a) Using null model report avg MSE of the abalone dataset.

Import modules and locate dataset

```
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

import os

os.chdir("C:/Users/rique/Downloads")
```

per the homework; first 7 variables are predictors X, and the 8th variable is our response y.

use test_size of 0.15 to divide each random split into 85% training and 15% testing

null model really just means to use the numpy mean to "fit" our model. and full_like to make our predictions.

```
abalone_data = pd.read_csv("abalone.csv")

X = abalone_data.iloc[:, :7]
y = abalone_data.iloc[:, 7]

splits = np.arange(20)+1

train_mse_list = []
test_mse_list = []

for i in range(splits.shape[0]):
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.15)

    null_model = np.mean(y_train)

    y_pred = np.full_like(y_train, null_model)
    yTest_pred = np.full_like(y_test, null_model)

    train_mse = mean_squared_error(y_train, y_pred)
```

```
    train_mse_list.append(train_mse)
    test_mse = mean_squared_error(y_test, yTest_pred)
    test_mse_list.append(test_mse)

null_train_avg = np.mean(train_mse_list)
null_test_avg = np.mean(test_mse_list)
```

We have our results. Simply output them.

```
print(f"Train Average MSE: {null_train_avg}")
print(f"Test Average MSE: {null_test_avg}")
```

```
Train Average MSE: 11.162862778247392
Test Average MSE: 11.800637958532695
```

b) Using OLS regression, analytically (gross) report avg MSE and R-squared.

the formula is Y = α + βX + ε

where Y is our response

where α is our intercept/constant

where β are the coefficients of our predictors represented by X

and where ε is the error

Treat β as our "fit" method, and can be found with (X^TX)^-1 * X^Ty as shown in slide 14 of the Regression lecture

```
from sklearn.metrics import r2_score

OLS_train_mse_list = []
OLS_test_mse_list = []
OLS_train_r2_list = []
OLS_test_r2_list = []

lambda_value = 0.0001
```

```
for i in range(splits.shape[0]):
   X_train, X_test, y_train, y_test = train_test_split(X,
y,test_size=0.15)

   #all of this is our "fit" method for OLS or β as described above.
   X_transpose = X_train.T
   XTX = np.dot(X_transpose, X_train)
   matrix = lambda_value * np.identity(XTX.shape[0])
   XTX_lambda = XTX + matrix
   XTy = np.dot(X_transpose, y_train)
   coefficients = np.linalg.solve(XTX_lambda, XTy)

   #from here, simply use dot for our predictions, and find our
results as per usual
   OLS_y_pred = np.dot(X_train, coefficients)
   OLS_yTest_pred = np.dot(X_test, coefficients)
   train_mse = mean_squared_error(y_train, OLS_y_pred)
   OLS_train_mse_list.append(train_mse)
   test_mse = mean_squared_error(y_test, OLS_yTest_pred)
   OLS_test_mse_list.append(test_mse)

   r2_train = r2_score(y_train, OLS_y_pred)
   OLS_train_r2_list.append(r2_train)
   r2_test = r2_score(y_test, OLS_yTest_pred)
   OLS_test_r2_list.append(r2_test)

OLS_train_avg = np.mean(OLS_train_mse_list)
OLS_test_avg = np.mean(OLS_test_mse_list)
OLS_r2_train_avg = np.mean(OLS_test_r2_list)
OLS_r2_test_avg = np.mean(OLS_test_r2_list)
```

We have our results for the OLS regression. Simply output them.

```
print(f"Train Average MSE: {OLS_train_avg}")
print(f"Test Average MSE: {OLS_test_avg}")
print(f"Train Average r-squared: {OLS_r2_train_avg}")
print(f"Test Average r-squared: {OLS_r2_test_avg}")
```

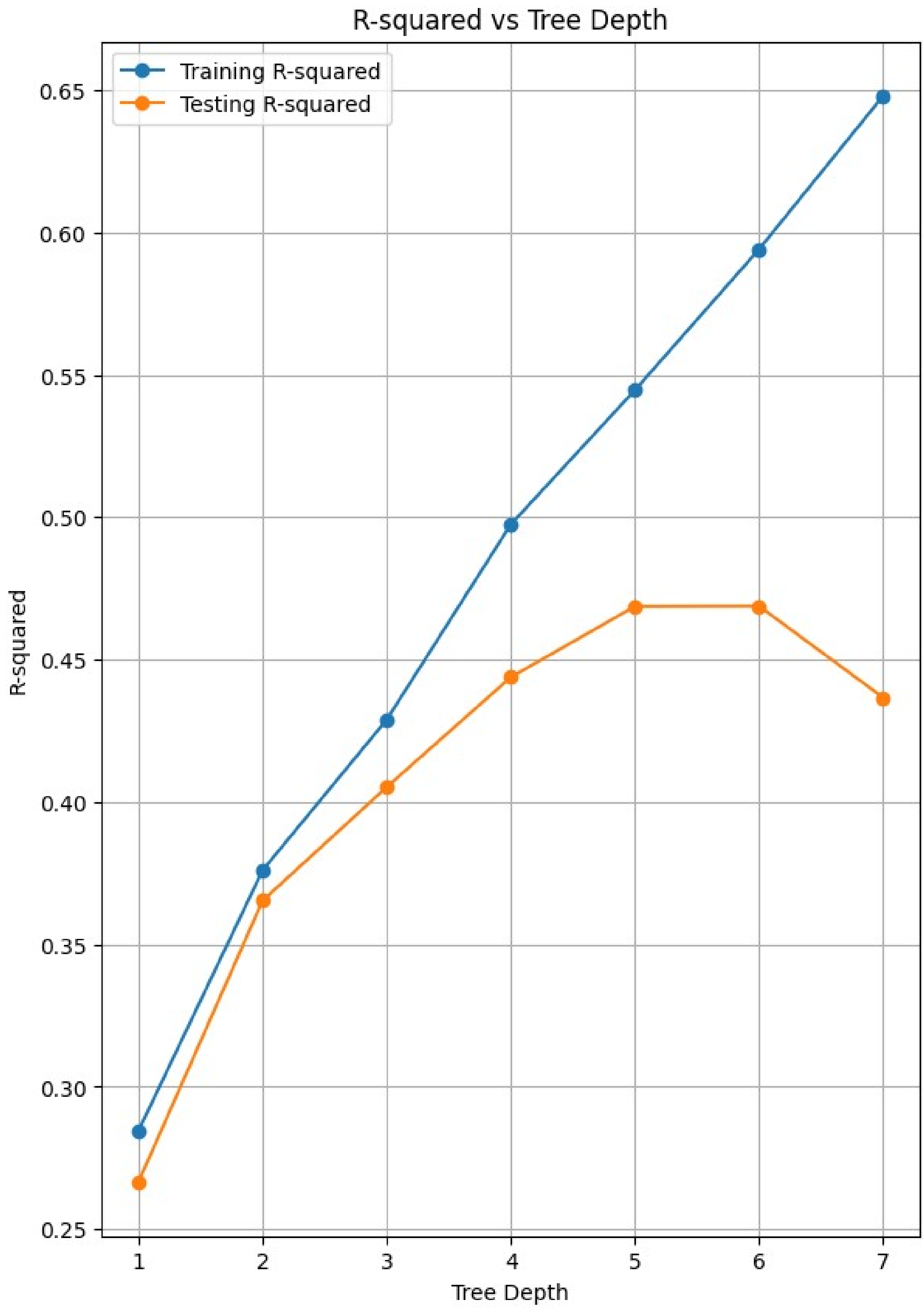
```
Train Average MSE: 5.051139932259096
Test Average MSE: 5.083171029726857
Train Average r-squared: 0.49890446080239714
Test Average r-squared: 0.49890446080239714
```

c) Use a regression tree, of max depth 7. Find the avg MSE and R-squared. Plot on separate graphs. For the MSE one, also plot the null MSE as horizontal line.

```
from sklearn.tree import DecisionTreeRegressor
max_depth = np.arange(7)+1
regTree_train_mse_avg = []
regTree_test_mse_avg = []
regTree_train_r2_avg = []
regTree_test_r2_avg = []
#we should only care for the test results from the null model
null_mse = null_test_avg
for depth in max_depth:
 #since we have to do this per depth, i need another set of my lists
 regTree_train_mse_list = []
 regTree_test_mse_list = []
 regTree_train_r2_list = []
 regTree_test_r2_list = []

 for i in range(splits.shape[0]):
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.15)

    dt = DecisionTreeRegressor(max_depth = depth)
    dt.fit(X_train, y_train)

    dt_y_pred = dt.predict(X_train)
    dt_yTest_pred = dt.predict(X_test)
    train_mse = mean_squared_error(y_train, dt_y_pred)
    regTree_train_mse_list.append(train_mse)
    test_mse = mean_squared_error(y_test, dt_yTest_pred)
    regTree_test_mse_list.append(test_mse)

    r2_train = r2_score(y_train, dt_y_pred)
    regTree_train_r2_list.append(r2_train)
    r2_test = r2_score(y_test, dt_yTest_pred)
    regTree_test_r2_list.append(r2_test)

 #find the average for this specific depth. Annoying thing about
python, i miss my brackets for clearer end of for-loops.
 regTree_train_mse_avg.append(np.mean(regTree_train_mse_list))
 regTree_test_mse_avg.append(np.mean(regTree_test_mse_list))
 regTree_train_r2_avg.append(np.mean(regTree_train_r2_list))
 regTree_test_r2_avg.append(np.mean(regTree_test_r2_list))
```

We have our results. Now plot. Here is R-squared vs Tree Depth

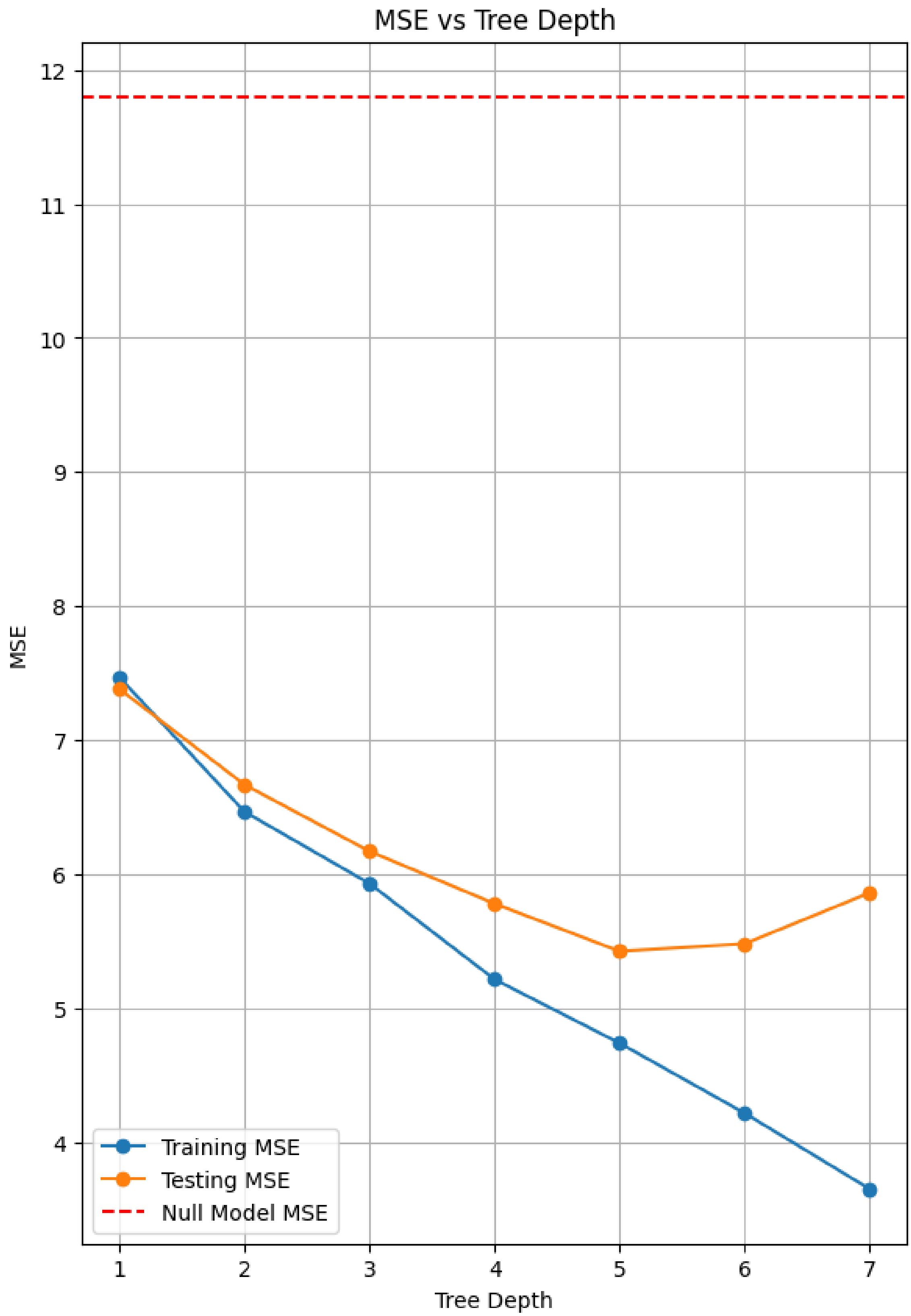
```
import matplotlib.pyplot as plt

plt.figure(figsize=(15, 10))
plt.subplot(1, 2, 1)
plt.plot(max_depth, regTree_train_r2_avg, label='Training R-squared',
marker='o')
plt.plot(max_depth, regTree_test_r2_avg, label='Testing R-squared',
marker='o')
plt.xlabel('Tree Depth')
plt.ylabel('R-squared')
plt.title('R-squared vs Tree Depth')
plt.legend()
plt.grid(True)
plt.show()
```



R-squared vs Tree Depth
Training R-squared
Testing R-squared
R-squared
Tree Depth
0.25
0.30
0.35
0.40
0.45
0.50
0.55
0.60
0.65
1
2
3
4
5
6
7

Here is MSE vs Tree Depth & Null MSE as the horizontal line.

```
plt.figure(figsize=(15, 10))
plt.subplot(1, 2, 2)
plt.plot(max_depth, regTree_train_mse_avg, label='Training MSE',
marker='o')
plt.plot(max_depth, regTree_test_mse_avg, label='Testing MSE',
marker='o')
plt.axhline(y=null_mse, color='r', linestyle='--', label='Null Model
MSE')
plt.xlabel('Tree Depth')
plt.ylabel('MSE')
plt.title('MSE vs Tree Depth')
plt.legend()
plt.grid(True)
plt.show()
```


MSE vs Tree Depth
MSE
Tree Depth
12
11
10
9
8
7
6
5
4
1
2
3
4
5
6
7
Training MSE
Testing MSE
Null Model MSE

e) Using random forest regression, and k tree values of [10,30,100,300] report avg R-squared and MSE.

```
from sklearn.ensemble import RandomForestRegressor

k_values = np.array([10,30,100,300])

rForest_train_mse_avg = []
rForest_test_mse_avg = []
rForest_train_r2_avg = []
rForest_test_r2_avg = []

for k in k_values:
 #since we have to do this per k value tree, i need another set of my
lists
 rForest_train_mse_list = []
 rForest_test_mse_list = []
 rForest_train_r2_list = []
 rForest_test_r2_list = []

 for i in range(splits.shape[0]):
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.15)

    rf = RandomForestRegressor(n_estimators=k, random_state=1000)
    rf.fit(X_train, y_train)

    rf_y_pred = rf.predict(X_train)
    rf_yTest_pred = rf.predict(X_test)
    train_mse = mean_squared_error(y_train, rf_y_pred)
    rForest_train_mse_list.append(train_mse)
    test_mse = mean_squared_error(y_test, rf_yTest_pred)
    rForest_test_mse_list.append(test_mse)

    r2_train = r2_score(y_train, rf_y_pred)
    rForest_train_r2_list.append(r2_train)
    r2_test = r2_score(y_test, rf_yTest_pred)
    rForest_test_r2_list.append(r2_test)

 #find the average for this specific k value tree. Annoying thing
about python, i miss my brackets for clearer end of for-loops.

 rForest_train_mse_avg.append(np.mean(rForest_train_mse_list))
 print(f"K tree: {k}'s Train Average MSE:
{np.mean(rForest_train_mse_list)}")
 rForest_test_mse_avg.append(np.mean(rForest_test_mse_list))
```

```
 print(f"K tree: {k}'s Test Average MSE:
{np.mean(rForest_test_mse_list)}")
 rForest_train_r2_avg.append(np.mean(rForest_train_r2_list))
 print(f"K tree: {k}'s Train Average R-squared:
{np.mean(rForest_train_r2_list)}")
 rForest_test_r2_avg.append(np.mean(rForest_test_r2_list))
 print(f"K tree: {k}'s Test Average R-squared:
{np.mean(rForest_test_r2_list)}\n")
```

```
K tree: 10's Train Average MSE: 0.9277744435052127
K tree: 10's Test Average MSE: 5.219666666666666
K tree: 10's Train Average R-squared: 0.9104926141696714
K tree: 10's Test Average R-squared: 0.5014526683460753

K tree: 30's Train Average MSE: 0.7340138380138379
K tree: 30's Test Average MSE: 4.969228070175438
K tree: 30's Train Average R-squared: 0.929293037146639
K tree: 30's Test Average R-squared: 0.5223710650338125

K tree: 100's Train Average MSE: 0.6681857748661595
K tree: 100's Test Average MSE: 4.864031323763955
K tree: 100's Train Average R-squared: 0.9356508478934134
K tree: 100's Test Average R-squared: 0.5313308848994536

K tree: 300's Train Average MSE: 0.652961310384772
K tree: 300's Test Average MSE: 4.692826666666666
K tree: 300's Train Average R-squared: 0.9371328316773815
K tree: 300's Test Average R-squared: 0.5476798914854717
```