Decision Tree Problems Solutions:

1)

Should see the file,

data_generator.py

The main function is,

```
    def data_Generator(self, k, m, iter = 1):
```

Use the method of weighted average sampling to general random samples

2)

Should see the file,

decision_tree_classifier.py

In the class *DecisionTreeClassifier*

The function for building a decision tree is,

```
def fit_ID3(self, data_tree, data, originaldata, features,
target_attribute_name = "Y", parent_node_class = None):
```

We use ID3 algorithm to build the tree and return the tree. And in the meanwhile, we store the tree and the tree combined with the data set partitioned in each step in the Instance of the Class *DTC* for further use.

The structure of each tree is like:

DTC.tree:

```
{'X1': {0: {'X3': {0.0: 1.0,
                   1.0: {'X4': {0.0: {'X2': {0.0: 1.0,
                                             1.0: 0.0}},
                                1.0: 0.0}}}},
        1: {'X3': {0.0: 0.0, 1.0: {'X2': {0.0: 0.0, 1.0: 1.0}}}}}}
```

DTC.tree_with_data:

```
{'X1': {"data": for root node it is the original data we can denote as D1,
        "child": {
            {0:
                {'X3': {"data": data partitioned from D1 by X1=0 so we get D2,
                        "child": {
                                {0.0: 1.0,
                                 1.0: {'X4': {……
                                ……
```

The function we use to predict the outcome given some X is:

```
def predict(self, data_tree, query, tree):
```

The function we use to calculate the $err_{train}(\hat{f})$ is:

```
def score(self, data):
```

The caller of the Class is in the *main.py.*

```
classifier = DecisionTreeClassifier(4, 30, dataset)
tree = classifier.fit_ID3(classifier.tree_with_data, dataset, dataset,
dataset.columns[:-1])

print("The text format tree is: ")
pprint(tree)
pprint(classifier.tree_with_data)

err_train = classifier.score(dataset)
print("err_train = ", err_train)
```

There are no other things need to specific because I think I have written sufficient annotations for understanding the code.

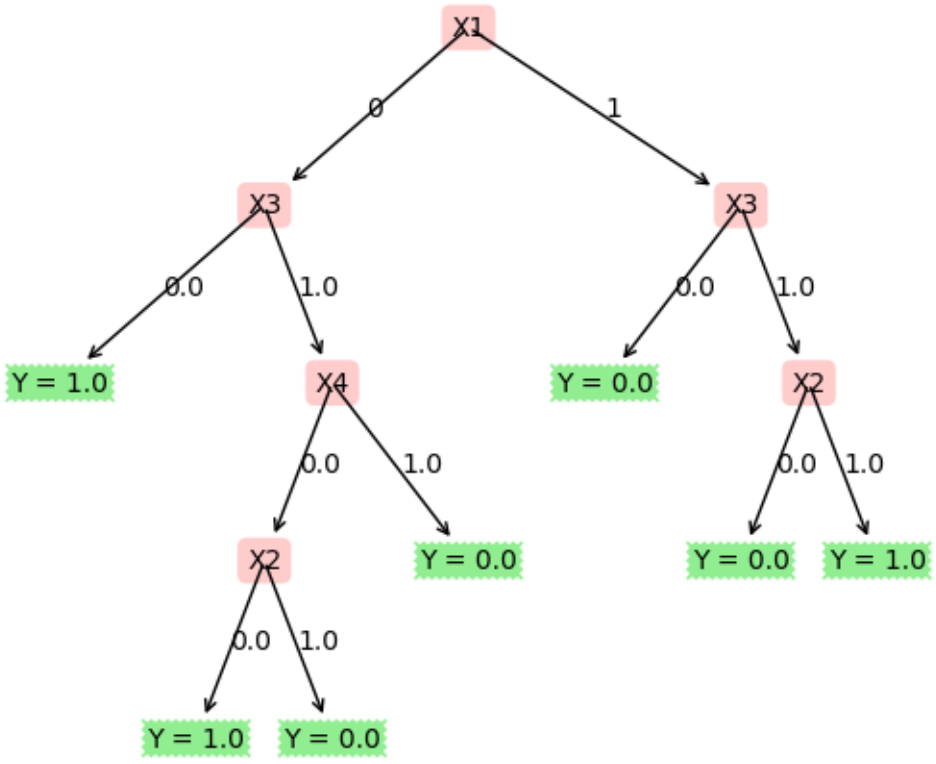
3)

The data is in the file ***data/question3/data_k_4_m_30.csv***

The text format tree is:

{'X1': {0: {'X3': {0.0: 1.0,
1.0: {'X4': {0.0: {'X2': {0.0: 1.0, 1.0: 0.0}}, 1.0: 0.0}}}}

The image is of the tree is:



The ordering of the variables in the decision tree make sense. Because ID3 proceeds by trying to greedily order the variables in terms of information gain in the hopes that his leads to short trees over all. So, only if we follow this order, we can tell that it may be a 'simple' tree that it is the shortest possible tree we can find to best fit the data. So, the order of the variables should not be change.

4)

The function is in file *cal_typical_error.py,* the function is:

```
def test_score(self, k, m, iteration, classifier):
```

The method to call the function is in *main.py*:

```
typical_err = CalTypicalError()
err_test = typical_err.test_score(4, 30, 1000, classifier)
```

For k=4, m=30, we have 1000 iterations that is 30000 data in total, the typical error $err(\hat{f}) = 0.0549$.

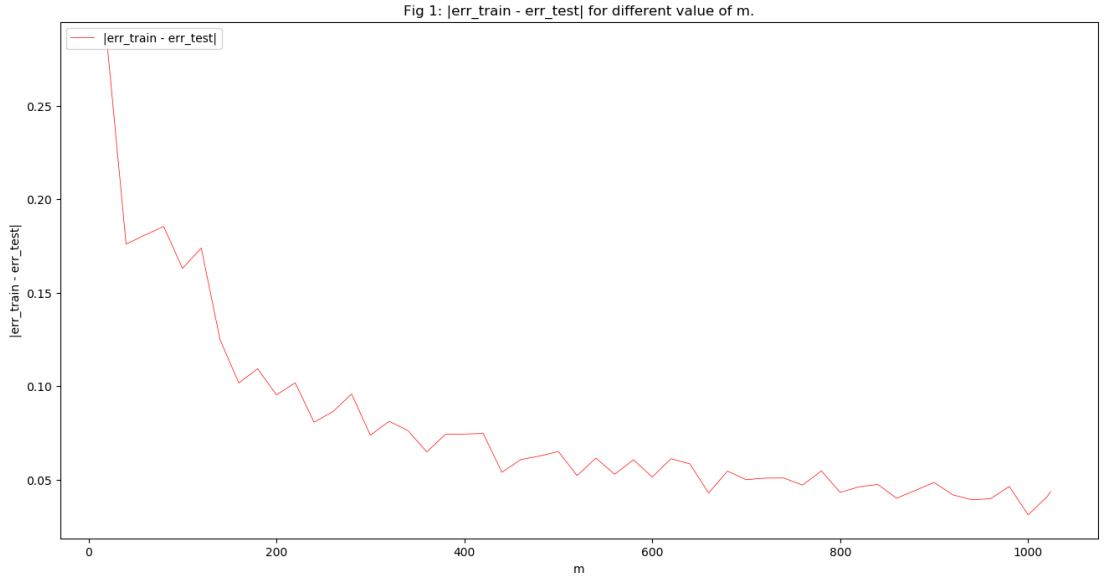
For k=4, m=30, we have 5000 iterations that is 1.5×10^5 data in total, the typical error $err(\hat{f}) = 0.0556533$.

5)

The code is in *question5.py*.

For every m, we know that $|H| = 2^{2^k} = 2^{1024}$, but my computer is not strong enough for so much data, and from 4). we know the error is stable, so I use 4×10^4 data for test.

And the m is from 10 to 1000, set every step 20, and especially run at an m = 1024. Below is the graph.

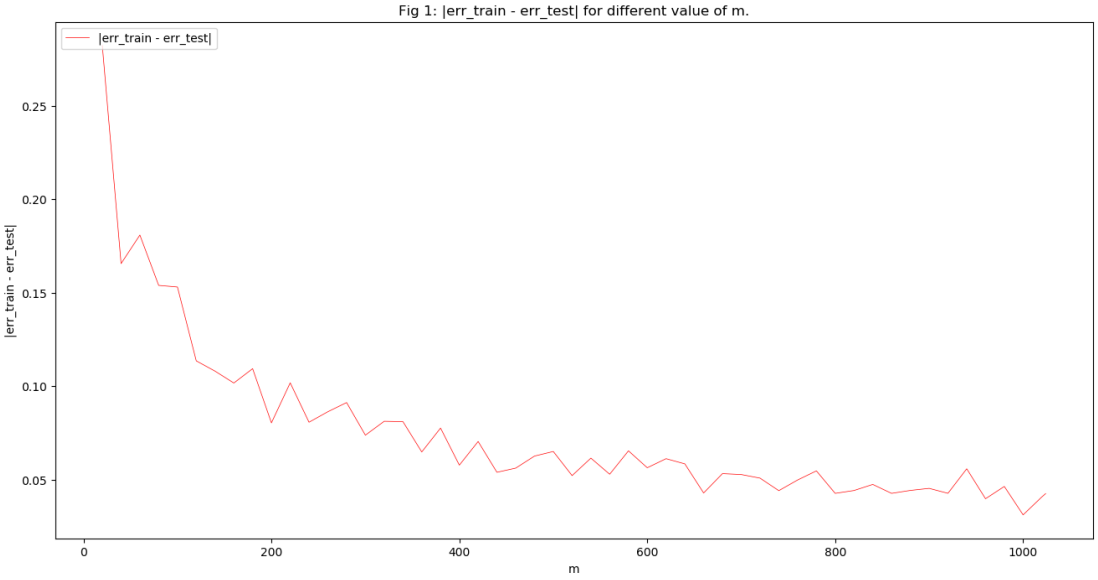


We can see that when m is larger than 400 the value will be less than 0.05 and there is a sharp decrease when m approach 200. And 1024 is what we have stated above. So, we can say that the

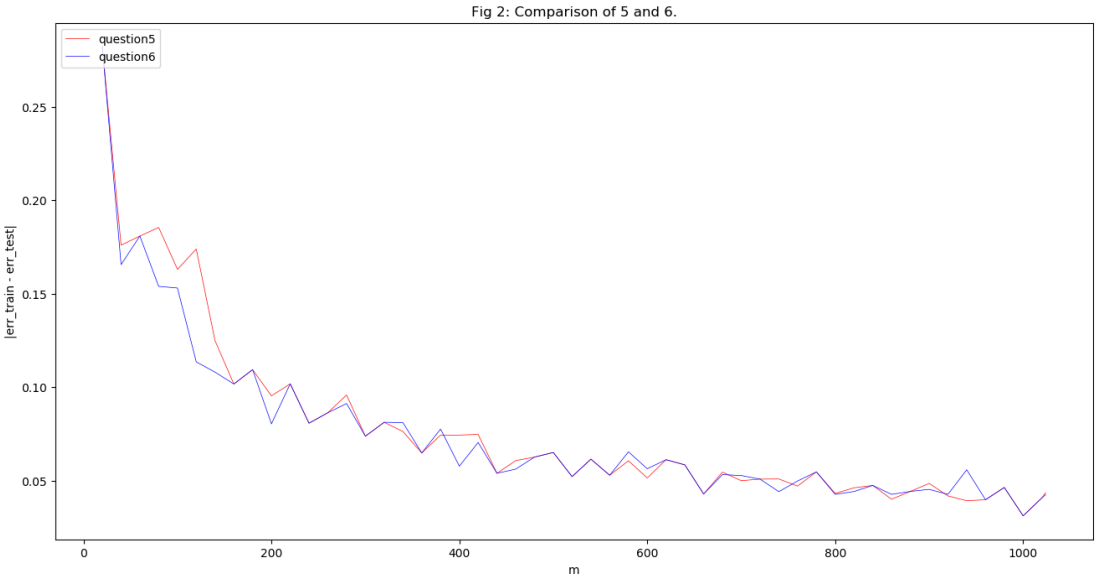
marginal value for good performance of the tree can be m = 400.

6)

I use the gini to general the CART tree and others are the same and use the same dataset to repeat the question 5). Below is the graph.



And use the data we have general in question 5 and 6 (in file ***data/question5.csv*** and ***data/question6.csv***), we have the following figure.



From the figure we can know when m is small the ID3 is worse than CART and when m is getting bigger, the difference between them will be smaller then they will have same performance.