decile_score'].median()) + ", Mode of␣
↪Caucasians: "+ str(df2['decile_score'].mode().values[0]))
```

```
Median of Caucasians: 3.0, Mode of Caucasians: 1
```

### 1.1.3 From this figure, we can see the histogram for Caucasians is also positively skewed

[11]:
```
plt.hist(df3['decile_score'], ec='black')
plt.title("African American")
plt.show()
```




[12]:
```
print("Median of African Americans: " + str(df3['decile_score'].median()) + ",␣
↪Mode of African Americans: "+ str(df1['decile_score'].mode().values[0]))
```

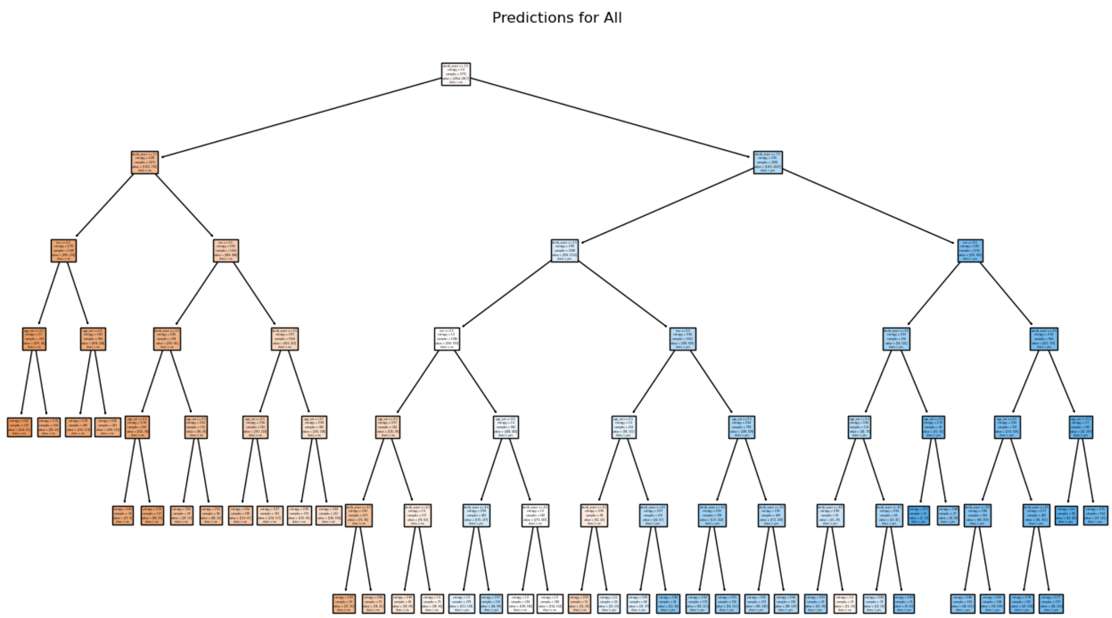
```
Median of African Americans: 5.0, Mode of African Americans: 1
```

#### 1.1.4 Finally, African Americans have a slight positive skew; however, the median is higher than both the latter graphs indicating its more evenly skewed (ie more entries with higher decile scores)

[13]:
```
#Preprocess some attributes to make scikit more digestible
df1['sex'] = preprocessing.LabelEncoder().fit(df1['sex']).transform(df1['sex'])
df1['age_cat'] = preprocessing.OneHotEncoder(sparse=False).
↪fit_transform(df1['age_cat'].values.reshape(-1, 1))
```

[14]:
```
#Make Trained Decision Tree for 'All' (df1)

#Grab attributes and class attribute
allAtr = df1[['sex', 'age_cat', 'decile_score']]
classAtr = df1['is_recid']
```

```
#Partition 20% of data to be tested, map to xtr-Training Attributes, xt-Test␣
↪Attributes, ytr-Class Training Attributes, yt-Class Test Attributes (used␣
↪for accuracy)
xtr, xt, ytr, yt = model_selection.train_test_split(allAtr, classAtr,␣
↪test_size=0.2, random_state=42)

#instantiates tree object
AllTree = tree.DecisionTreeClassifier(criterion="entropy")

#make the tree
AllTree.fit(xtr, ytr)

#predict based on test data
prediction = AllTree.predict(xt)

#plot data
plt.figure(figsize=(16, 9))
tree.plot_tree(AllTree, feature_names=allAtr.columns, class_names=['no',␣
↪'yes'], filled=True, impurity=True, precision=2)
plt.title('Predictions for All')
plt.savefig('decision_tree.png', dpi=300) #very low res in out[]:, see picture
plt.show()
```




[15]:
```
importances = AllTree.feature_importances_ #get gain of each attribute␣
↪examined, put in list
```

```
print("Gain:")

# Print the attributes in the order of importance
for i in xtr.columns:
    print(f'Attribute: {i}, Importance: {importances[xtr.columns.get_loc(i)]:.
 ↪2f}')
```

```
Gain:
Attribute: sex, Importance: 0.05
Attribute: age_cat, Importance: 0.03
Attribute: decile_score, Importance: 0.92
```

[16]:
```
#Run the test

print("Confusion matrix: \n" + str(confusion_matrix(yt,prediction)) +␣
 ↪"\nTP,FP\nFN,TN")
```

```
Confusion matrix:
[[570 219]
 [285 369]]
TP,FP
FN,TN
```

[17]:
```
print("Accuracy for All: " + str(accuracy_score(yt,prediction)))
```

```
Accuracy for All: 0.6507276507276507
```

### 1.1.5 The main model is accurate about 65% of the time. False positives occur 27% of the time while false negatives occur 43% of the time.

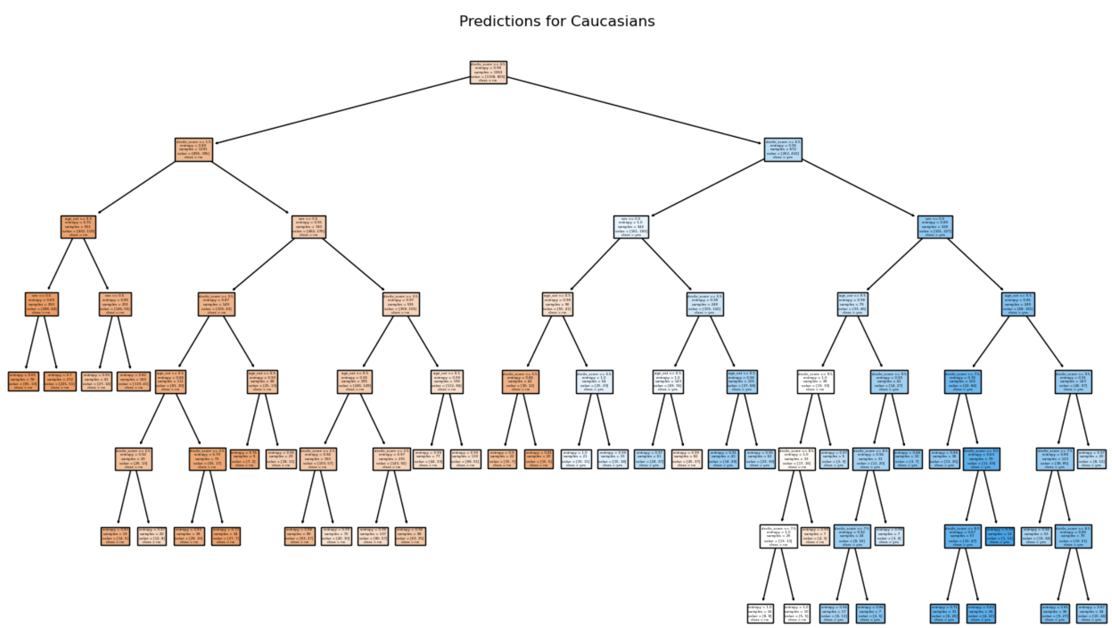
### 1.1.6 I ran the same data through the software Weka and got an accuracy of about 68%, very interesting to see how some models may classify more accurately than other models! Really goes to show why most contemporary models are proprietary.

### 1.1.7 Now, we repeat the same analysis for Caucasians and African Americans. Find Caucasian analysis below:

[18]:
```
#Preprocess some attributes to make scikit more digestible
df2['sex'] = preprocessing.LabelEncoder().fit(df2['sex']).transform(df2['sex'])
df2['age_cat'] = preprocessing.OneHotEncoder(sparse=False).
 ↪fit_transform(df2['age_cat'].values.reshape(-1, 1))
```

[19]:
```
#Make Trained Decision Tree for 'Caucasians' (df2)

#Grab attributes and class attribute
allAtr = df2[['sex', 'age_cat', 'decile_score']]
classAtr = df2['is_recid']
```

```
#Partition 20% of data to be tested, map to xtr-Training Attributes, xt-Test Attributes, ytr-Class Training Attributes, yt-Class Test Attributes (used for accuracy)
xtr, xt, ytr, yt = model_selection.train_test_split(allAtr, classAtr, test_size=0.2, random_state=42)

#instantiates tree object
AllTree = tree.DecisionTreeClassifier(criterion="entropy")

#make the tree
AllTree.fit(xtr, ytr)

#predict based on test data
prediction = AllTree.predict(xt)

#plot data
plt.figure(figsize=(16, 9))
tree.plot_tree(AllTree, feature_names=allAtr.columns, class_names=['no', 'yes'], filled=True, impurity=True, precision=2)
plt.title('Predictions for Caucasians')
plt.savefig('decision_tree_caucasian.png', dpi=300) #very low res in out[]:, see picture
plt.show()
```



```
[20]: importances = AllTree.feature_importances_ #get gain of each attribute␣
       ↪examined, put in list

      print("Gain:")

      # Print the attributes in the order of importance
      for i in xtr.columns:
          print(f'Attribute: {i}, Importance: {importances[xtr.columns.get_loc(i)]:.
       ↪2f}')
```

```
Gain:
Attribute: sex, Importance: 0.07
Attribute: age_cat, Importance: 0.09
Attribute: decile_score, Importance: 0.85
```

```
[21]: #Run the test

      print("Confusion matrix: \n" + str(confusion_matrix(yt,prediction)) +␣
       ↪"\nTP,FP\nFN,TN")
```

```
Confusion matrix:
[[225  46]
 [125  95]]
TP,FP
FN,TN
```

```
[22]: print("Accuracy for Caucasians: " + str(accuracy_score(yt,prediction)))
```

```
Accuracy for Caucasians: 0.6517311608961304
```

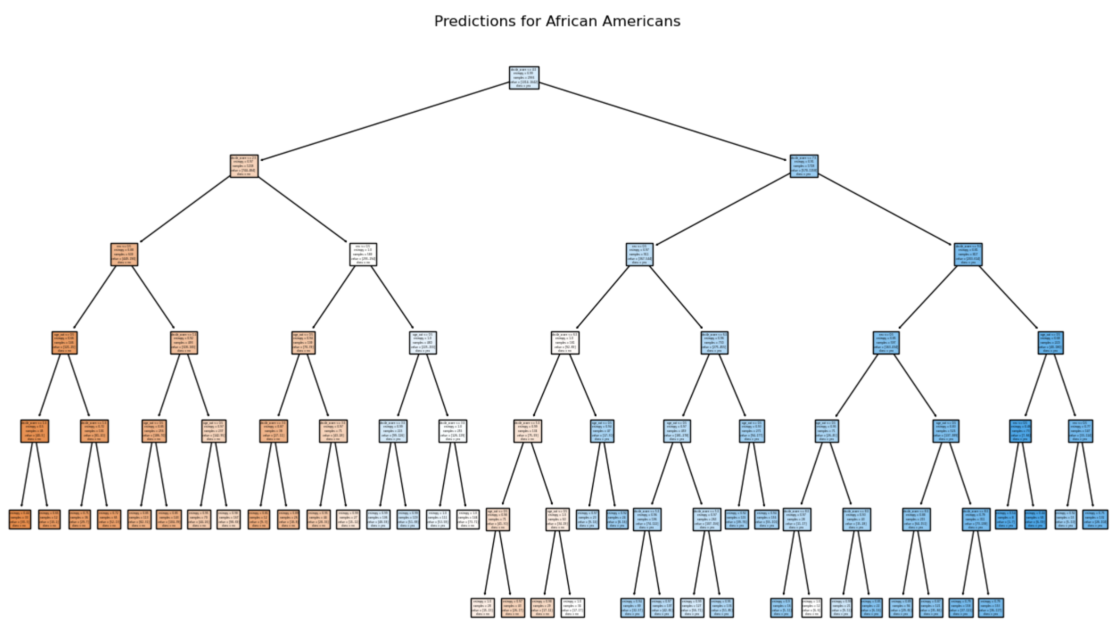
### 1.1.8 The model made for caucasians is slightly more accurate, by approximately 0.1%. False positives occurred 16% of the time while false negatives occurred 56% of the time.

### 1.1.9 Find African Amercian analysis below:

```
[23]: #Preprocess some attributes to make scikit more digestible
      df3['sex'] = preprocessing.LabelEncoder().fit(df3['sex']).transform(df3['sex'])
      df3['age_cat'] = preprocessing.OneHotEncoder(sparse=False).
       ↪fit_transform(df3['age_cat'].values.reshape(-1, 1))
```

```
[24]: #Make Trained Decision Tree for 'African Americans' (df3)

      #Grab attributes and class attribute
      allAtr = df3[['sex', 'age_cat', 'decile_score']]
      classAtr = df3['is_recid']
```

```python
#Partition 20% of data to be tested, map to xtr-Training Attributes, xt-Test Attributes, ytr-Class Training Attributes, yt-Class Test Attributes (used for accuracy)
xtr, xt, ytr, yt = model_selection.train_test_split(allAtr, classAtr, test_size=0.2, random_state=42)

#instantiates tree object
AllTree = tree.DecisionTreeClassifier(criterion="entropy")

#make the tree
AllTree.fit(xtr, ytr)

#predict based on test data
prediction = AllTree.predict(xt)

#plot data
plt.figure(figsize=(16, 9))
tree.plot_tree(AllTree, feature_names=allAtr.columns, class_names=['no', 'yes'], filled=True, impurity=True, precision=2)
plt.title('Predictions for African Americans')
plt.savefig('decision_tree_africanamerican.png', dpi=300) #very low res in out[]:, see picture
plt.show()
```



[25]:
```
importances = AllTree.feature_importances_ #get gain of each attribute␣
↪examined, put in list

print("Gain:")

# Print the attributes in the order of importance
for i in xtr.columns:
    print(f'Attribute: {i}, Importance: {importances[xtr.columns.get_loc(i)]:.
↪2f}')
```

```
Gain:
Attribute: sex, Importance: 0.10
Attribute: age_cat, Importance: 0.04
Attribute: decile_score, Importance: 0.86
```

[26]:
```
#Run the test

print("Confusion matrix: \n" + str(confusion_matrix(yt,prediction)) +␣
↪"\nTP,FP\nFN,TN")
```

```
Confusion matrix:
[[155 191]
 [ 85 309]]
TP,FP
FN,TN
```

[27]:
```
print("Accuracy for African Americans: " + str(accuracy_score(yt,prediction)))
```

```
Accuracy for African Americans: 0.6270270270270271
```

#### 1.1.10 The model for African Americans is only accurate about 62% of the time, 2.4% less accurate than the caucasian model. False positives occurred 55% of the time, while false negatives occurred 21% of the time. The disparity in the False Positives goes to show why this prediction should not be the sole deciding factor in determining recidivism.

#### 1.1.11 We can also check if we add more attributes (for the decision tree) if our accuracy will increase. Let's attempt this with our 'all' data, but this time we will also consider juvenile misdemeanors.

[28]:
```
#read the data
df4 = pd.read_csv('compas_small.csv')

df4 = df4[['sex', 'age_cat', 'decile_score', 'juv_misd_count', 'is_recid']]
```

[29]:
```
df4
```

[29]:
```
       sex          age_cat  decile_score  juv_misd_count is_recid
0     Male  Greater than 45             1               0       no
```

```
1       Male          25 - 45             3               0      yes
2       Male     Less than 25             4               0      yes
3       Male     Less than 25             8               1       no
4       Male          25 - 45             1               0       no
…       …             …         …            …       …
7209    Male     Less than 25             7               0       no
7210    Male     Less than 25             3               0       no
7211    Male  Greater than 45             1               0       no
7212  Female          25 - 45             2               0       no
7213  Female     Less than 25             4               0      yes

[7214 rows x 5 columns]
```

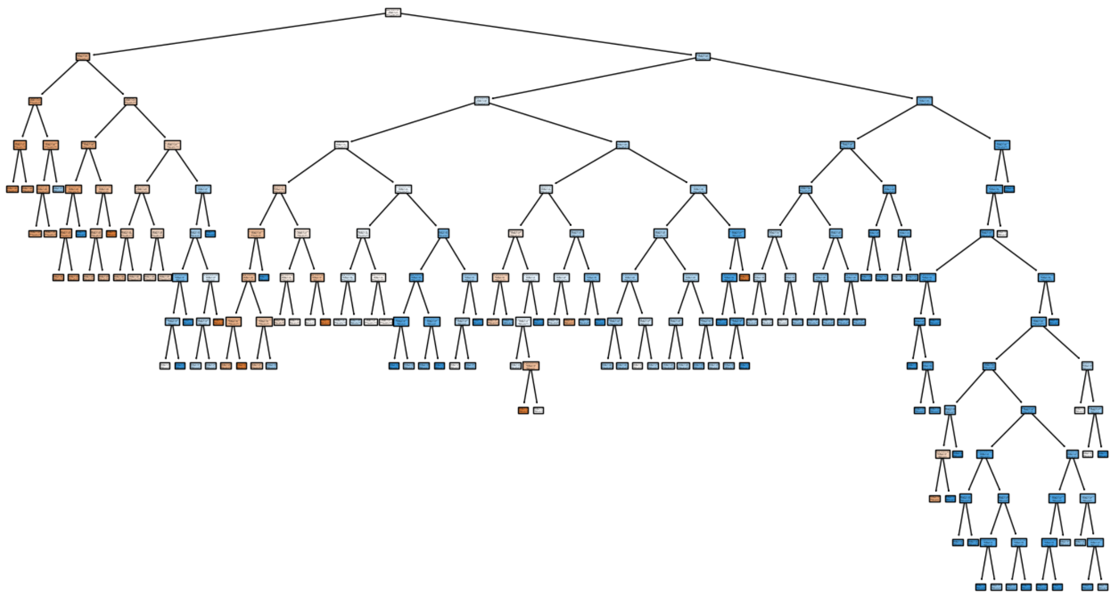
[30]:

```
#Preprocess some attributes to make scikit more digestible
df4['sex'] = preprocessing.LabelEncoder().fit(df4['sex']).transform(df4['sex'])
df4['age_cat'] = preprocessing.OneHotEncoder(sparse=False).
 ↪fit_transform(df4['age_cat'].values.reshape(-1, 1))
```

[31]:

```
#Make Trained Decision Tree for 'All with # of Juvenile Misdemeanors' (df4)

#Grab attributes and class attribute
allAtr = df4[['sex', 'age_cat', 'decile_score', 'juv_misd_count']]
classAtr = df4['is_recid']

#Partition 20% of data to be tested, map to xtr-Training Attributes, xt-Test␣
 ↪Attributes, ytr-Class Training Attributes, yt-Class Test Attributes (used␣
 ↪for accuracy)
xtr, xt, ytr, yt = model_selection.train_test_split(allAtr, classAtr,␣
 ↪test_size=0.2, random_state=42)

#instantiates tree object
AllTree = tree.DecisionTreeClassifier(criterion="entropy")

#make the tree
AllTree.fit(xtr, ytr)

#predict based on test data
prediction = AllTree.predict(xt)

#plot data
plt.figure(figsize=(16, 9))
tree.plot_tree(AllTree, feature_names=allAtr.columns, class_names=['no',␣
 ↪'yes'], filled=True, impurity=True, precision=2)
plt.title('Predictions for All with # of Juvenile Misdemeanors')
plt.savefig('decision_tree_juvc.png', dpi=300) #very low res in out[]:, see␣
 ↪picture
plt.show()
```

Predictions for All with # of Juvenile Misdemeanors



[32]:
```
importances = AllTree.feature_importances_ #get gain of each attribute␣
↪examined, put in list

# Print the attributes in the order of importance
for i in xtr.columns:
    print(f'Attribute: {i}, Importance: {importances[xtr.columns.get_loc(i)]:.
↪2f}')
```

```
Attribute: sex, Importance: 0.05
Attribute: age_cat, Importance: 0.04
Attribute: decile_score, Importance: 0.83
Attribute: juv_misd_count, Importance: 0.08
```

[33]:
```
#Run the test

print("Confusion matrix: \n" + str(confusion_matrix(yt,prediction)) +␣
↪"\nTP,FP\nFN,TN")
```

```
Confusion matrix:
[[564 225]
 [280 374]]
TP,FP
FN,TN
```

[34]:
```
print("Accuracy for All (after juv count added): " +␣
↪str(accuracy_score(yt,prediction)))
```

```
Accuracy for All (after juv count added): 0.65003465003465
```

#### 1.1.12 After we added in the juvenile misdemeanor count, the model decreased in accuracy. This is why attribute selection is vital to ensure there are no redundancies examined in our data. A good way to avoid redundancy in our attributes is to check correlation. We can see below that, although weak, juvenile misdemeanor count is positively correlated with decile score. It's the most notable correlation as compared to the other attributes.

[35]:
```
corref = py.corrcoef(df4['decile_score'].values,df4['juv_misd_count'].
 ↪values)[0, 1]

corref
```

[35]:
```
0.21592745594052032
```