```
import pandas as pd
import numpy as np
import random
from sklearn.ensemble import RandomForestClassifier
import matplotlib.pyplot as plt

train_data = np.array(pd.read_csv('./H2_data_Train.csv', header =
0).values)
test_data = np.array(pd.read_csv('./H2_data_Test.csv', header = 0).values)

X_train = train_data[:, :train_data.shape[1] - 1]
y_train = train_data[:, train_data.shape[1] - 1]
X_test = test_data[:, :test_data.shape[1] - 1]
y_test = test_data[:, test_data.shape[1] - 1]

# Trivial classifier I

correct_count = 0
for y in y_train:
    if random.choice([1,2,3,4]) == y:
        correct_count += 1
print("Trivial classifier I training error:", (len(y_train) -
correct_count) / len(y_train))

correct_count = 0
for y in y_test:
    if random.choice([1,2,3,4]) == y:
        correct_count += 1
print("Trivial classifier I testing error:", (len(y_test) - correct_count)
/ len(y_test))

# Trivial classifier II

fixed_predicted_label = 2

correct_count = 0
for y in y_train:
    if fixed_predicted_label == y:
        correct_count += 1
print("Trivial classifier II training error:", (len(y_train) -
correct_count) / len(y_train))

correct_count = 0
for y in y_test:
    if fixed_predicted_label == y:
        correct_count += 1
print("Trivial classifier II testing error:", (len(y_test) - correct_count)
/ len(y_test))

# Random forest

mean_error_rate_training = []
```

```
mean_error_rate_testing = []
std_error_rate_training = []
std_error_rate_testing = []

for n_estimators in range(1, 101+5, 5):
    error_rate_list_train = []
    error_rate_list_test = []
    for i in range(10):
        random_forest_classifier =
RandomForestClassifier(n_estimators=n_estimators ,max_samples=0.8,
criterion='gini', max_depth=5, bootstrap=True, max_features=7)
        random_forest_classifier.fit(X_train, y_train)

        y_pred = random_forest_classifier.predict(X_train)
        error_count = 0
        for j in range(len(y_train)):
            if y_pred[j] != y_train[j]:
                error_count += 1
        error_rate_list_train.append(error_count / len(y_train))

        y_pred = random_forest_classifier.predict(X_test)
        error_count = 0
        for j in range(len(y_test)):
            if y_pred[j] != y_test[j]:
                error_count += 1
        error_rate_list_test.append(error_count / len(y_test))

    mean_error_rate_training.append(sum(error_rate_list_train) /
len(error_rate_list_train))
    mean_error_rate_testing.append(sum(error_rate_list_test) /
len(error_rate_list_test))
    std_error_rate_training.append(np.std(error_rate_list_train))
    std_error_rate_testing.append(np.std(error_rate_list_test))

# print(mean_error_rate_training, mean_error_rate_testing,
std_error_rate_training, std_error_rate_testing)
print(mean_error_rate_training)
print(mean_error_rate_testing)
print(std_error_rate_training)
print(std_error_rate_testing)

plt.figure()
X = range(1, 101 + 5, 5)
plt.plot(X, mean_error_rate_training)
plt.plot(X, mean_error_rate_testing)
plt.plot(X, std_error_rate_training)
plt.plot(X, std_error_rate_testing)
plt.show()

print("B for min test error rate: ",
X[mean_error_rate_testing.index(min(mean_error_rate_testing))])
print("min test error rate: ",
mean_error_rate_testing[mean_error_rate_testing.index(min(mean_error_rate_t
esting))])
```

```
print("corresponding std: ",
std_error_rate_testing[mean_error_rate_testing.index(min(mean_error_rate_te
sting))])


# Random forest for (iv)

mean_error_rate_training = []
mean_error_rate_testing = []
std_error_rate_training = []
std_error_rate_testing = []

for max_depth in range(2, 25, 2):
    error_rate_list_train = []
    error_rate_list_test = []
    for i in range(10):
        random_forest_classifier = RandomForestClassifier(n_estimators=56
,max_samples=0.8, criterion='gini', max_depth=max_depth, bootstrap=True,
max_features=7)
        random_forest_classifier.fit(X_train, y_train)

        y_pred = random_forest_classifier.predict(X_train)
        error_count = 0
        for j in range(len(y_train)):
            if y_pred[j] != y_train[j]:
                error_count += 1
        error_rate_list_train.append(error_count / len(y_train))

        y_pred = random_forest_classifier.predict(X_test)
        error_count = 0
        for j in range(len(y_test)):
            if y_pred[j] != y_test[j]:
                error_count += 1
        error_rate_list_test.append(error_count / len(y_test))

    mean_error_rate_training.append(sum(error_rate_list_train) /
len(error_rate_list_train))
    mean_error_rate_testing.append(sum(error_rate_list_test) /
len(error_rate_list_test))
    std_error_rate_training.append(np.std(error_rate_list_train))
    std_error_rate_testing.append(np.std(error_rate_list_test))

# print(mean_error_rate_training, mean_error_rate_testing,
std_error_rate_training, std_error_rate_testing)
print(mean_error_rate_training)
print(mean_error_rate_testing)
print(std_error_rate_training)
print(std_error_rate_testing)

plt.figure()
X = range(2, 25, 2)
plt.plot(X, mean_error_rate_training)
plt.plot(X, mean_error_rate_testing)
plt.plot(X, std_error_rate_training)
```

```
plt.plot(X, std_error_rate_testing)
plt.show()

print("B for min test error rate: ",
X[mean_error_rate_testing.index(min(mean_error_rate_testing))])
print("min test error rate: ",
mean_error_rate_testing[mean_error_rate_testing.index(min(mean_error_rate_t
esting))])
print("corresponding std: ",
std_error_rate_testing[mean_error_rate_testing.index(min(mean_error_rate_te
sting))])
```