# Hw2_decision_trees_ensemble_learning

April 12, 2022

## 1 Data Sets

This assignment consists a number of implementation and result analysis questions. For these we will again consider the problem and data sets from Questions 2 and 3 in the first assignment where we had height, weight, age, and gender information. Assume the same datasets you generated for the first assignment. Make sure that you use the data you got using your own student ID. Results on data for other student ID numbers will not be

## 2 Decision Trees

1. Consider the problem from the previous assignments where we want to predict gender from information about height, weight, and age. Here we will use Decision Trees to make this prediction. Note that as the data attributes are continuous numbers yo have to use the attribute and determine a threshold for each node in the tree. As a result you need to solve the information gain for each threshold that is half way between two data points and thus the complexity of the computations increases with the number of data items.

a) Show the construction steps in the construction of a 2 level decision tree using a single step looka- head search and maximum information gain as the construction criterion. You should include the entropy calculations and the construction decisions for each node you include in the 2-level tree.

Since the size of the depth-limited search used in the construction of the tree depends on the training set size, you should limit the data to only the first 3 data items for each gender in the data set your generated for Questions 2 a) and 3a) (the smaller data set for manual work) in Homework 1.

### 2.0.1 Importing the relevant data.

[1]:
```
from data import (
    heights,
    weights,
    age,
    gender,
    samples,
    k_values,
    program_height,
    program_age,
```

```
    program_gender,
    program_weight,
    df,
)
import numpy as np
import math
import scipy as sp
```

[291]:
```
import warnings
#warnings.filter(Setti)
```

[2]:
```
df.shape
```

[2]: (120, 4)

[4]:
```
df.head()
```

[4]:

| | height | weight | age | gender |
|---|---|---|---|---|
| 0 | 1.702639 | 76.802746 | 25 | M |
| 1 | 1.697199 | 77.269872 | 24 | W |
| 2 | 1.834627 | 83.110254 | 23 | M |
| 3 | 1.937070 | 81.158630 | 30 | M |
| 4 | 1.883345 | 79.561306 | 29 | M |

**entropy formula**

$$Entropy = \sum_i -p_i log_2 p_i$$

[207]:
```
def entropy_function(count,n):
    """The above formula."""
    if n==0:n=1
    return -(count*1.0/n)*math.log(count*1.0/n,2)

def entropy_calculate(class1_count,class2_count):
    """Returns entropy of a group of data."""
    if class1_count==0 or class2_count==0:
        return 0
    return␣
 ↪entropy_function(class1_count,class1_count+class2_count)+entropy_function(class2_count,clas

def entropy_of_one_division(division):
    """Returns entropy of a divided group of data which may have multiple␣
 ↪classes."""
    s=0
    n=len(division)
    if n==0:n=1
```

```
    classes=set(division)
    for c in classes: #for each class get its entropy
        n_c=sum(division==c)
        # weighted average
        e=n_c*1.0/n*entropy_calculate(sum(division==c),sum(division!=c))
        s+=e
    return s,n

def get_entropy(y_pred,y_real):
    """Returns the entropy of a split.

    y_pred is the split decision, True/False, and y_real can be multiclass.
    """
    if len(y_pred)!=len(y_real):
        raise ValueError(f"y_pred: {len(y_pred)} and y_real: {len(y_real)}␣
 ↪should be of the same length.")
    n=len(y_real)
    if n==0:n=1
    s_true,n_true=entropy_of_one_division(y_real[y_pred]) # LHS entropy
    s_false,n_false=entropy_of_one_division(y_real[-y_pred]) # RHS entropy
    s=n_true * 1.0/n * s_true + n_false * 1.0/n # Overall entropy
    return s
```

**Decision tree classifier**

[279]:
```
class DecisionTreeClf(object):
    def __init__(self, max_depth=2,␣
 ↪column_names=["height","age","weight","gender_code"]):
        self.depth=0
        self.max_depth=max_depth
        self.column_names=column_names
        self.start_ix=0

    def find_best_split(self,column,target_var):
        """column: col we split on
        target_var.

        """
        min_entropy=8
        n=len(target_var)
        for value in set(self.y):
            y_pred=column<value
            # separate into 2 groups
            y_true=(self.y<value)#.iloc[self.start_ix:self.start_ix+3]
```

```
            y_pred=y_pred#.iloc[self.start_ix:self.start_ix+3] # limit to first␣
↪three data items for each gender.
            self.start_ix+=3
            split_entropy=get_entropy(y_pred,y_true)
            if split_entropy<=min_entropy: # is it the best we have done?
                min_entropy=split_entropy
                cutoff=value
        return min_entropy,cutoff
    def find_best_split_of_all(self, x, y):
        """
        Find the best split from all features
        returns: the column to split on, the cutoff value, and the actual␣
↪entropy
        """
        col = None
        min_entropy = 1
        cutoff = None
        for i, c in enumerate([y]):  # iterating through each feature
            entropy, cur_cutoff = self.find_best_split(c, y)  # find the best␣
↪split of that feature
            if entropy == 0:    # find the first perfect cutoff. Stop Iterating
                return i, cur_cutoff, entropy
            elif entropy <= min_entropy:  # check if it's best so far
                min_entropy = entropy
                col = i
                cutoff = cur_cutoff
        return col, cutoff, min_entropy
    def fit(self, x, y, par_node={}, depth=0):
        """
        x: Feature set
        y: target variable
        par_node: will be the tree generated for this x and y.
        depth: the depth of the current layer
        """
        self.x=x
        self.y=y
        if par_node is None:   # base case 1: tree stops at previous level
            return None
        elif len(y) == 0:   # base case 2: no data in this group
            return None
        elif self.all_same(y):   # base case 3: all y is the same in this group
            return {'val':y.iloc[0]}
        elif depth >= self.max_depth:   # base case 4: max depth reached
            return None
        else:   # Recursively generate trees!
            # find one split given an information gain
            cutoff, entropy = self.find_best_split(y, y)
```

```
            col=3
            y_left = y[ y< cutoff]  # left hand side data
            y_right = y[y>= cutoff ] # right hand side data
            par_node = {'col': self.column_names[col], 'index_col':col,
                        'cutoff':cutoff,
                       'val': np.round(np.mean(y))}  # save the information
            # generate tree for the left hand side data
            par_node['left'] = self.fit(x[ y< cutoff], y_left, {}, depth+1)
            # right hand side trees
            par_node['right'] = self.fit(x[y >= cutoff], y_right, {}, depth+1)
            self.depth += 1   # increase the depth since we call fit once
            self.trees = par_node
            return par_node
    def predict(self, x):
        """Predict."""
        results = np.array([0]*len(x))
        for i, c in enumerate(x):  # for each row in test data
            results[i] = self._get_prediction(c)
        return results

    def all_same(self, items):
        return all(x == items.iloc[0] for x in items)

    def _get_prediction(self,row):
        cur_layer = self.trees  # get the tree we build in training
        while cur_layer.get('cutoff'):
            # if not leaf node
            #print(row)
            if all(self.x[row] < cur_layer['cutoff']):   # get the direction
                cur_layer = cur_layer['left']
            else:
                cur_layer = cur_layer['right']
        else:   # if leaf node, return value
            return cur_layer.get('val')


from pprint import pprint
```

**Training data and preprocessing**

[280]:
```
df['gender_code']=df.gender.apply(lambda g: int(g=="M"))
df.head()
```

[280]:

| | height | weight | age | gender | gender_code |
|---|---|---|---|---|---|
| 0 | 1.702639 | 76.802746 | 25 | M | 1 |

```
1   1.697199   77.269872   24       W              0
2   1.834627   83.110254   23       M              1
3   1.937070   81.158630   30       M              1
4   1.883345   79.561306   29       M              1
```

c) Divide the data set from Question 2c) in Homework 1 (the large training data set) into a training set comprising the first 90 data points and a test set consisting of the last 30 data elements. Use the resulting training set to derive trees of depths 1 - 8 and evaluate the accuracy of the resulting trees for the 90 training samples and for the test set containing the last 30 data items. Compare the classification accuracy on the test set with the one on the training set for each tree depth. For which

[281]:
```
train_df=df.iloc[0:90]
test_df=df.iloc[90:]
```

[282]:
```
perfs={}
for depth in range(1,9):
    clf=DecisionTreeClf(max_depth=depth)
    output=clf.fit(train_df[["height","age","weight"]],train_df.gender_code)

    df.dtypes

    y_h=clf.predict(train_df[["height","age","weight"]])
    y_h
    cols=["height","age","weight","gender","gender_code","prediction"]
    train_df["prediction"]=y_h
    train_df[cols];

    correct=train_df[train_df.gender_code==train_df.prediction]
    print(f"Depth={depth} Correct Predictions on train set:\n{len(correct)}/
 ↪{len(train_df)} {len(correct)/len(train_df)}")
    correct
    pct=100*len(correct)/len(train_df)
    perfs[f"depth_{depth}_train"]=pct

    y_hat=clf.predict(test_df[["height","age","weight"]])
    y_hat

    cols=["height","age","weight","gender","gender_code","prediction"]
    test_df["prediction"]=y_hat
    test_df[cols];

    correct=test_df[test_df.gender_code==test_df.prediction]
    print(f"Depth={depth} Correct Predictions on test set:\n{len(correct)}/
 ↪{len(test_df)} {len(correct)/len(test_df)}")
    pct=100*len(correct)/len(test_df)
    perfs[f"depth_{depth}_test"]=pct
```

```
    correct
perfs
```

[282]:
```
height         float64
weight         float64
age              int64
gender          object
gender_code      int64
dtype: object
```

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  train_df["prediction"]=y_h
```

[282]:

| | height | age | weight | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 0 | 1.702639 | 25 | 76.802746 | M | 1 | 1 |
| 1 | 1.697199 | 24 | 77.269872 | W | 0 | 1 |
| 2 | 1.834627 | 23 | 83.110254 | M | 1 | 1 |
| 3 | 1.937070 | 30 | 81.158630 | M | 1 | 0 |
| 4 | 1.883345 | 29 | 79.561306 | M | 1 | 0 |
| .. | … | … | … | … | … | … |
| 85 | 1.965364 | 31 | 86.262710 | M | 1 | 0 |
| 86 | 1.850676 | 27 | 86.960834 | M | 1 | 0 |
| 87 | 1.966283 | 37 | 85.552579 | M | 1 | 0 |
| 88 | 1.773983 | 29 | 79.642015 | W | 0 | 0 |
| 89 | 1.900062 | 25 | 91.090231 | M | 1 | 0 |

[90 rows x 6 columns]

```
Depth=1 Correct Predictions on train set:
47/90 0.5222222222222223
```

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 0 | 1.702639 | 76.802746 | 25 | M | 1 | 1 |
| 2 | 1.834627 | 83.110254 | 23 | M | 1 | 1 |
| 9 | 1.853558 | 83.137510 | 28 | W | 0 | 0 |
| 11 | 1.829861 | 82.203526 | 26 | W | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 12 | 1.862427 | 86.337328 | 35 | W | 0 | 0 |
| 13 | 1.782723 | 78.196155 | 29 | W | 0 | 0 |
| 14 | 1.709718 | 81.856716 | 30 | W | 0 | 0 |
| 16 | 1.668507 | 77.666072 | 27 | W | 0 | 0 |
| 17 | 1.788375 | 78.061393 | 31 | W | 0 | 0 |
| 18 | 1.577425 | 74.280574 | 24 | W | 0 | 0 |
| 19 | 1.671554 | 84.663993 | 27 | W | 0 | 0 |
| 20 | 1.706190 | 75.046986 | 25 | 'W' | 0 | 0 |
| 22 | 1.769952 | 72.378755 | 27 | W | 0 | 0 |
| 24 | 1.733634 | 84.789650 | 28 | W | 0 | 0 |
| 25 | 1.763756 | 82.519046 | 33 | W | 0 | 0 |
| 26 | 1.961237 | 79.264595 | 31 | W | 0 | 0 |
| 27 | 1.700674 | 75.044275 | 32 | W | 0 | 0 |
| 29 | 1.666823 | 77.220145 | 29 | W | 0 | 0 |
| 32 | 1.803888 | 81.623692 | 31 | W | 0 | 0 |
| 33 | 1.798957 | 83.403147 | 28 | W | 0 | 0 |
| 34 | 1.667418 | 74.992834 | 28 | W | 0 | 0 |
| 37 | 1.849190 | 82.364749 | 26 | W | 0 | 0 |
| 38 | 1.679098 | 82.020502 | 33 | W | 0 | 0 |
| 40 | 1.685154 | 75.471532 | 31 | W | 0 | 0 |
| 42 | 1.806573 | 82.362830 | 22 | W | 0 | 0 |
| 43 | 1.877087 | 84.093265 | 34 | W | 0 | 0 |
| 48 | 1.710512 | 74.244260 | 29 | W | 0 | 0 |
| 49 | 1.874713 | 81.501337 | 31 | W | 0 | 0 |
| 50 | 1.827066 | 76.324794 | 26 | W | 0 | 0 |
| 53 | 1.796951 | 83.750707 | 35 | W | 0 | 0 |
| 56 | 1.826650 | 72.953277 | 29 | W | 0 | 0 |
| 57 | 1.552963 | 80.245162 | 23 | W | 0 | 0 |
| 60 | 1.633936 | 75.404183 | 32 | W | 0 | 0 |
| 61 | 1.727307 | 82.988647 | 26 | W | 0 | 0 |
| 64 | 1.777084 | 80.256590 | 29 | W | 0 | 0 |
| 66 | 1.891844 | 79.767829 | 29 | W | 0 | 0 |
| 69 | 1.699993 | 79.345944 | 33 | W | 0 | 0 |
| 70 | 1.805395 | 75.671820 | 27 | W | 0 | 0 |
| 71 | 1.799262 | 85.555169 | 32 | W | 0 | 0 |
| 72 | 1.842958 | 88.801241 | 27 | W | 0 | 0 |
| 73 | 1.676077 | 80.070363 | 32 | W | 0 | 0 |
| 74 | 1.706260 | 74.122181 | 25 | W | 0 | 0 |
| 75 | 1.824621 | 83.905609 | 29 | W | 0 | 0 |
| 77 | 1.879400 | 83.226992 | 26 | W | 0 | 0 |
| 78 | 1.871326 | 79.413841 | 34 | W | 0 | 0 |
| 80 | 1.771728 | 84.736232 | 37 | W | 0 | 0 |
| 88 | 1.773983 | 79.642015 | 29 | W | 0 | 0 |

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df["prediction"]=y_hat
```

[282]:

| | height | age | weight | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 33 | 90.577993 | M | 1 | 1 |
| 91 | 1.892626 | 23 | 76.458690 | W | 0 | 1 |
| 92 | 1.942391 | 31 | 94.784788 | M | 1 | 1 |
| 93 | 1.682035 | 28 | 72.618518 | W | 0 | 0 |
| 94 | 1.881043 | 38 | 87.473840 | W | 0 | 0 |
| 95 | 1.792505 | 34 | 75.305459 | W | 0 | 0 |
| 96 | 1.713533 | 29 | 73.429952 | W | 0 | 0 |
| 97 | 1.692998 | 30 | 78.575801 | W | 0 | 0 |
| 98 | 1.838507 | 27 | 86.287091 | M | 1 | 0 |
| 99 | 1.954838 | 27 | 85.839734 | M | 1 | 0 |
| 100 | 1.800308 | 28 | 75.019501 | W | 0 | 0 |
| 101 | 1.906319 | 27 | 79.934613 | W | 0 | 0 |
| 102 | 1.891926 | 32 | 88.178802 | W | 0 | 0 |
| 103 | 1.697087 | 29 | 77.453411 | M | 1 | 0 |
| 104 | 1.798921 | 29 | 81.784528 | M | 1 | 0 |
| 105 | 1.749636 | 30 | 81.512257 | M | 1 | 0 |
| 106 | 2.011484 | 27 | 94.448590 | M | 1 | 0 |
| 107 | 1.903283 | 29 | 90.140224 | M | 1 | 0 |
| 108 | 1.875671 | 29 | 90.281078 | M | 1 | 0 |
| 109 | 1.683467 | 34 | 76.949406 | M | 1 | 0 |
| 110 | 1.789365 | 27 | 81.821186 | M | 1 | 0 |
| 111 | 1.760697 | 26 | 87.774844 | M | 1 | 0 |
| 112 | 1.727774 | 38 | 82.169604 | M | 1 | 0 |
| 113 | 1.796586 | 30 | 90.562409 | M | 1 | 0 |
| 114 | 1.933535 | 28 | 91.069324 | M | 1 | 0 |
| 115 | 1.707365 | 20 | 80.025263 | M | 1 | 0 |
| 116 | 1.770973 | 27 | 78.076260 | M | 1 | 0 |
| 117 | 1.889259 | 30 | 84.519229 | M | 1 | 0 |
| 118 | 1.980576 | 25 | 95.756978 | M | 1 | 0 |
| 119 | 1.867820 | 29 | 91.046693 | M | 1 | 0 |

```
Depth=1 Correct Predictions on test set:
10/30 0.3333333333333333
```

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 90.577993 | 33 | M | 1 | 1 |
| 92 | 1.942391 | 94.784788 | 31 | M | 1 | 1 |
| 93 | 1.682035 | 72.618518 | 28 | W | 0 | 0 |
| 94 | 1.881043 | 87.473840 | 38 | W | 0 | 0 |

```
95   1.792505  75.305459   34      W             0           0
96   1.713533  73.429952   29      W             0           0
97   1.692998  78.575801   30      W             0           0
100  1.800308  75.019501   28      W             0           0
101  1.906319  79.934613   27      W             0           0
102  1.891926  88.178802   32      W             0           0
```

[282]:
```
height         float64
weight         float64
age              int64
gender          object
gender_code      int64
dtype: object
```

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  train_df["prediction"]=y_h
```

[282]:
```
      height  age     weight gender  gender_code  prediction
0   1.702639   25  76.802746      M            1           1
1   1.697199   24  77.269872      W            0           1
2   1.834627   23  83.110254      M            1           1
3   1.937070   30  81.158630      M            1           0
4   1.883345   29  79.561306      M            1           0
..       …  …        …    …           …          …
85  1.965364   31  86.262710      M            1           0
86  1.850676   27  86.960834      M            1           0
87  1.966283   37  85.552579      M            1           0
88  1.773983   29  79.642015      W            0           0
89  1.900062   25  91.090231      M            1           0

[90 rows x 6 columns]
```

```
Depth=2 Correct Predictions on train set:
47/90 0.5222222222222223
```

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 0 | 1.702639 | 76.802746 | 25 | M | 1 | 1 |
| 2 | 1.834627 | 83.110254 | 23 | M | 1 | 1 |
| 9 | 1.853558 | 83.137510 | 28 | W | 0 | 0 |
| 11 | 1.829861 | 82.203526 | 26 | W | 0 | 0 |
| 12 | 1.862427 | 86.337328 | 35 | W | 0 | 0 |
| 13 | 1.782723 | 78.196155 | 29 | W | 0 | 0 |
| 14 | 1.709718 | 81.856716 | 30 | W | 0 | 0 |
| 16 | 1.668507 | 77.666072 | 27 | W | 0 | 0 |
| 17 | 1.788375 | 78.061393 | 31 | W | 0 | 0 |
| 18 | 1.577425 | 74.280574 | 24 | W | 0 | 0 |
| 19 | 1.671554 | 84.663993 | 27 | W | 0 | 0 |
| 20 | 1.706190 | 75.046986 | 25 | 'W' | 0 | 0 |
| 22 | 1.769952 | 72.378755 | 27 | W | 0 | 0 |
| 24 | 1.733634 | 84.789650 | 28 | W | 0 | 0 |
| 25 | 1.763756 | 82.519046 | 33 | W | 0 | 0 |
| 26 | 1.961237 | 79.264595 | 31 | W | 0 | 0 |
| 27 | 1.700674 | 75.044275 | 32 | W | 0 | 0 |
| 29 | 1.666823 | 77.220145 | 29 | W | 0 | 0 |
| 32 | 1.803888 | 81.623692 | 31 | W | 0 | 0 |
| 33 | 1.798957 | 83.403147 | 28 | W | 0 | 0 |
| 34 | 1.667418 | 74.992834 | 28 | W | 0 | 0 |
| 37 | 1.849190 | 82.364749 | 26 | W | 0 | 0 |
| 38 | 1.679098 | 82.020502 | 33 | W | 0 | 0 |
| 40 | 1.685154 | 75.471532 | 31 | W | 0 | 0 |
| 42 | 1.806573 | 82.362830 | 22 | W | 0 | 0 |
| 43 | 1.877087 | 84.093265 | 34 | W | 0 | 0 |
| 48 | 1.710512 | 74.244260 | 29 | W | 0 | 0 |
| 49 | 1.874713 | 81.501337 | 31 | W | 0 | 0 |
| 50 | 1.827066 | 76.324794 | 26 | W | 0 | 0 |
| 53 | 1.796951 | 83.750707 | 35 | W | 0 | 0 |
| 56 | 1.826650 | 72.953277 | 29 | W | 0 | 0 |
| 57 | 1.552963 | 80.245162 | 23 | W | 0 | 0 |
| 60 | 1.633936 | 75.404183 | 32 | W | 0 | 0 |
| 61 | 1.727307 | 82.988647 | 26 | W | 0 | 0 |
| 64 | 1.777084 | 80.256590 | 29 | W | 0 | 0 |
| 66 | 1.891844 | 79.767829 | 29 | W | 0 | 0 |
| 69 | 1.699993 | 79.345944 | 33 | W | 0 | 0 |
| 70 | 1.805395 | 75.671820 | 27 | W | 0 | 0 |
| 71 | 1.799262 | 85.555169 | 32 | W | 0 | 0 |
| 72 | 1.842958 | 88.801241 | 27 | W | 0 | 0 |
| 73 | 1.676077 | 80.070363 | 32 | W | 0 | 0 |
| 74 | 1.706260 | 74.122181 | 25 | W | 0 | 0 |
| 75 | 1.824621 | 83.905609 | 29 | W | 0 | 0 |
| 77 | 1.879400 | 83.226992 | 26 | W | 0 | 0 |
| 78 | 1.871326 | 79.413841 | 34 | W | 0 | 0 |
| 80 | 1.771728 | 84.736232 | 37 | W | 0 | 0 |

```
 88  1.773983  79.642015   29      W            0           0
```

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df["prediction"]=y_hat
```

[282]:

| | height | age | weight | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 33 | 90.577993 | M | 1 | 1 |
| 91 | 1.892626 | 23 | 76.458690 | W | 0 | 1 |
| 92 | 1.942391 | 31 | 94.784788 | M | 1 | 1 |
| 93 | 1.682035 | 28 | 72.618518 | W | 0 | 0 |
| 94 | 1.881043 | 38 | 87.473840 | W | 0 | 0 |
| 95 | 1.792505 | 34 | 75.305459 | W | 0 | 0 |
| 96 | 1.713533 | 29 | 73.429952 | W | 0 | 0 |
| 97 | 1.692998 | 30 | 78.575801 | W | 0 | 0 |
| 98 | 1.838507 | 27 | 86.287091 | M | 1 | 0 |
| 99 | 1.954838 | 27 | 85.839734 | M | 1 | 0 |
| 100 | 1.800308 | 28 | 75.019501 | W | 0 | 0 |
| 101 | 1.906319 | 27 | 79.934613 | W | 0 | 0 |
| 102 | 1.891926 | 32 | 88.178802 | W | 0 | 0 |
| 103 | 1.697087 | 29 | 77.453411 | M | 1 | 0 |
| 104 | 1.798921 | 29 | 81.784528 | M | 1 | 0 |
| 105 | 1.749636 | 30 | 81.512257 | M | 1 | 0 |
| 106 | 2.011484 | 27 | 94.448590 | M | 1 | 0 |
| 107 | 1.903283 | 29 | 90.140224 | M | 1 | 0 |
| 108 | 1.875671 | 29 | 90.281078 | M | 1 | 0 |
| 109 | 1.683467 | 34 | 76.949406 | M | 1 | 0 |
| 110 | 1.789365 | 27 | 81.821186 | M | 1 | 0 |
| 111 | 1.760697 | 26 | 87.774844 | M | 1 | 0 |
| 112 | 1.727774 | 38 | 82.169604 | M | 1 | 0 |
| 113 | 1.796586 | 30 | 90.562409 | M | 1 | 0 |
| 114 | 1.933535 | 28 | 91.069324 | M | 1 | 0 |
| 115 | 1.707365 | 20 | 80.025263 | M | 1 | 0 |
| 116 | 1.770973 | 27 | 78.076260 | M | 1 | 0 |
| 117 | 1.889259 | 30 | 84.519229 | M | 1 | 0 |
| 118 | 1.980576 | 25 | 95.756978 | M | 1 | 0 |
| 119 | 1.867820 | 29 | 91.046693 | M | 1 | 0 |

```
Depth=2 Correct Predictions on test set:
10/30 0.3333333333333333
```

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 90.577993 | 33 | M | 1 | 1 |
| 92 | 1.942391 | 94.784788 | 31 | M | 1 | 1 |
| 93 | 1.682035 | 72.618518 | 28 | W | 0 | 0 |
| 94 | 1.881043 | 87.473840 | 38 | W | 0 | 0 |
| 95 | 1.792505 | 75.305459 | 34 | W | 0 | 0 |
| 96 | 1.713533 | 73.429952 | 29 | W | 0 | 0 |
| 97 | 1.692998 | 78.575801 | 30 | W | 0 | 0 |
| 100 | 1.800308 | 75.019501 | 28 | W | 0 | 0 |
| 101 | 1.906319 | 79.934613 | 27 | W | 0 | 0 |
| 102 | 1.891926 | 88.178802 | 32 | W | 0 | 0 |

[282]:
```
height         float64
weight         float64
age              int64
gender          object
gender_code      int64
dtype: object
```

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  train_df["prediction"]=y_h
```

[282]:

| | height | age | weight | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 0 | 1.702639 | 25 | 76.802746 | M | 1 | 1 |
| 1 | 1.697199 | 24 | 77.269872 | W | 0 | 1 |
| 2 | 1.834627 | 23 | 83.110254 | M | 1 | 1 |
| 3 | 1.937070 | 30 | 81.158630 | M | 1 | 0 |
| 4 | 1.883345 | 29 | 79.561306 | M | 1 | 0 |
| .. | … | … | … | … | … | … |
| 85 | 1.965364 | 31 | 86.262710 | M | 1 | 0 |
| 86 | 1.850676 | 27 | 86.960834 | M | 1 | 0 |
| 87 | 1.966283 | 37 | 85.552579 | M | 1 | 0 |
| 88 | 1.773983 | 29 | 79.642015 | W | 0 | 0 |
| 89 | 1.900062 | 25 | 91.090231 | M | 1 | 0 |

[90 rows x 6 columns]

```
Depth=3 Correct Predictions on train set:
47/90 0.5222222222222223
```

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 0 | 1.702639 | 76.802746 | 25 | M | 1 | 1 |
| 2 | 1.834627 | 83.110254 | 23 | M | 1 | 1 |
| 9 | 1.853558 | 83.137510 | 28 | W | 0 | 0 |
| 11 | 1.829861 | 82.203526 | 26 | W | 0 | 0 |
| 12 | 1.862427 | 86.337328 | 35 | W | 0 | 0 |
| 13 | 1.782723 | 78.196155 | 29 | W | 0 | 0 |
| 14 | 1.709718 | 81.856716 | 30 | W | 0 | 0 |
| 16 | 1.668507 | 77.666072 | 27 | W | 0 | 0 |
| 17 | 1.788375 | 78.061393 | 31 | W | 0 | 0 |
| 18 | 1.577425 | 74.280574 | 24 | W | 0 | 0 |
| 19 | 1.671554 | 84.663993 | 27 | W | 0 | 0 |
| 20 | 1.706190 | 75.046986 | 25 | 'W' | 0 | 0 |
| 22 | 1.769952 | 72.378755 | 27 | W | 0 | 0 |
| 24 | 1.733634 | 84.789650 | 28 | W | 0 | 0 |
| 25 | 1.763756 | 82.519046 | 33 | W | 0 | 0 |
| 26 | 1.961237 | 79.264595 | 31 | W | 0 | 0 |
| 27 | 1.700674 | 75.044275 | 32 | W | 0 | 0 |
| 29 | 1.666823 | 77.220145 | 29 | W | 0 | 0 |
| 32 | 1.803888 | 81.623692 | 31 | W | 0 | 0 |
| 33 | 1.798957 | 83.403147 | 28 | W | 0 | 0 |
| 34 | 1.667418 | 74.992834 | 28 | W | 0 | 0 |
| 37 | 1.849190 | 82.364749 | 26 | W | 0 | 0 |
| 38 | 1.679098 | 82.020502 | 33 | W | 0 | 0 |
| 40 | 1.685154 | 75.471532 | 31 | W | 0 | 0 |
| 42 | 1.806573 | 82.362830 | 22 | W | 0 | 0 |
| 43 | 1.877087 | 84.093265 | 34 | W | 0 | 0 |
| 48 | 1.710512 | 74.244260 | 29 | W | 0 | 0 |
| 49 | 1.874713 | 81.501337 | 31 | W | 0 | 0 |
| 50 | 1.827066 | 76.324794 | 26 | W | 0 | 0 |
| 53 | 1.796951 | 83.750707 | 35 | W | 0 | 0 |
| 56 | 1.826650 | 72.953277 | 29 | W | 0 | 0 |
| 57 | 1.552963 | 80.245162 | 23 | W | 0 | 0 |
| 60 | 1.633936 | 75.404183 | 32 | W | 0 | 0 |
| 61 | 1.727307 | 82.988647 | 26 | W | 0 | 0 |
| 64 | 1.777084 | 80.256590 | 29 | W | 0 | 0 |
| 66 | 1.891844 | 79.767829 | 29 | W | 0 | 0 |
| 69 | 1.699993 | 79.345944 | 33 | W | 0 | 0 |
| 70 | 1.805395 | 75.671820 | 27 | W | 0 | 0 |
| 71 | 1.799262 | 85.555169 | 32 | W | 0 | 0 |
| 72 | 1.842958 | 88.801241 | 27 | W | 0 | 0 |
| 73 | 1.676077 | 80.070363 | 32 | W | 0 | 0 |
| 74 | 1.706260 | 74.122181 | 25 | W | 0 | 0 |
| 75 | 1.824621 | 83.905609 | 29 | W | 0 | 0 |

```
77  1.879400  83.226992   26      W            0           0
78  1.871326  79.413841   34      W            0           0
80  1.771728  84.736232   37      W            0           0
88  1.773983  79.642015   29      W            0           0
```

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df["prediction"]=y_hat
```

[282]:

| | height | age | weight | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 33 | 90.577993 | M | 1 | 1 |
| 91 | 1.892626 | 23 | 76.458690 | W | 0 | 1 |
| 92 | 1.942391 | 31 | 94.784788 | M | 1 | 1 |
| 93 | 1.682035 | 28 | 72.618518 | W | 0 | 0 |
| 94 | 1.881043 | 38 | 87.473840 | W | 0 | 0 |
| 95 | 1.792505 | 34 | 75.305459 | W | 0 | 0 |
| 96 | 1.713533 | 29 | 73.429952 | W | 0 | 0 |
| 97 | 1.692998 | 30 | 78.575801 | W | 0 | 0 |
| 98 | 1.838507 | 27 | 86.287091 | M | 1 | 0 |
| 99 | 1.954838 | 27 | 85.839734 | M | 1 | 0 |
| 100 | 1.800308 | 28 | 75.019501 | W | 0 | 0 |
| 101 | 1.906319 | 27 | 79.934613 | W | 0 | 0 |
| 102 | 1.891926 | 32 | 88.178802 | W | 0 | 0 |
| 103 | 1.697087 | 29 | 77.453411 | M | 1 | 0 |
| 104 | 1.798921 | 29 | 81.784528 | M | 1 | 0 |
| 105 | 1.749636 | 30 | 81.512257 | M | 1 | 0 |
| 106 | 2.011484 | 27 | 94.448590 | M | 1 | 0 |
| 107 | 1.903283 | 29 | 90.140224 | M | 1 | 0 |
| 108 | 1.875671 | 29 | 90.281078 | M | 1 | 0 |
| 109 | 1.683467 | 34 | 76.949406 | M | 1 | 0 |
| 110 | 1.789365 | 27 | 81.821186 | M | 1 | 0 |
| 111 | 1.760697 | 26 | 87.774844 | M | 1 | 0 |
| 112 | 1.727774 | 38 | 82.169604 | M | 1 | 0 |
| 113 | 1.796586 | 30 | 90.562409 | M | 1 | 0 |
| 114 | 1.933535 | 28 | 91.069324 | M | 1 | 0 |
| 115 | 1.707365 | 20 | 80.025263 | M | 1 | 0 |
| 116 | 1.770973 | 27 | 78.076260 | M | 1 | 0 |
| 117 | 1.889259 | 30 | 84.519229 | M | 1 | 0 |
| 118 | 1.980576 | 25 | 95.756978 | M | 1 | 0 |
| 119 | 1.867820 | 29 | 91.046693 | M | 1 | 0 |

```
Depth=3 Correct Predictions on test set:
10/30 0.3333333333333333
```

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 90.577993 | 33 | M | 1 | 1 |
| 92 | 1.942391 | 94.784788 | 31 | M | 1 | 1 |
| 93 | 1.682035 | 72.618518 | 28 | W | 0 | 0 |
| 94 | 1.881043 | 87.473840 | 38 | W | 0 | 0 |
| 95 | 1.792505 | 75.305459 | 34 | W | 0 | 0 |
| 96 | 1.713533 | 73.429952 | 29 | W | 0 | 0 |
| 97 | 1.692998 | 78.575801 | 30 | W | 0 | 0 |
| 100 | 1.800308 | 75.019501 | 28 | W | 0 | 0 |
| 101 | 1.906319 | 79.934613 | 27 | W | 0 | 0 |
| 102 | 1.891926 | 88.178802 | 32 | W | 0 | 0 |

[282]:
```
height         float64
weight         float64
age              int64
gender          object
gender_code      int64
dtype: object
```

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  train_df["prediction"]=y_h
```

[282]:

| | height | age | weight | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 0 | 1.702639 | 25 | 76.802746 | M | 1 | 1 |
| 1 | 1.697199 | 24 | 77.269872 | W | 0 | 1 |
| 2 | 1.834627 | 23 | 83.110254 | M | 1 | 1 |
| 3 | 1.937070 | 30 | 81.158630 | M | 1 | 0 |
| 4 | 1.883345 | 29 | 79.561306 | M | 1 | 0 |
| .. | … | … | … | … | … | … |
| 85 | 1.965364 | 31 | 86.262710 | M | 1 | 0 |
| 86 | 1.850676 | 27 | 86.960834 | M | 1 | 0 |
| 87 | 1.966283 | 37 | 85.552579 | M | 1 | 0 |
| 88 | 1.773983 | 29 | 79.642015 | W | 0 | 0 |
| 89 | 1.900062 | 25 | 91.090231 | M | 1 | 0 |

[90 rows x 6 columns]

Depth=4 Correct Predictions on train set:
47/90 0.5222222222222223

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 0 | 1.702639 | 76.802746 | 25 | M | 1 | 1 |
| 2 | 1.834627 | 83.110254 | 23 | M | 1 | 1 |
| 9 | 1.853558 | 83.137510 | 28 | W | 0 | 0 |
| 11 | 1.829861 | 82.203526 | 26 | W | 0 | 0 |
| 12 | 1.862427 | 86.337328 | 35 | W | 0 | 0 |
| 13 | 1.782723 | 78.196155 | 29 | W | 0 | 0 |
| 14 | 1.709718 | 81.856716 | 30 | W | 0 | 0 |
| 16 | 1.668507 | 77.666072 | 27 | W | 0 | 0 |
| 17 | 1.788375 | 78.061393 | 31 | W | 0 | 0 |
| 18 | 1.577425 | 74.280574 | 24 | W | 0 | 0 |
| 19 | 1.671554 | 84.663993 | 27 | W | 0 | 0 |
| 20 | 1.706190 | 75.046986 | 25 | 'W' | 0 | 0 |
| 22 | 1.769952 | 72.378755 | 27 | W | 0 | 0 |
| 24 | 1.733634 | 84.789650 | 28 | W | 0 | 0 |
| 25 | 1.763756 | 82.519046 | 33 | W | 0 | 0 |
| 26 | 1.961237 | 79.264595 | 31 | W | 0 | 0 |
| 27 | 1.700674 | 75.044275 | 32 | W | 0 | 0 |
| 29 | 1.666823 | 77.220145 | 29 | W | 0 | 0 |
| 32 | 1.803888 | 81.623692 | 31 | W | 0 | 0 |
| 33 | 1.798957 | 83.403147 | 28 | W | 0 | 0 |
| 34 | 1.667418 | 74.992834 | 28 | W | 0 | 0 |
| 37 | 1.849190 | 82.364749 | 26 | W | 0 | 0 |
| 38 | 1.679098 | 82.020502 | 33 | W | 0 | 0 |
| 40 | 1.685154 | 75.471532 | 31 | W | 0 | 0 |
| 42 | 1.806573 | 82.362830 | 22 | W | 0 | 0 |
| 43 | 1.877087 | 84.093265 | 34 | W | 0 | 0 |
| 48 | 1.710512 | 74.244260 | 29 | W | 0 | 0 |
| 49 | 1.874713 | 81.501337 | 31 | W | 0 | 0 |
| 50 | 1.827066 | 76.324794 | 26 | W | 0 | 0 |
| 53 | 1.796951 | 83.750707 | 35 | W | 0 | 0 |
| 56 | 1.826650 | 72.953277 | 29 | W | 0 | 0 |
| 57 | 1.552963 | 80.245162 | 23 | W | 0 | 0 |
| 60 | 1.633936 | 75.404183 | 32 | W | 0 | 0 |
| 61 | 1.727307 | 82.988647 | 26 | W | 0 | 0 |
| 64 | 1.777084 | 80.256590 | 29 | W | 0 | 0 |
| 66 | 1.891844 | 79.767829 | 29 | W | 0 | 0 |
| 69 | 1.699993 | 79.345944 | 33 | W | 0 | 0 |
| 70 | 1.805395 | 75.671820 | 27 | W | 0 | 0 |
| 71 | 1.799262 | 85.555169 | 32 | W | 0 | 0 |
| 72 | 1.842958 | 88.801241 | 27 | W | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 73 | 1.676077 | 80.070363 | 32 | W | 0 | 0 |
| 74 | 1.706260 | 74.122181 | 25 | W | 0 | 0 |
| 75 | 1.824621 | 83.905609 | 29 | W | 0 | 0 |
| 77 | 1.879400 | 83.226992 | 26 | W | 0 | 0 |
| 78 | 1.871326 | 79.413841 | 34 | W | 0 | 0 |
| 80 | 1.771728 | 84.736232 | 37 | W | 0 | 0 |
| 88 | 1.773983 | 79.642015 | 29 | W | 0 | 0 |

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df["prediction"]=y_hat
```

[282]:

| | height | age | weight | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 33 | 90.577993 | M | 1 | 1 |
| 91 | 1.892626 | 23 | 76.458690 | W | 0 | 1 |
| 92 | 1.942391 | 31 | 94.784788 | M | 1 | 1 |
| 93 | 1.682035 | 28 | 72.618518 | W | 0 | 0 |
| 94 | 1.881043 | 38 | 87.473840 | W | 0 | 0 |
| 95 | 1.792505 | 34 | 75.305459 | W | 0 | 0 |
| 96 | 1.713533 | 29 | 73.429952 | W | 0 | 0 |
| 97 | 1.692998 | 30 | 78.575801 | W | 0 | 0 |
| 98 | 1.838507 | 27 | 86.287091 | M | 1 | 0 |
| 99 | 1.954838 | 27 | 85.839734 | M | 1 | 0 |
| 100 | 1.800308 | 28 | 75.019501 | W | 0 | 0 |
| 101 | 1.906319 | 27 | 79.934613 | W | 0 | 0 |
| 102 | 1.891926 | 32 | 88.178802 | W | 0 | 0 |
| 103 | 1.697087 | 29 | 77.453411 | M | 1 | 0 |
| 104 | 1.798921 | 29 | 81.784528 | M | 1 | 0 |
| 105 | 1.749636 | 30 | 81.512257 | M | 1 | 0 |
| 106 | 2.011484 | 27 | 94.448590 | M | 1 | 0 |
| 107 | 1.903283 | 29 | 90.140224 | M | 1 | 0 |
| 108 | 1.875671 | 29 | 90.281078 | M | 1 | 0 |
| 109 | 1.683467 | 34 | 76.949406 | M | 1 | 0 |
| 110 | 1.789365 | 27 | 81.821186 | M | 1 | 0 |
| 111 | 1.760697 | 26 | 87.774844 | M | 1 | 0 |
| 112 | 1.727774 | 38 | 82.169604 | M | 1 | 0 |
| 113 | 1.796586 | 30 | 90.562409 | M | 1 | 0 |
| 114 | 1.933535 | 28 | 91.069324 | M | 1 | 0 |
| 115 | 1.707365 | 20 | 80.025263 | M | 1 | 0 |
| 116 | 1.770973 | 27 | 78.076260 | M | 1 | 0 |

```
     117  1.889259   30  84.519229      M             1           0
     118  1.980576   25  95.756978      M             1           0
     119  1.867820   29  91.046693      M             1           0

     Depth=4 Correct Predictions on test set:
     10/30 0.3333333333333333
```

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 90.577993 | 33 | M | 1 | 1 |
| 92 | 1.942391 | 94.784788 | 31 | M | 1 | 1 |
| 93 | 1.682035 | 72.618518 | 28 | W | 0 | 0 |
| 94 | 1.881043 | 87.473840 | 38 | W | 0 | 0 |
| 95 | 1.792505 | 75.305459 | 34 | W | 0 | 0 |
| 96 | 1.713533 | 73.429952 | 29 | W | 0 | 0 |
| 97 | 1.692998 | 78.575801 | 30 | W | 0 | 0 |
| 100 | 1.800308 | 75.019501 | 28 | W | 0 | 0 |
| 101 | 1.906319 | 79.934613 | 27 | W | 0 | 0 |
| 102 | 1.891926 | 88.178802 | 32 | W | 0 | 0 |

[282]:
```
height         float64
weight         float64
age              int64
gender          object
gender_code      int64
dtype: object
```

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  train_df["prediction"]=y_h
```

[282]:

| | height | age | weight | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 0 | 1.702639 | 25 | 76.802746 | M | 1 | 1 |
| 1 | 1.697199 | 24 | 77.269872 | W | 0 | 1 |
| 2 | 1.834627 | 23 | 83.110254 | M | 1 | 1 |
| 3 | 1.937070 | 30 | 81.158630 | M | 1 | 0 |
| 4 | 1.883345 | 29 | 79.561306 | M | 1 | 0 |
| .. | … | … | … | … | … | … |
| 85 | 1.965364 | 31 | 86.262710 | M | 1 | 0 |

```
 86  1.850676   27  86.960834      M            1            0
 87  1.966283   37  85.552579      M            1            0
 88  1.773983   29  79.642015      W            0            0
 89  1.900062   25  91.090231      M            1            0

 [90 rows x 6 columns]

Depth=5 Correct Predictions on train set:
47/90 0.5222222222222223
```

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 0 | 1.702639 | 76.802746 | 25 | M | 1 | 1 |
| 2 | 1.834627 | 83.110254 | 23 | M | 1 | 1 |
| 9 | 1.853558 | 83.137510 | 28 | W | 0 | 0 |
| 11 | 1.829861 | 82.203526 | 26 | W | 0 | 0 |
| 12 | 1.862427 | 86.337328 | 35 | W | 0 | 0 |
| 13 | 1.782723 | 78.196155 | 29 | W | 0 | 0 |
| 14 | 1.709718 | 81.856716 | 30 | W | 0 | 0 |
| 16 | 1.668507 | 77.666072 | 27 | W | 0 | 0 |
| 17 | 1.788375 | 78.061393 | 31 | W | 0 | 0 |
| 18 | 1.577425 | 74.280574 | 24 | W | 0 | 0 |
| 19 | 1.671554 | 84.663993 | 27 | W | 0 | 0 |
| 20 | 1.706190 | 75.046986 | 25 | 'W' | 0 | 0 |
| 22 | 1.769952 | 72.378755 | 27 | W | 0 | 0 |
| 24 | 1.733634 | 84.789650 | 28 | W | 0 | 0 |
| 25 | 1.763756 | 82.519046 | 33 | W | 0 | 0 |
| 26 | 1.961237 | 79.264595 | 31 | W | 0 | 0 |
| 27 | 1.700674 | 75.044275 | 32 | W | 0 | 0 |
| 29 | 1.666823 | 77.220145 | 29 | W | 0 | 0 |
| 32 | 1.803888 | 81.623692 | 31 | W | 0 | 0 |
| 33 | 1.798957 | 83.403147 | 28 | W | 0 | 0 |
| 34 | 1.667418 | 74.992834 | 28 | W | 0 | 0 |
| 37 | 1.849190 | 82.364749 | 26 | W | 0 | 0 |
| 38 | 1.679098 | 82.020502 | 33 | W | 0 | 0 |
| 40 | 1.685154 | 75.471532 | 31 | W | 0 | 0 |
| 42 | 1.806573 | 82.362830 | 22 | W | 0 | 0 |
| 43 | 1.877087 | 84.093265 | 34 | W | 0 | 0 |
| 48 | 1.710512 | 74.244260 | 29 | W | 0 | 0 |
| 49 | 1.874713 | 81.501337 | 31 | W | 0 | 0 |
| 50 | 1.827066 | 76.324794 | 26 | W | 0 | 0 |
| 53 | 1.796951 | 83.750707 | 35 | W | 0 | 0 |
| 56 | 1.826650 | 72.953277 | 29 | W | 0 | 0 |
| 57 | 1.552963 | 80.245162 | 23 | W | 0 | 0 |
| 60 | 1.633936 | 75.404183 | 32 | W | 0 | 0 |
| 61 | 1.727307 | 82.988647 | 26 | W | 0 | 0 |
| 64 | 1.777084 | 80.256590 | 29 | W | 0 | 0 |
| 66 | 1.891844 | 79.767829 | 29 | W | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 69 | 1.699993 | 79.345944 | 33 | W | 0 | 0 |
| 70 | 1.805395 | 75.671820 | 27 | W | 0 | 0 |
| 71 | 1.799262 | 85.555169 | 32 | W | 0 | 0 |
| 72 | 1.842958 | 88.801241 | 27 | W | 0 | 0 |
| 73 | 1.676077 | 80.070363 | 32 | W | 0 | 0 |
| 74 | 1.706260 | 74.122181 | 25 | W | 0 | 0 |
| 75 | 1.824621 | 83.905609 | 29 | W | 0 | 0 |
| 77 | 1.879400 | 83.226992 | 26 | W | 0 | 0 |
| 78 | 1.871326 | 79.413841 | 34 | W | 0 | 0 |
| 80 | 1.771728 | 84.736232 | 37 | W | 0 | 0 |
| 88 | 1.773983 | 79.642015 | 29 | W | 0 | 0 |

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df["prediction"]=y_hat
```

[282]:

| | height | age | weight | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 33 | 90.577993 | M | 1 | 1 |
| 91 | 1.892626 | 23 | 76.458690 | W | 0 | 1 |
| 92 | 1.942391 | 31 | 94.784788 | M | 1 | 1 |
| 93 | 1.682035 | 28 | 72.618518 | W | 0 | 0 |
| 94 | 1.881043 | 38 | 87.473840 | W | 0 | 0 |
| 95 | 1.792505 | 34 | 75.305459 | W | 0 | 0 |
| 96 | 1.713533 | 29 | 73.429952 | W | 0 | 0 |
| 97 | 1.692998 | 30 | 78.575801 | W | 0 | 0 |
| 98 | 1.838507 | 27 | 86.287091 | M | 1 | 0 |
| 99 | 1.954838 | 27 | 85.839734 | M | 1 | 0 |
| 100 | 1.800308 | 28 | 75.019501 | W | 0 | 0 |
| 101 | 1.906319 | 27 | 79.934613 | W | 0 | 0 |
| 102 | 1.891926 | 32 | 88.178802 | W | 0 | 0 |
| 103 | 1.697087 | 29 | 77.453411 | M | 1 | 0 |
| 104 | 1.798921 | 29 | 81.784528 | M | 1 | 0 |
| 105 | 1.749636 | 30 | 81.512257 | M | 1 | 0 |
| 106 | 2.011484 | 27 | 94.448590 | M | 1 | 0 |
| 107 | 1.903283 | 29 | 90.140224 | M | 1 | 0 |
| 108 | 1.875671 | 29 | 90.281078 | M | 1 | 0 |
| 109 | 1.683467 | 34 | 76.949406 | M | 1 | 0 |
| 110 | 1.789365 | 27 | 81.821186 | M | 1 | 0 |
| 111 | 1.760697 | 26 | 87.774844 | M | 1 | 0 |
| 112 | 1.727774 | 38 | 82.169604 | M | 1 | 0 |

```
113  1.796586   30  90.562409      M            1           0
114  1.933535   28  91.069324      M            1           0
115  1.707365   20  80.025263      M            1           0
116  1.770973   27  78.076260      M            1           0
117  1.889259   30  84.519229      M            1           0
118  1.980576   25  95.756978      M            1           0
119  1.867820   29  91.046693      M            1           0
```

```
Depth=5 Correct Predictions on test set:
10/30 0.3333333333333333
```

[282]:
```
       height     weight  age gender  gender_code  prediction
90   1.808332  90.577993   33      M            1           1
92   1.942391  94.784788   31      M            1           1
93   1.682035  72.618518   28      W            0           0
94   1.881043  87.473840   38      W            0           0
95   1.792505  75.305459   34      W            0           0
96   1.713533  73.429952   29      W            0           0
97   1.692998  78.575801   30      W            0           0
100  1.800308  75.019501   28      W            0           0
101  1.906319  79.934613   27      W            0           0
102  1.891926  88.178802   32      W            0           0
```

[282]:
```
height         float64
weight         float64
age              int64
gender          object
gender_code      int64
dtype: object
```

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  train_df["prediction"]=y_h
```

[282]:
```
      height  age     weight gender  gender_code  prediction
0   1.702639   25  76.802746      M            1           1
1   1.697199   24  77.269872      W            0           1
2   1.834627   23  83.110254      M            1           1
```

```
3   1.937070   30  81.158630      M            1            0
4   1.883345   29  79.561306      M            1            0
..       …  …        …    …          …          …
85  1.965364   31  86.262710      M            1            0
86  1.850676   27  86.960834      M            1            0
87  1.966283   37  85.552579      M            1            0
88  1.773983   29  79.642015      W            0            0
89  1.900062   25  91.090231      M            1            0

[90 rows x 6 columns]

Depth=6 Correct Predictions on train set:
47/90 0.5222222222222223
```

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 0 | 1.702639 | 76.802746 | 25 | M | 1 | 1 |
| 2 | 1.834627 | 83.110254 | 23 | M | 1 | 1 |
| 9 | 1.853558 | 83.137510 | 28 | W | 0 | 0 |
| 11 | 1.829861 | 82.203526 | 26 | W | 0 | 0 |
| 12 | 1.862427 | 86.337328 | 35 | W | 0 | 0 |
| 13 | 1.782723 | 78.196155 | 29 | W | 0 | 0 |
| 14 | 1.709718 | 81.856716 | 30 | W | 0 | 0 |
| 16 | 1.668507 | 77.666072 | 27 | W | 0 | 0 |
| 17 | 1.788375 | 78.061393 | 31 | W | 0 | 0 |
| 18 | 1.577425 | 74.280574 | 24 | W | 0 | 0 |
| 19 | 1.671554 | 84.663993 | 27 | W | 0 | 0 |
| 20 | 1.706190 | 75.046986 | 25 | 'W' | 0 | 0 |
| 22 | 1.769952 | 72.378755 | 27 | W | 0 | 0 |
| 24 | 1.733634 | 84.789650 | 28 | W | 0 | 0 |
| 25 | 1.763756 | 82.519046 | 33 | W | 0 | 0 |
| 26 | 1.961237 | 79.264595 | 31 | W | 0 | 0 |
| 27 | 1.700674 | 75.044275 | 32 | W | 0 | 0 |
| 29 | 1.666823 | 77.220145 | 29 | W | 0 | 0 |
| 32 | 1.803888 | 81.623692 | 31 | W | 0 | 0 |
| 33 | 1.798957 | 83.403147 | 28 | W | 0 | 0 |
| 34 | 1.667418 | 74.992834 | 28 | W | 0 | 0 |
| 37 | 1.849190 | 82.364749 | 26 | W | 0 | 0 |
| 38 | 1.679098 | 82.020502 | 33 | W | 0 | 0 |
| 40 | 1.685154 | 75.471532 | 31 | W | 0 | 0 |
| 42 | 1.806573 | 82.362830 | 22 | W | 0 | 0 |
| 43 | 1.877087 | 84.093265 | 34 | W | 0 | 0 |
| 48 | 1.710512 | 74.244260 | 29 | W | 0 | 0 |
| 49 | 1.874713 | 81.501337 | 31 | W | 0 | 0 |
| 50 | 1.827066 | 76.324794 | 26 | W | 0 | 0 |
| 53 | 1.796951 | 83.750707 | 35 | W | 0 | 0 |
| 56 | 1.826650 | 72.953277 | 29 | W | 0 | 0 |
| 57 | 1.552963 | 80.245162 | 23 | W | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 60 | 1.633936 | 75.404183 | 32 | W | 0 | 0 |
| 61 | 1.727307 | 82.988647 | 26 | W | 0 | 0 |
| 64 | 1.777084 | 80.256590 | 29 | W | 0 | 0 |
| 66 | 1.891844 | 79.767829 | 29 | W | 0 | 0 |
| 69 | 1.699993 | 79.345944 | 33 | W | 0 | 0 |
| 70 | 1.805395 | 75.671820 | 27 | W | 0 | 0 |
| 71 | 1.799262 | 85.555169 | 32 | W | 0 | 0 |
| 72 | 1.842958 | 88.801241 | 27 | W | 0 | 0 |
| 73 | 1.676077 | 80.070363 | 32 | W | 0 | 0 |
| 74 | 1.706260 | 74.122181 | 25 | W | 0 | 0 |
| 75 | 1.824621 | 83.905609 | 29 | W | 0 | 0 |
| 77 | 1.879400 | 83.226992 | 26 | W | 0 | 0 |
| 78 | 1.871326 | 79.413841 | 34 | W | 0 | 0 |
| 80 | 1.771728 | 84.736232 | 37 | W | 0 | 0 |
| 88 | 1.773983 | 79.642015 | 29 | W | 0 | 0 |

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df["prediction"]=y_hat
```

[282]:

| | height | age | weight | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 33 | 90.577993 | M | 1 | 1 |
| 91 | 1.892626 | 23 | 76.458690 | W | 0 | 1 |
| 92 | 1.942391 | 31 | 94.784788 | M | 1 | 1 |
| 93 | 1.682035 | 28 | 72.618518 | W | 0 | 0 |
| 94 | 1.881043 | 38 | 87.473840 | W | 0 | 0 |
| 95 | 1.792505 | 34 | 75.305459 | W | 0 | 0 |
| 96 | 1.713533 | 29 | 73.429952 | W | 0 | 0 |
| 97 | 1.692998 | 30 | 78.575801 | W | 0 | 0 |
| 98 | 1.838507 | 27 | 86.287091 | M | 1 | 0 |
| 99 | 1.954838 | 27 | 85.839734 | M | 1 | 0 |
| 100 | 1.800308 | 28 | 75.019501 | W | 0 | 0 |
| 101 | 1.906319 | 27 | 79.934613 | W | 0 | 0 |
| 102 | 1.891926 | 32 | 88.178802 | W | 0 | 0 |
| 103 | 1.697087 | 29 | 77.453411 | M | 1 | 0 |
| 104 | 1.798921 | 29 | 81.784528 | M | 1 | 0 |
| 105 | 1.749636 | 30 | 81.512257 | M | 1 | 0 |
| 106 | 2.011484 | 27 | 94.448590 | M | 1 | 0 |
| 107 | 1.903283 | 29 | 90.140224 | M | 1 | 0 |
| 108 | 1.875671 | 29 | 90.281078 | M | 1 | 0 |

```
109  1.683467   34  76.949406      M             1           0
110  1.789365   27  81.821186      M             1           0
111  1.760697   26  87.774844      M             1           0
112  1.727774   38  82.169604      M             1           0
113  1.796586   30  90.562409      M             1           0
114  1.933535   28  91.069324      M             1           0
115  1.707365   20  80.025263      M             1           0
116  1.770973   27  78.076260      M             1           0
117  1.889259   30  84.519229      M             1           0
118  1.980576   25  95.756978      M             1           0
119  1.867820   29  91.046693      M             1           0

Depth=6 Correct Predictions on test set:
10/30 0.3333333333333333
```

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 90.577993 | 33 | M | 1 | 1 |
| 92 | 1.942391 | 94.784788 | 31 | M | 1 | 1 |
| 93 | 1.682035 | 72.618518 | 28 | W | 0 | 0 |
| 94 | 1.881043 | 87.473840 | 38 | W | 0 | 0 |
| 95 | 1.792505 | 75.305459 | 34 | W | 0 | 0 |
| 96 | 1.713533 | 73.429952 | 29 | W | 0 | 0 |
| 97 | 1.692998 | 78.575801 | 30 | W | 0 | 0 |
| 100 | 1.800308 | 75.019501 | 28 | W | 0 | 0 |
| 101 | 1.906319 | 79.934613 | 27 | W | 0 | 0 |
| 102 | 1.891926 | 88.178802 | 32 | W | 0 | 0 |

[282]:
```
height         float64
weight         float64
age              int64
gender          object
gender_code      int64
dtype: object
```

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0])

/tmp/ipykernel_4105445/3811237255.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  train_df["prediction"]=y_h
```

[282]:

| | height | age | weight | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 0 | 1.702639 | 25 | 76.802746 | M | 1 | 1 |
| 1 | 1.697199 | 24 | 77.269872 | W | 0 | 1 |
| 2 | 1.834627 | 23 | 83.110254 | M | 1 | 1 |
| 3 | 1.937070 | 30 | 81.158630 | M | 1 | 0 |
| 4 | 1.883345 | 29 | 79.561306 | M | 1 | 0 |
| .. | … | … | … | … | … | … |
| 85 | 1.965364 | 31 | 86.262710 | M | 1 | 0 |
| 86 | 1.850676 | 27 | 86.960834 | M | 1 | 0 |
| 87 | 1.966283 | 37 | 85.552579 | M | 1 | 0 |
| 88 | 1.773983 | 29 | 79.642015 | W | 0 | 0 |
| 89 | 1.900062 | 25 | 91.090231 | M | 1 | 0 |

[90 rows x 6 columns]

Depth=7 Correct Predictions on train set:
47/90 0.5222222222222223

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 0 | 1.702639 | 76.802746 | 25 | M | 1 | 1 |
| 2 | 1.834627 | 83.110254 | 23 | M | 1 | 1 |
| 9 | 1.853558 | 83.137510 | 28 | W | 0 | 0 |
| 11 | 1.829861 | 82.203526 | 26 | W | 0 | 0 |
| 12 | 1.862427 | 86.337328 | 35 | W | 0 | 0 |
| 13 | 1.782723 | 78.196155 | 29 | W | 0 | 0 |
| 14 | 1.709718 | 81.856716 | 30 | W | 0 | 0 |
| 16 | 1.668507 | 77.666072 | 27 | W | 0 | 0 |
| 17 | 1.788375 | 78.061393 | 31 | W | 0 | 0 |
| 18 | 1.577425 | 74.280574 | 24 | W | 0 | 0 |
| 19 | 1.671554 | 84.663993 | 27 | W | 0 | 0 |
| 20 | 1.706190 | 75.046986 | 25 | 'W' | 0 | 0 |
| 22 | 1.769952 | 72.378755 | 27 | W | 0 | 0 |
| 24 | 1.733634 | 84.789650 | 28 | W | 0 | 0 |
| 25 | 1.763756 | 82.519046 | 33 | W | 0 | 0 |
| 26 | 1.961237 | 79.264595 | 31 | W | 0 | 0 |
| 27 | 1.700674 | 75.044275 | 32 | W | 0 | 0 |
| 29 | 1.666823 | 77.220145 | 29 | W | 0 | 0 |
| 32 | 1.803888 | 81.623692 | 31 | W | 0 | 0 |
| 33 | 1.798957 | 83.403147 | 28 | W | 0 | 0 |
| 34 | 1.667418 | 74.992834 | 28 | W | 0 | 0 |
| 37 | 1.849190 | 82.364749 | 26 | W | 0 | 0 |
| 38 | 1.679098 | 82.020502 | 33 | W | 0 | 0 |
| 40 | 1.685154 | 75.471532 | 31 | W | 0 | 0 |
| 42 | 1.806573 | 82.362830 | 22 | W | 0 | 0 |
| 43 | 1.877087 | 84.093265 | 34 | W | 0 | 0 |
| 48 | 1.710512 | 74.244260 | 29 | W | 0 | 0 |
| 49 | 1.874713 | 81.501337 | 31 | W | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 50 | 1.827066 | 76.324794 | 26 | W | 0 | 0 |
| 53 | 1.796951 | 83.750707 | 35 | W | 0 | 0 |
| 56 | 1.826650 | 72.953277 | 29 | W | 0 | 0 |
| 57 | 1.552963 | 80.245162 | 23 | W | 0 | 0 |
| 60 | 1.633936 | 75.404183 | 32 | W | 0 | 0 |
| 61 | 1.727307 | 82.988647 | 26 | W | 0 | 0 |
| 64 | 1.777084 | 80.256590 | 29 | W | 0 | 0 |
| 66 | 1.891844 | 79.767829 | 29 | W | 0 | 0 |
| 69 | 1.699993 | 79.345944 | 33 | W | 0 | 0 |
| 70 | 1.805395 | 75.671820 | 27 | W | 0 | 0 |
| 71 | 1.799262 | 85.555169 | 32 | W | 0 | 0 |
| 72 | 1.842958 | 88.801241 | 27 | W | 0 | 0 |
| 73 | 1.676077 | 80.070363 | 32 | W | 0 | 0 |
| 74 | 1.706260 | 74.122181 | 25 | W | 0 | 0 |
| 75 | 1.824621 | 83.905609 | 29 | W | 0 | 0 |
| 77 | 1.879400 | 83.226992 | 26 | W | 0 | 0 |
| 78 | 1.871326 | 79.413841 | 34 | W | 0 | 0 |
| 80 | 1.771728 | 84.736232 | 37 | W | 0 | 0 |
| 88 | 1.773983 | 79.642015 | 29 | W | 0 | 0 |

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df["prediction"]=y_hat
```

[282]:

| | height | age | weight | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 33 | 90.577993 | M | 1 | 1 |
| 91 | 1.892626 | 23 | 76.458690 | W | 0 | 1 |
| 92 | 1.942391 | 31 | 94.784788 | M | 1 | 1 |
| 93 | 1.682035 | 28 | 72.618518 | W | 0 | 0 |
| 94 | 1.881043 | 38 | 87.473840 | W | 0 | 0 |
| 95 | 1.792505 | 34 | 75.305459 | W | 0 | 0 |
| 96 | 1.713533 | 29 | 73.429952 | W | 0 | 0 |
| 97 | 1.692998 | 30 | 78.575801 | W | 0 | 0 |
| 98 | 1.838507 | 27 | 86.287091 | M | 1 | 0 |
| 99 | 1.954838 | 27 | 85.839734 | M | 1 | 0 |
| 100 | 1.800308 | 28 | 75.019501 | W | 0 | 0 |
| 101 | 1.906319 | 27 | 79.934613 | W | 0 | 0 |
| 102 | 1.891926 | 32 | 88.178802 | W | 0 | 0 |
| 103 | 1.697087 | 29 | 77.453411 | M | 1 | 0 |
| 104 | 1.798921 | 29 | 81.784528 | M | 1 | 0 |

```
105  1.749636   30  81.512257      M            1           0
106  2.011484   27  94.448590      M            1           0
107  1.903283   29  90.140224      M            1           0
108  1.875671   29  90.281078      M            1           0
109  1.683467   34  76.949406      M            1           0
110  1.789365   27  81.821186      M            1           0
111  1.760697   26  87.774844      M            1           0
112  1.727774   38  82.169604      M            1           0
113  1.796586   30  90.562409      M            1           0
114  1.933535   28  91.069324      M            1           0
115  1.707365   20  80.025263      M            1           0
116  1.770973   27  78.076260      M            1           0
117  1.889259   30  84.519229      M            1           0
118  1.980576   25  95.756978      M            1           0
119  1.867820   29  91.046693      M            1           0
```

```
Depth=7 Correct Predictions on test set:
10/30 0.3333333333333333
```

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 90.577993 | 33 | M | 1 | 1 |
| 92 | 1.942391 | 94.784788 | 31 | M | 1 | 1 |
| 93 | 1.682035 | 72.618518 | 28 | W | 0 | 0 |
| 94 | 1.881043 | 87.473840 | 38 | W | 0 | 0 |
| 95 | 1.792505 | 75.305459 | 34 | W | 0 | 0 |
| 96 | 1.713533 | 73.429952 | 29 | W | 0 | 0 |
| 97 | 1.692998 | 78.575801 | 30 | W | 0 | 0 |
| 100 | 1.800308 | 75.019501 | 28 | W | 0 | 0 |
| 101 | 1.906319 | 79.934613 | 27 | W | 0 | 0 |
| 102 | 1.891926 | 88.178802 | 32 | W | 0 | 0 |

[282]:
```
height         float64
weight         float64
age              int64
gender          object
gender_code      int64
dtype: object
```

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  train_df["prediction"]=y_h
```

[282]:

| | height | age | weight | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 0 | 1.702639 | 25 | 76.802746 | M | 1 | 1 |
| 1 | 1.697199 | 24 | 77.269872 | W | 0 | 1 |
| 2 | 1.834627 | 23 | 83.110254 | M | 1 | 1 |
| 3 | 1.937070 | 30 | 81.158630 | M | 1 | 0 |
| 4 | 1.883345 | 29 | 79.561306 | M | 1 | 0 |
| .. | … | … | … | … | … | … |
| 85 | 1.965364 | 31 | 86.262710 | M | 1 | 0 |
| 86 | 1.850676 | 27 | 86.960834 | M | 1 | 0 |
| 87 | 1.966283 | 37 | 85.552579 | M | 1 | 0 |
| 88 | 1.773983 | 29 | 79.642015 | W | 0 | 0 |
| 89 | 1.900062 | 25 | 91.090231 | M | 1 | 0 |

[90 rows x 6 columns]

```
Depth=8 Correct Predictions on train set:
47/90 0.5222222222222223
```

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 0 | 1.702639 | 76.802746 | 25 | M | 1 | 1 |
| 2 | 1.834627 | 83.110254 | 23 | M | 1 | 1 |
| 9 | 1.853558 | 83.137510 | 28 | W | 0 | 0 |
| 11 | 1.829861 | 82.203526 | 26 | W | 0 | 0 |
| 12 | 1.862427 | 86.337328 | 35 | W | 0 | 0 |
| 13 | 1.782723 | 78.196155 | 29 | W | 0 | 0 |
| 14 | 1.709718 | 81.856716 | 30 | W | 0 | 0 |
| 16 | 1.668507 | 77.666072 | 27 | W | 0 | 0 |
| 17 | 1.788375 | 78.061393 | 31 | W | 0 | 0 |
| 18 | 1.577425 | 74.280574 | 24 | W | 0 | 0 |
| 19 | 1.671554 | 84.663993 | 27 | W | 0 | 0 |
| 20 | 1.706190 | 75.046986 | 25 | 'W' | 0 | 0 |
| 22 | 1.769952 | 72.378755 | 27 | W | 0 | 0 |
| 24 | 1.733634 | 84.789650 | 28 | W | 0 | 0 |
| 25 | 1.763756 | 82.519046 | 33 | W | 0 | 0 |
| 26 | 1.961237 | 79.264595 | 31 | W | 0 | 0 |
| 27 | 1.700674 | 75.044275 | 32 | W | 0 | 0 |
| 29 | 1.666823 | 77.220145 | 29 | W | 0 | 0 |
| 32 | 1.803888 | 81.623692 | 31 | W | 0 | 0 |
| 33 | 1.798957 | 83.403147 | 28 | W | 0 | 0 |
| 34 | 1.667418 | 74.992834 | 28 | W | 0 | 0 |
| 37 | 1.849190 | 82.364749 | 26 | W | 0 | 0 |
| 38 | 1.679098 | 82.020502 | 33 | W | 0 | 0 |
| 40 | 1.685154 | 75.471532 | 31 | W | 0 | 0 |
| 42 | 1.806573 | 82.362830 | 22 | W | 0 | 0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 43 | 1.877087 | 84.093265 | 34 | W | 0 | 0 |
| 48 | 1.710512 | 74.244260 | 29 | W | 0 | 0 |
| 49 | 1.874713 | 81.501337 | 31 | W | 0 | 0 |
| 50 | 1.827066 | 76.324794 | 26 | W | 0 | 0 |
| 53 | 1.796951 | 83.750707 | 35 | W | 0 | 0 |
| 56 | 1.826650 | 72.953277 | 29 | W | 0 | 0 |
| 57 | 1.552963 | 80.245162 | 23 | W | 0 | 0 |
| 60 | 1.633936 | 75.404183 | 32 | W | 0 | 0 |
| 61 | 1.727307 | 82.988647 | 26 | W | 0 | 0 |
| 64 | 1.777084 | 80.256590 | 29 | W | 0 | 0 |
| 66 | 1.891844 | 79.767829 | 29 | W | 0 | 0 |
| 69 | 1.699993 | 79.345944 | 33 | W | 0 | 0 |
| 70 | 1.805395 | 75.671820 | 27 | W | 0 | 0 |
| 71 | 1.799262 | 85.555169 | 32 | W | 0 | 0 |
| 72 | 1.842958 | 88.801241 | 27 | W | 0 | 0 |
| 73 | 1.676077 | 80.070363 | 32 | W | 0 | 0 |
| 74 | 1.706260 | 74.122181 | 25 | W | 0 | 0 |
| 75 | 1.824621 | 83.905609 | 29 | W | 0 | 0 |
| 77 | 1.879400 | 83.226992 | 26 | W | 0 | 0 |
| 78 | 1.871326 | 79.413841 | 34 | W | 0 | 0 |
| 80 | 1.771728 | 84.736232 | 37 | W | 0 | 0 |
| 88 | 1.773983 | 79.642015 | 29 | W | 0 | 0 |

[282]:
```
array([1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0])
```

```
/tmp/ipykernel_4105445/3811237255.py:24: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df["prediction"]=y_hat
```

[282]:

| | height | age | weight | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 33 | 90.577993 | M | 1 | 1 |
| 91 | 1.892626 | 23 | 76.458690 | W | 0 | 1 |
| 92 | 1.942391 | 31 | 94.784788 | M | 1 | 1 |
| 93 | 1.682035 | 28 | 72.618518 | W | 0 | 0 |
| 94 | 1.881043 | 38 | 87.473840 | W | 0 | 0 |
| 95 | 1.792505 | 34 | 75.305459 | W | 0 | 0 |
| 96 | 1.713533 | 29 | 73.429952 | W | 0 | 0 |
| 97 | 1.692998 | 30 | 78.575801 | W | 0 | 0 |
| 98 | 1.838507 | 27 | 86.287091 | M | 1 | 0 |
| 99 | 1.954838 | 27 | 85.839734 | M | 1 | 0 |
| 100 | 1.800308 | 28 | 75.019501 | W | 0 | 0 |
| 101 | 1.906319 | 27 | 79.934613 | W | 0 | 0 |

```
102  1.891926   32  88.178802      W             0           0
103  1.697087   29  77.453411      M             1           0
104  1.798921   29  81.784528      M             1           0
105  1.749636   30  81.512257      M             1           0
106  2.011484   27  94.448590      M             1           0
107  1.903283   29  90.140224      M             1           0
108  1.875671   29  90.281078      M             1           0
109  1.683467   34  76.949406      M             1           0
110  1.789365   27  81.821186      M             1           0
111  1.760697   26  87.774844      M             1           0
112  1.727774   38  82.169604      M             1           0
113  1.796586   30  90.562409      M             1           0
114  1.933535   28  91.069324      M             1           0
115  1.707365   20  80.025263      M             1           0
116  1.770973   27  78.076260      M             1           0
117  1.889259   30  84.519229      M             1           0
118  1.980576   25  95.756978      M             1           0
119  1.867820   29  91.046693      M             1           0

Depth=8 Correct Predictions on test set:
10/30 0.3333333333333333
```

[282]:

| | height | weight | age | gender | gender_code | prediction |
|---|---|---|---|---|---|---|
| 90 | 1.808332 | 90.577993 | 33 | M | 1 | 1 |
| 92 | 1.942391 | 94.784788 | 31 | M | 1 | 1 |
| 93 | 1.682035 | 72.618518 | 28 | W | 0 | 0 |
| 94 | 1.881043 | 87.473840 | 38 | W | 0 | 0 |
| 95 | 1.792505 | 75.305459 | 34 | W | 0 | 0 |
| 96 | 1.713533 | 73.429952 | 29 | W | 0 | 0 |
| 97 | 1.692998 | 78.575801 | 30 | W | 0 | 0 |
| 100 | 1.800308 | 75.019501 | 28 | W | 0 | 0 |
| 101 | 1.906319 | 79.934613 | 27 | W | 0 | 0 |
| 102 | 1.891926 | 88.178802 | 32 | W | 0 | 0 |

[282]:
```
{'depth_1_train': 52.22222222222222,
 'depth_1_test': 33.333333333333336,
 'depth_2_train': 52.22222222222222,
 'depth_2_test': 33.333333333333336,
 'depth_3_train': 52.22222222222222,
 'depth_3_test': 33.333333333333336,
 'depth_4_train': 52.22222222222222,
 'depth_4_test': 33.333333333333336,
 'depth_5_train': 52.22222222222222,
 'depth_5_test': 33.333333333333336,
 'depth_6_train': 52.22222222222222,
 'depth_6_test': 33.333333333333336,
 'depth_7_train': 52.22222222222222,
```

```
 'depth_7_test': 33.333333333333336,
 'depth_8_train': 52.22222222222222,
 'depth_8_test': 33.333333333333336}
```

[ ]:

## 3 Ensemble Classifiers

2. Using the data and decision tree algorithm from Problem 1, chose a decision tree depth that does not overfit but achieves some baseline classification performance (but at least depth 4) and apply bagging to the problem.

a). Implement a bagging routine for the decision tree classifier.

[229]:
```
## make an ensemble classifier based on decision trees ##
import scipy as sp
import scipy.stats
def mean_squared_error(x, y):
    y_p = np.asarray(y).reshape(-1)
    return np.mean((x - y_p) ** 2)


def r2_score(x, y):
    """Return R^2 where x and y are array-like."""
    y_p = np.asarray(y).reshape(-1)
    slope, intercept, r_value, p_value, std_err = sp.stats.linregress(x, y_p)
    return r_value ** 2



class BaggedTreeClassifier(object):
    #initializer
    def __init__(self,n_elements=100):
        self.n_elements = n_elements
        self.models     = []

    #destructor
    def __del__(self):
        del self.n_elements
        del self.models

    #private function to make bootstrap samples
    def __make_bootstraps(self,data):
        #initialize output dictionary & unique value count
        dc   = {}
        unip = 0
        #get sample size
```

```
        b_size = data.shape[0]
        #get list of row indexes
        idx = [i for i in range(len(data))]
        #loop through the required number of bootstraps
        for b in range(self.n_elements):
            #obtain boostrap samples with replacement
            sidx   = np.random.choice(idx,replace=True,size=b_size)
            b_samp = data.loc[sidx]
            #compute number of unique values contained in the bootstrap sample
            unip  += len(set(sidx))
            #obtain out-of-bag samples for the current b
            oidx   = list(set(idx) - set(sidx))
            o_samp = np.array([])
            if oidx:
                o_samp = data.loc[oidx]
            #store results
            dc['boot_'+str(b)] = {'boot':b_samp,'test':o_samp}
        #return the bootstrap results
        return(dc)
    #public function to return model parameters
    def get_params(self, deep = False):
        return {'n_elements':self.n_elements}

    #train the ensemble
    def fit(self,X_train,y_train,print_metrics=False):
        #package the input data
        training_data = pd.concat((X_train,y_train),axis=1)
        #make bootstrap samples
        dcBoot = self.__make_bootstraps(training_data)
        #initialise metric arrays
        accs = []
        pres = []
        #recs = np.array([])
        #iterate through each bootstrap sample & fit a model ##
        cls = DecisionTreeClf(max_depth=4)
        for b in dcBoot:
            #make a clone of the model
            model = cls
            #fit a decision tree classifier to the current sample
            model.
↪fit(dcBoot[b]['boot'][["height","age","weight"]],dcBoot[b]['boot']["gender_code"])
            #append the fitted model
            self.models.append(model)
            #compute the predictions on the out-of-bag test set & compute␣
↪metrics
            if dcBoot[b]['test'].size:
```

```
                yp  = model.
↪predict(dcBoot[b]['test'][["height","age","weight"]])
                acc = r2_score(dcBoot[b]['test']["gender_code"],yp)
                pre = mean_squared_error(dcBoot[b]['test']["gender_code"],yp)
                #rec = recall_score(dcBoot[b]['test']["gender_code"],yp)
                #store the error metrics
                accs.append(acc)
                pres.append(pre)
                #recs = np.concatenate((recs,rec.flatten()))
        #compute standard errors for error metrics
        if print_metrics:
            print("Model accuracy: %.2f" % np.mean(accs))
            print("Mean Squared error : %.2f" % np.mean(pres))
            #print("Standard error in recall: %.2f" % np.std(recs))

    #predict from the ensemble
    def predict(self,X):
        #check we've fit the ensemble
        if not self.models:
            print('You must train the ensemble before making predictions!')
            return(None)
        #loop through each fitted model
        predictions = []
        for m in self.models:
            #make predictions on the input X
            yp = m.predict(X)
            #append predictions to storage list
            predictions.append(yp.reshape(-1,1))
        #compute the ensemble prediction
        ypred = np.round(np.mean(np.concatenate(predictions,axis=1),axis=1)).
↪astype(int)
        #return the prediction
        return(ypred)
```

[290]:
```
for n in [10,50,100]:
    bclf=BaggedTreeClassifier(n)
    bclf.
↪fit(train_df[["height","age","weight"]],train_df["gender_code"],print_metrics=True)

    test_df["bpred"]=bclf.predict(test_df[["height","age","weight"]])
#test_df
    correct=test_df[test_df.gender_code==test_df.bpred]
    print(f"Correct Predictions on test set:\n{len(correct)}/{len(test_df)}␣
↪{len(correct)/len(test_df)}")
    pct=100*len(correct)/len(test_df)
    print(pct)
```

```
Model accuracy: 0.05
Mean Squared error : 0.44
Correct Predictions on test set:
10/30 0.3333333333333333
33.333333333333336

/tmp/ipykernel_4105445/809157271.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df["bpred"]=bclf.predict(test_df[["height","age","weight"]])

Model accuracy: 0.05
Mean Squared error : 0.44
Correct Predictions on test set:
10/30 0.3333333333333333
33.333333333333336

/tmp/ipykernel_4105445/809157271.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df["bpred"]=bclf.predict(test_df[["height","age","weight"]])

Model accuracy: 0.05
Mean Squared error : 0.44
Correct Predictions on test set:
10/30 0.3333333333333333
33.333333333333336

/tmp/ipykernel_4105445/809157271.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df["bpred"]=bclf.predict(test_df[["height","age","weight"]])
```

b) Apply boosting 10, 25, and 50 times to the training data. For each of the three cases, evaluate the resulting ensemble classifier on the test data set and compare the error rates for a single classifier with the chosen depth and the three ensemble classifiers. Briefly discuss the results you obtained.

[272]:
```
def I(flag):
    return 1 if flag else 0
def sign(x):
```

```
    return abs(x)/x if x!=0 else 1


class AdaBoost:

    def __init__(self,n_estimators=50):
        self.n_estimators = n_estimators
        self.models = [None]*n_estimators

    def fit(self,X,y):

        X = X
        N = len(y)
        w = np.array([1/N for i in range(N)])

        for m in range(self.n_estimators):

            c = DecisionTreeClf(max_depth=4)
            c.fit(X,y)
            Gm=c.predict

            errM = sum([w[i]*I(all(y!=Gm(X))) \
                        for i in range(N)])/sum(w)

            AlphaM = np.log((1-errM)/errM)

            w = [w[i]*np.exp(AlphaM*I(all(y!=Gm(X))))\
                     for i in range(N)]


            self.models[m] = (AlphaM,Gm)

    def predict(self,X):

        y = 0
        for m in range(self.n_estimators):
            AlphaM,Gm = self.models[m]
            y += AlphaM*Gm(X)
        signA = np.vectorize(sign)
        y = np.where(signA(y)==-1,-1,1)
        return y
```

[287]:
```
for i in [10,50,100]:
    aclf=AdaBoost(n_estimators=i)
    aclf.fit(train_df[["height","age","weight"]],train_df["gender_code"])
    test_df["apred"]=aclf.predict(test_df[["height","age","weight"]])
    #test_df
```

```
    correct=test_df[test_df.gender_code==test_df.apred]
    print(f"Correct Predictions on test set:\n{len(correct)}/{len(test_df)}␣
↪{len(correct)/len(test_df)}")
    pct=100*len(correct)/len(test_df)
    print(pct)
```

```
/tmp/ipykernel_4105445/3947165126.py:28: RuntimeWarning: divide by zero
encountered in double_scalars
  AlphaM = np.log((1-errM)/errM)
/tmp/ipykernel_4105445/3947165126.py:30: RuntimeWarning: invalid value
encountered in double_scalars
  w = [w[i]*np.exp(AlphaM*I(all(y!=Gm(X))))\
/tmp/ipykernel_4105445/3947165126.py:41: RuntimeWarning: invalid value
encountered in multiply
  y += AlphaM*Gm(X)
/tmp/ipykernel_4105445/2504921989.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df["apred"]=aclf.predict(test_df[["height","age","weight"]])
/tmp/ipykernel_4105445/3947165126.py:28: RuntimeWarning: divide by zero
encountered in double_scalars
  AlphaM = np.log((1-errM)/errM)
/tmp/ipykernel_4105445/3947165126.py:30: RuntimeWarning: invalid value
encountered in double_scalars
  w = [w[i]*np.exp(AlphaM*I(all(y!=Gm(X))))\
```

```
Correct Predictions on test set:
21/30 0.7
70.0
```

```
/tmp/ipykernel_4105445/3947165126.py:41: RuntimeWarning: invalid value
encountered in multiply
  y += AlphaM*Gm(X)
/tmp/ipykernel_4105445/2504921989.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df["apred"]=aclf.predict(test_df[["height","age","weight"]])
/tmp/ipykernel_4105445/3947165126.py:28: RuntimeWarning: divide by zero
encountered in double_scalars
  AlphaM = np.log((1-errM)/errM)
/tmp/ipykernel_4105445/3947165126.py:30: RuntimeWarning: invalid value
encountered in double_scalars
```

```
  w = [w[i]*np.exp(AlphaM*I(all(y!=Gm(X))))\
```

```
Correct Predictions on test set:
21/30 0.7
70.0
Correct Predictions on test set:
21/30 0.7
70.0
```

```
/tmp/ipykernel_4105445/3947165126.py:41: RuntimeWarning: invalid value
encountered in multiply
  y += AlphaM*Gm(X)
/tmp/ipykernel_4105445/2504921989.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  test_df["apred"]=aclf.predict(test_df[["height","age","weight"]])
```

[ ]:

[289]:
```
single=DecisionTreeClf(max_depth=4)
single.fit(train_df[["height","age","weight"]],train_df["gender_code"])
test_df["apred"]=single.predict(test_df[["height","age","weight"]])
    #test_df
correct=test_df[test_df.gender_code==test_df.apred]
print(f"Correct Predictions on test set:\n{len(correct)}/{len(test_df)}␣
 ↪{len(correct)/len(test_df)}")
pct=100*len(correct)/len(test_df)
print(pct)
```

[289]:
```
{'col': 'gender_code',
 'index_col': 3,
 'cutoff': 0.4888888888888889,
 'val': 0.0,
 'left': {'val': 0},
 'right': {'val': 1}}
```

```
Correct Predictions on test set:
10/30 0.3333333333333333
33.333333333333336
```

```
/tmp/ipykernel_4105445/914948173.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
test_df["apred"]=single.predict(test_df[["height","age","weight"]])
```

[ ]: