DMA Fall '21

In [1]:

```
NAME = "Tyler Freund"
COLLABORATORS = ""
```

Lab 3: Decision Trees

Please read the following instructions very carefully

Working on the assignment / FAQs

- **Always use the seed/random_state as *42* wherever applicable** (This is to ensure repeatability in answers, across students and coding environments)
- Questions can be either autograded and manually graded.
- The type of question and the points they carry are indicated in each question cell
- An autograded question has 3 cells
  - **Question cell** : Read only cell containing the question
  - **Code Cell** : This is where you write the code
  - **Grading cell** : This is where the grading occurs, and **you are required not to edit this cell**
- Manually graded questions only have the question and code cells. **All manually graded questions are explicitly stated**
- To avoid any ambiguity, each question also specifies what *value* must be set. Note that these are dummy values and not the answers
- If an autograded question has multiple answers (due to differences in handling NaNs, zeros etc.), all answers will be considered.
- Most assignments have bonus questions for extra credit, do try them out!
- You can delete the `raise NotImplementedError()` for all questions.
- **Submitting the assignment** : Download the '.ipynb' file from Colab and upload it to bcourses. Do not delete any outputs from cells before submitting.
- That's about it. Happy coding!

About the dataset

This assignment uses a dataset obtained from the JSE Data Archive that contains biological and self-reported activity traits of a sample of college students at a single university uploaded in 2013. The study associated with these data focused on exploring if a correspondence exists between eye color and and other traits. You will be using gender as the target/label in this lab.

FEATURE DESCRIPTIONS:

- Color (Blue, Brown, Green, Hazel, Other)
- Age (in years)
- YearinSchool (First, Second, Third, Fourth, Other)
- Height (in inches)
- Miles (distance from home town of student to Ames, IA)
- Brothers (number of brothers)
- Sisters (number of sisters)
- CompTime (number of hours spent on computer per week)
- Exercise (whether the student exercises Yes or No)
- ExerTime (number of hours spent exercising per week)
- MusicCDs (number of music CDs student owns)

- PlayGames (number of hours spent playing games per week)
- WatchTV (number of hours spent watching TV per week

Background Information on the dataset: http://jse.amstat.org/v21n2/froelich/eyecolorgender.txt

In [2]:

```
from collections import Counter, defaultdict
from itertools import combinations
import pandas as pd
import numpy as np
import operator
import math
import itertools
from sklearn.feature_extraction import DictVectorizer
from sklearn import preprocessing, tree
import matplotlib.pyplot as plt


!wget -nc http://askoski.berkeley.edu/~zp/eye_color.csv
!ls
df = pd.read_csv('eye_color.csv')
# remove NA's and reset the index
df = df.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)
df = df.reset_index(drop=True)

df.head()
```

```
--2021-09-15 16:38:38--  http://askoski.berkeley.edu/~zp/eye_color.csv
Resolving askoski.berkeley.edu (askoski.berkeley.edu)... 169.229.192.179
Connecting to askoski.berkeley.edu (askoski.berkeley.edu)|169.229.192.179|:80... connecte
d.
HTTP request sent, awaiting response... 200 OK
Length: 101507 (99K) [text/csv]
Saving to: ‘eye_color.csv’

eye_color.csv       100%[===================>]  99.13K   126KB/s    in 0.8s

2021-09-15 16:38:41 (126 KB/s) - ‘eye_color.csv’ saved [101507/101507]

eye_color.csv  sample_data
```

Out[2]:

	gender	age	year	eyecolor	height	miles	brothers	sisters	computertime	exercise	exercisehours	musiccds	playgan
0	female	18	first	hazel	68.0	195.0	0	1	20.0	Yes	3.0	75.0	
1	male	20	third	brown	70.0	120.0	3	0	24.0	No	0.0	50.0	
2	female	18	first	green	67.0	200.0	0	1	35.0	Yes	3.0	53.0	
3	male	23	fourth	hazel	74.0	140.0	1	1	5.0	Yes	25.0	50.0	
4	female	19	second	blue	62.0	60.0	0	1	5.0	Yes	4.0	30.0	

Question 1 (0.5 points, autograded): How many males and females exist in the dataset?

In [3]:

```
df1 = df.copy()
df1 = df1.groupby("gender").count()
df1
#raise NotImplementedError()
```

Out[3]:

	age	year	eyecolor	height	miles	brothers	sisters	computertime	exercise	exercisehours	musiccds	playgames

gender	age	year	eyecolor	height	miles	brothers	sisters	computertime	exercise	exercisehours	musiccds	playgames
female	1078	1078	1078	1078	1078	1078	1078	1078	1078	1078	1078	1078
male	910	910	910	910	910	910	910	910	910	910	910	910

In [4]:

```
#The value set in the variables must be integers
num_males = 910 #Replace 0 with the actual value
num_females = 1078 #Replace 0 with the actual value

# YOUR CODE HERE
#raise NotImplementedError()
```

In [5]:

```
#This is an autograded cell, do not edit
print(num_males, num_females)
```

```
910 1078
```

Question 2 (0.5 points, autograded): What is the Gini Index of this dataset, using males and females as the target classes?

In [6]:

```
m_prop = num_males/(num_males+num_females)
f_prop = num_females/(num_males+num_females)
gini_ = 1 - (m_prop**2 + f_prop**2)
gini_
#raise NotImplementedError()
```

Out[6]:

```
0.4964292799047807
```

In [7]:

```
#The value set in the variable must be float
gini_index = 0.4964292799047807 #Replace 0 with the actual value / formula

#raise NotImplementedError()
```

In [8]:

```
#This is an autograded cell, do not edit
print(gini_index)
```

```
0.4964292799047807
```

Best Split of a numeric feature

Question 3 (1.5 points, autograded): What is the best split point of the 'height' feature? (Still using males and females as the target classes, assuming a binary split)

Recall that, to calculate the best split of this numeric field, you'll need to order your data by 'height', then consider the midpoint between each pair of consecutive heights as a potential split point, then calculate the Gini Index for that partitioning. You'll want to keep track of the best split point and its Gini Index (remember that you are trying to minimize the Gini Index).

In [9]:

```
df3 = df.copy()
```

```
df3 = df3.sort_values("height")

rangg = np.arange(0, 1987)
heights = np.array(df3.iloc[:,4])
splits = []
for i in rangg:
  splits.append((heights[i]+heights[i+1])/2)

newrangg = np.arange(0, 1987)
gini_list = []
for j in newrangg:
  m_below = len(df3.loc[(df3['height'] <= splits[j]) & (df3['gender'] == 'male')])
  m_above = 910-m_below
  f_below = len(df3.loc[(df3['height'] <= splits[j]) & (df3['gender'] == 'female')])
  f_above = 1078-f_below

  mf_above = m_above + f_above
  mf_below = m_below + f_below

  gini_above = 1- ((m_above / (m_above + f_above))**2 + (f_above / (m_above + f_above))*
*2)
  gini_below = 1- ((m_below / (m_below + f_below))**2 + (f_below / (m_below + f_below))*
*2)

  weighted_gini = (mf_above/len(df))*gini_above + ((mf_below/len(df))*gini_below)
  gini_list.append(weighted_gini)

  if weighted_gini <= min(gini_list):
    best_point = splits[j]
    index_of_best_point = j

print(index_of_best_point)

#raise NotImplementedError()
```

```
1046
```

In [10]:

```
#The value set in the variable must be float
best_split_point = best_point #Replace 0 with the actual value


#raise NotImplementedError()
```

In [11]:

```
#This is an autograded cell, do not edit
print(best_split_point)
```

```
68.5
```

Question 4 (0.5 points, autograded): What is the Gini index of the best split point of the 'height' feature? (Still using males and females as the target classes, assuming a binary split)

In [12]:

```
best_gini = min(gini_list)
#raise NotImplementedError()
```

In [13]:

```
#The value set in the variable must be float
gini_of_best_split_point = best_gini #Replace 0 with the actual value

# YOUR CODE HERE
#raise NotImplementedError()
```

In [14]:

```
#This is an autograded cell, do not edit
print(gini_of_best_split_point)
```

```
0.2655288120702919
```

Question 5 (0.5 points, autograded): How much does this partitioning reduce the Gini Index over the Gini index of the overall dataset?

In [15]:

```
q5 = gini_index - best_gini
#raise NotImplementedError()
```

In [16]:

```
#The value set in the variable must be float
gini_difference = q5 #Replace 0 with the actual value

#raise NotImplementedError()
```

In [17]:

```
#This is an autograded cell, do not edit
print(gini_difference)
```

```
0.2309004678344888
```

Question 6 (0.5 points, autograded): How many 'female' and 'male' rows are shorter than the best height split point?

In [18]:

```
mm = df3.loc[(df3['height'] <= best_split_point) & (df3['gender'] == 'male')]
 ff = df3.loc[(df3['height'] <= best_split_point) & (df3['gender'] == 'female')]
#raise NotImplementedError()
```

In [19]:

```
#The value set in the variable must be integer
female_rows_below = len(ff) #Replace 0 with the actual value
male_rows_below = len(mm) #Replace 0 with the actual value
# YOUR CODE HERE
#raise NotImplementedError()
```

In [20]:

```
#This is an autograded cell, do not edit
print(female_rows_below, male_rows_below)
```

```
905 142
```

Question 7 (0.5 points, autograded): How many 'female' and 'male' rows are taller than the best height split point?

In [21]:

```
mm_q7 = df3.loc[(df3['height'] >= best_split_point) & (df3['gender'] == 'male')]
ff_q7 = df3.loc[(df3['height'] >= best_split_point) & (df3['gender'] == 'female')]
#raise NotImplementedError()
```

In [22]:

```
#The value set in the variable must be integer
female_rows_above = len(ff_q7) #Replace 0 with the actual value
male_rows_above = len(mm_q7) #Replace 0 with the actual value

#raise NotImplementedError()
```

In [23]:

```
#This is an autograded cell, do not edit
print(female_rows_above, male_rows_above)
```

```
173 768
```

Best Split of a Categorial Variable

Question 8 (0.5 points, autograded): How many possible splits are there of the eyecolor feature? (Assuming binary split)

Python tip: the combinations function of the itertools module allows you to enumerate combinations of a list. You might want to Google 'power set'.

In [24]:

```
df8 = df.copy()
df8 = df8.groupby("eyecolor").count()
colors = ["blue","brown","green","hazel","other"]
powerset = list(itertools.chain.from_iterable(itertools.combinations(colors, r) for r in range(len(colors)+1)))
powerset
#raise NotImplementedError()
```

Out[24]:

```
[(),
 ('blue',),
 ('brown',),
 ('green',),
 ('hazel',),
 ('other',),
 ('blue', 'brown'),
 ('blue', 'green'),
 ('blue', 'hazel'),
 ('blue', 'other'),
 ('brown', 'green'),
 ('brown', 'hazel'),
 ('brown', 'other'),
 ('green', 'hazel'),
 ('green', 'other'),
 ('hazel', 'other'),
 ('blue', 'brown', 'green'),
 ('blue', 'brown', 'hazel'),
 ('blue', 'brown', 'other'),
 ('blue', 'green', 'hazel'),
 ('blue', 'green', 'other'),
 ('blue', 'hazel', 'other'),
 ('brown', 'green', 'hazel'),
 ('brown', 'green', 'other'),
 ('brown', 'hazel', 'other'),
 ('green', 'hazel', 'other'),
 ('blue', 'brown', 'green', 'hazel'),
 ('blue', 'brown', 'green', 'other'),
 ('blue', 'brown', 'hazel', 'other'),
 ('blue', 'green', 'hazel', 'other'),
 ('brown', 'green', 'hazel', 'other'),
 ('blue', 'brown', 'green', 'hazel', 'other')]
```

In [25]:

```
#The value set in the variable must be integer
num_of_splits = 15 #Replace 0 with the actual value

#raise NotImplementedError()
```

In [26]:

```
#This is an autograded cell, do not edit
print(num_of_splits)
```

15

Question 9 (1 points, autograded): Which split of eyecolor best splits the female and male rows, as measured by the Gini Index?

In [27]:

```
df9 = df.copy()

color_splits_1 = ["blue", "brown", "green", "hazel", "other"]
color_splits_2 = [["blue","brown"], ["blue","green"], ["blue","hazel"], ["blue","other"]
, ["brown","green"], ["brown","hazel"], ["brown","other"], ["green","hazel"], ["green","
other"], ["hazel","other"]]

gini_list_q9 = []

for colo in color_splits_1:
  m_color = len(df9.loc[(df9['eyecolor'] == colo) & (df9['gender'] == 'male')])
  m_not = 910-m_color
  f_color = len(df3.loc[(df3['eyecolor'] == colo) & (df3['gender'] == 'female')])
  f_not = 1078-f_color

  mf_color = m_color + f_color
  mf_not = m_not + f_not

  gini_color = 1- ((m_color / (m_color + f_color))**2 + (f_color / (m_color + f_color))*
*2)
  gini_not = 1- ((m_not / (m_not + f_not))**2 + (f_not / (m_not + f_not))**2)

  weighted_gini = (mf_color/len(df))*gini_color + (mf_not/len(df))*gini_not

  gini_list_q9.append(weighted_gini)

  if weighted_gini <= min(gini_list_q9):
      best_color_split = colo

for elem in color_splits_2:
  m_color = len(df9.loc[((df9['eyecolor'] == elem[0]) | (df9['eyecolor'] == elem[1])) &
(df9['gender'] == 'male')])
  m_not = 910-m_color
  f_color = len(df9.loc[((df9['eyecolor'] == elem[0]) | (df9['eyecolor'] == elem[1])) &
(df9['gender'] == 'female')])
  f_not = 1078-f_color

  mf_color = m_color + f_color
  mf_not = m_not + f_not

  gini_color = 1- ((m_color / (m_color + f_color))**2 + (f_color / (m_color + f_color))*
*2)
  gini_not = 1- ((m_not / (m_not + f_not))**2 + (f_not / (m_not + f_not))**2)

  weighted_gini = (mf_color/len(df))*gini_color + (mf_not/len(df))*gini_not

  gini_list_q9.append(weighted_gini)

  if weighted_gini <= min(gini_list_q9):
```

```
    best_color_split = colo

print(best_color_split)
print(min(gini_list_q9))

#raise NotImplementedError()
```

```
green
0.4930915729509777
```

In [28]:

```
#The value set in the variable must be an array
colour_group_1 = ["green"] #Replace [] with the actual colours/values in the group
colour_group_2 = ["blue","brown","hazel","other"] #Replace [] with the actual colours/va
lues in the group

#raise NotImplementedError()
```

In [29]:

```
#This is an autograded cell, do not edit
print(colour_group_1, colour_group_2)
```

```
['green'] ['blue', 'brown', 'hazel', 'other']
```

Question 10 (0.5 points, autograded): What is the Gini Index of this best split?

In [30]:

```
gini_q10 = min(gini_list_q9)
#raise NotImplementedError()
```

In [31]:

```
#The value set in the variable must be float
gini_of_best_split_group = gini_q10 #Replace 0 with the actual value

#raise NotImplementedError()
```

In [32]:

```
#This is an autograded cell, do not edit
print(gini_of_best_split_group)
```

```
0.4930915729509777
```

Question 11 (0.5 points, autograded): How much does this partitioning reduce the Gini Index over the Gini index of the overall dataset?

In [33]:

```
q11 = gini_index - gini_of_best_split_group
#raise NotImplementedError()
```

In [34]:

```
#The value set in the variable must be float
gini_difference_2 = q11 #Replace 0 with the actual value

#raise NotImplementedError()
```

In [35]:

```
#This is an autograded cell, do not edit
```

```
print(gini_difference_2)
```

0.003337706953802977

Question 12 (1 points, autograded) : How many 'female' rows and 'male' rows are in your first partition? How many 'female' rows and 'male' rows are in your second partition?

In [36]:

```
part1_m = len(df9.loc[(df9['eyecolor'] == "green") & (df9['gender'] == 'male')])
part1_f = len(df9.loc[(df9['eyecolor'] == "green") & (df9['gender'] == 'female')])
part2_m = len(df9.loc[(df9['eyecolor'] != "green") & (df9['gender'] == 'male')])
part2_f = len(df9.loc[(df9['eyecolor'] != "green") & (df9['gender'] == 'female')])


#raise NotImplementedError()
```

In [36]:

```

```

In [37]:

```
#The value set in the variable must be integer, order doesn't matter
partition1_male = part1_m #Replace 0 with the actual value
partition1_female = part1_f #Replace 0 with the actual value
partition2_male = part2_m #Replace 0 with the actual value
partition2_female = part2_f #Replace 0 with the actual value


#raise NotImplementedError()
```

In [38]:

```
#This is an autograded cell, do not edit
print(partition1_male, partition1_female, partition2_male, partition2_female)
```

107 190 803 888

Training a decision tree

Question 13 (1 points, autograded): Using all of the features in the original dataframe read in at the top of this notebook, train a decision tree classifier that has a depth of three (not including the root node). What is the accuracy of this classifier on the training data?

Scikit-learn classifiers require class labels and features to be in numeric arrays. As such, you will need to turn your categorical features into numeric arrays using DictVectorizer. This is a helpful notebook for understanding how to do this: http://nbviewer.ipython.org/gist/sarguido/7423289. You can turn a pandas dataframe of features into a dictionary of the form needed by DictVectorizer by using df.to_dict('records'). Make sure you remove the class label first (in this case, gender). If you use the class label as a feature, your classifier will have a training accuracy of 100%! The example notebook link also shows how to turn your class labels into a numeric array using sklearn.preprocessing.LabelEncoder().

In [39]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

df13 = df.copy()
df13 = df.drop('gender', 1)
_dict_ = df13.to_dict('records')

vec = DictVectorizer()
```

```
vectored_array = vec.fit_transform(_dict_).toarray()

X = vectored_array

df13_with_gen = df.copy()
df13_with_gen['gender'].replace(0, 'female',inplace=True)
df13_with_gen['gender'].replace(1, 'male',inplace=True)

y = df13_with_gen.gender
#X = df13[["age","year","eyecolor","height","miles","brothers","sisters","computertime","exercise","exercisehours","musiccds","playgames","watchtv"]]

clf = DecisionTreeClassifier(max_depth=3)

clf = clf.fit(X, y)
# # raise NotImplementedError()
y_pred = clf.predict(X)

print(accuracy_score(y, y_pred))
```

```
0.8646881287726358
```

In [40]:

```
#The value set in the variable must be float
accuracy = 0.8646881287726358 #Replace 0 with the actual value


#raise NotImplementedError()
```

In [41]:

```
#This is an autograded cell, do not edit
print(accuracy)
```

```
0.8646881287726358
```

Question 14 (1 points, manually graded) : Using the following code snippet, visualize your decision tree. In your write-up, write down the interpretation of the rule at each node which is used to perform the splitting.

We provide two options to visualize decision trees. The first option uses `tree.plot_tree` and the other uses an external tool called `GraphViz`. You can use either of the two options. `tree.plot_tree` is the recommended and easier option as it is a built-in function in `sklearn` and doesn't require any additional setup.

Uncomment the code, fill in the clf (classifier) and `feature_names` arguments. Executing the code will display the tree visualization in the output cell.

Note for users who want to install graphviz on their local machines (you don't need to do install graphviz if you're running the notebook Colab, which is the class' recommended way of doing assignments):

> **In order to install graphviz, you may need to download the tool from [this website](), and then pip3/conda install the python libraries you do not have. Mac users can use `brew install graphviz` instead of following the link, and linux users can do the same using their favourite package manager (for example, Ubuntu users can use `sudo apt-get install graphviz`, followed by the necessary pip3/conda installations.**

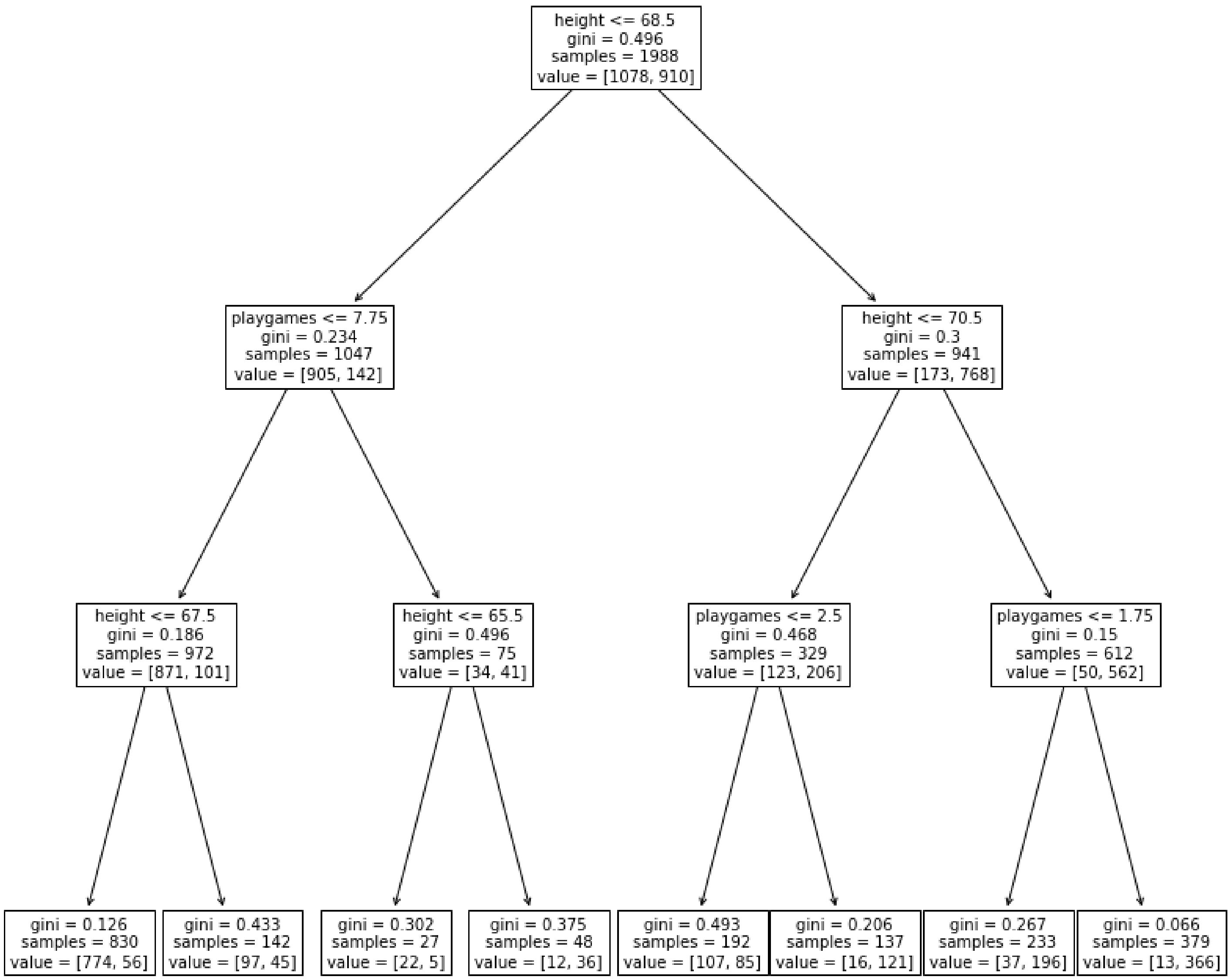
In [42]:

```
#Option 1 (Recommended Option) - Using `tree.plot_tree`
clf = DecisionTreeClassifier(max_depth=3)
clf = clf.fit(X, y)
#clf = your classifier
```

```
fig, ax = plt.subplots(figsize=(14, 14))

df14 = df.copy()
df14 = df.drop('gender', 1)
_dict_ = df14.to_dict('records')
vec = DictVectorizer(sparse=False)
vectored_array = vec.fit_transform(_dict_)

tree.plot_tree(clf, fontsize=10, feature_names = vec.feature_names_);
```



QUESTION 14 WRITE UP: My interpretation of the rule at each node used for splitting is that whatever split point yields the minimum gini value will be the chosen split point. For instance as we saw in earlier questions like q3 our best split point was at 68.5 for height because it yielded the minimum gini value. Similarly, we saw how splitting by eye color we also chose to use the partition that yielded the minimum gini value. We also see that as we follow the branches of the tree the gini values are decreasing (which is what we want for a strong model).

In [43]:

```
#Option 2 - Using GraphViz. Visualization is prettier, but additional setup may be requir
ed if running on your local machine (although no setup required on Colab)

from IPython.display import Image
import pydotplus
import pydot
```

```
from sklearn.externals.six import StringIO


#clf = your classifier
#dotfile = StringIO()
#tree.export_graphviz(clf, out_file=dotfile,
#                       feature_names=<Names of columns>,
#                           class_names=['Female', 'Male'],
#                           filled=True, rounded=True,
#                           special_characters=True)
#graph = pydotplus.graph_from_dot_data(dotfile.getvalue())
#Image(graph.create_png())


#Ignore the cell below, but do not delete it. It is used to grade the image output of thi
s cell.
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/externals/six.py:31: FutureWarning: The mo
dule is deprecated in version 0.21 and will be removed in version 0.23 since we've droppe
d support for Python 2.7. Please rely on the official version of six (https://pypi.org/pr
oject/six/).
  "(https://pypi.org/project/six/).", FutureWarning)
```

In [44]:

```
# YOUR CODE HERE
#raise NotImplementedError()
```

Bonus Question (2 points, auto graded)

For each of your leaf nodes, specify the percentage of 'female' rows in that node (out of the total number of rows at that node)

In [45]:

```
#The value set in the variable must be array
node1 = (774/830)*100
node2 = (97/142)*100
node3 = (22/27)*100
node4 = (12/48)*100
node5 = (107/192)*100
node6 = (16/137)*100
node7 = (37/233)*100
node8 = (13/379)*100

ratios = [node1,node2,node3,node4,node5,node6,node7,node8] #Replace 0 with the actual va
lue

# YOUR CODE HERE
#raise NotImplementedError()
```

In [46]:

```
#This is an autograded cell, do not edit
print(ratios)
```

```
[93.25301204819277, 68.30985915492957, 81.48148148148148, 25.0, 55.729166666666664, 11.67
8832116788321, 15.879828326180256, 3.430079155672823]
```

In [46]:

```

```