Problem – 2

NOTE:

For testing purposes, basically to run my .py file , I have the following within my function file to call my main().

```
#including for testing purposes, basically to run it.
if __name__ == '__main__':
    main()
```

It is not commented out because I was not sure whether it would be required for grading or not.

Request you to comment it out before running my program , if it is not required and may possibly throw off the autograder.

Thank you.

Question 1: Complete the function normalize to normalize the feature matrix X. The nine columns of this matrix, X = [x1 x2 · · · x9], must each be normalized to have a mean of 0 and a standard deviation of 1.

Answer – 1:

```
def normalize(X):

    #fill in
    mean = []
    std = []
    mean = np.mean(X,axis = 0)
    std = np.std(X,axis = 0)
    X = (X - mean) / std
    return X
```

Question-2: Define the range of λ to test in main as 0.1, 0.2, ..., 1.0, 1.5, ..., 10.0, 11.0, ..., 100. This type of logarithmic scale, moving from a smaller to a larger increment, is common for regularization.

Answer-2:

```
#Define the range of lambda to test
one = 0
two = 1.5
three = 11
first = []
second = []
third = []

while one <= 1:
    first.append(one)
    one = one + 0.1

while two <= 10:
    second.append(two)
    two = two + 0.5

while three <= 100:
    third.append(three)
    three = three + 1

lmbda = first + second + third #fill in
```

Question -3: Complete the function train model to fit a ridge regression model with regularization parameter λ = l on a training dataset Xtrain, ytrain. You may use the linear model.Ridge class in sklearn to do this. Note that the partition of the training and testing set has already been done for you.

Answer-3:

```
#Output: model, a numpy object containing the trained model.
def train_model(X,y,l):

    #fill in
    model = linear_model.Ridge(alpha = l, fit_intercept = True)
    model.fit(X,y)
    return model
```

Question-4: Complete the function error to calculate the mean squared error of the model on a testing dataset Xtest, ytest.

Answer -4:

```
#Output: mse, the mean squared error
def error(X,y,model):

    #Fill in
    ypredict = model.predict(X)
    mse = mean_squared_error(y,ypredict)
    return mse
```

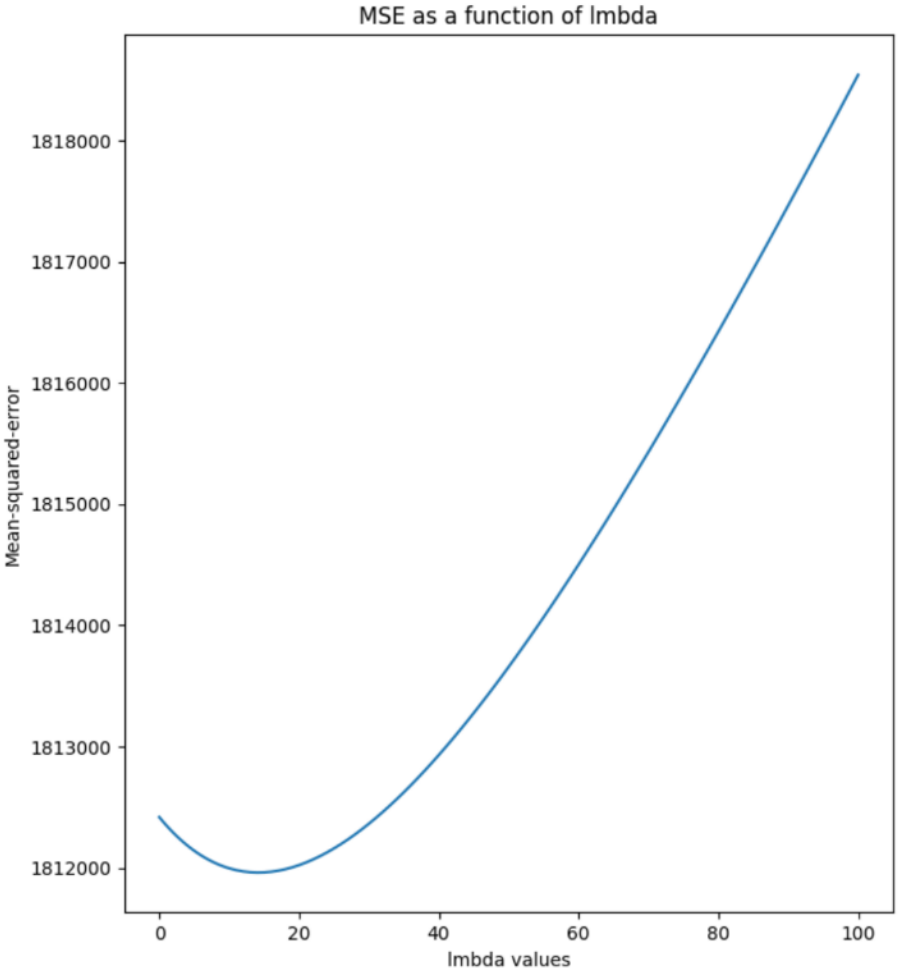
Question – 5: Complete the code in main for plotting the mean squared error as a function of λ, and for finding the model and mse corresponding to the best lmbda. Be sure to include a title and axes labels with your plot.

Answer-5:

Code Snippet for Plotting:

```
def main():
    #Importing dataset
    diamonds = pd.read_csv('diamonds.csv')

    #Feature and target matrices
    X = diamonds[['carat', 'depth', 'table', 'x', 'y', 'z', 'clarity', 'cut', 'color']]
    y = diamonds[['price']]

    #Normalizing X
    X = normalize(X)

    #Training and testing split, with 25% of the data reserved as the test set
    [X_train, X_test, y_train, y_test] = train_test_split(X, y, test_size=0.25, random_state=101)

    #Define the range of lambda to test
    one = 0
    two = 1.5
    three = 11
    first = []
    second = []
    third = []

    while one <= 1:
        first.append(one)
        one = one + 0.1

    while two <= 10:
        second.append(two)
        two = two + 0.5

    while three <= 100:
        third.append(three)
        three = three + 1

    lmbda = first + second + third #fill in

    MODEL = []
    MSE = []
    for l in lmbda:
        #Train the regression model using a regularization parameter of l
        model = train_model(X_train,y_train,l)

        #Evaluate the MSE on the test set
        mse = error(X_test,y_test,model)

        #Store the model and mse in lists for further processing
        MODEL.append(model)
        MSE.append(mse)

    #Plot the MSE as a function of lmbda
    plt.plot(lmbda,MSE) #fill in
    plt.title('MSE as a function of lmbda')
    plt.xlabel('lmbda values')
    plt.ylabel('Mean-squared-error')
    plt.show()

    #Find best value of lmbda in terms of MSE
    ind = MSE.index(min(MSE))#fill in
    [lmda_best,MSE_best,model_best] = [lmbda[ind],MSE[ind],MODEL[ind]]
    print('Best lambda tested is ' + str(lmda_best) + ', which yields an MSE of ' + str(MSE_best))

    X_test = [0.25,60,55,4,3,2,5,3,3]
    X_test = normalize(X_test)
    coeffs = model_best.coef_[0]
    intercept =model_best.intercept_[0]

    predictedvalue = 0
    for i in range (len(X_test)):
        predictedvalue = predictedvalue + X_test[len(X_test)-i-1]*coeffs[i]
    predictedvalue = predictedvalue + intercept
    print("The predicted y value is: " + str(predictedvalue))
    return model_best
```

Plot obtained:



Answer Obtained after completing all the function is:

```
Unknown), minor code: 20
Best lambda tested is 14, which yields an MSE of 1811961.4669904104
```

Question – 6: Using the coefficients (and intercept) β = (c9, c8, · · · c0) from the returned model best, write out the equation ˆy(x) = c9x1 + c8x2 + · · · + c1x9 + c0 of your fitted model for a sample x. What is the predicted price ˆy for a 0.25 carat, 3 cut, 3 color, 5 clarity, 60 depth, 55 table, 4 x, 3 y, 2 z diamond?

Answer – 6:

```
X_test = [0.25,60,55,4,3,2,5,3,3]
X_test = normalize(X_test)
coeffs = model_best.coef_[0]
intercept =model_best.intercept_[0]

predictedvalue = 0
for i in range (len(X_test)):
    predictedvalue = predictedvalue + X_test[len(X_test)-i-1]*coeffs[i]
predictedvalue = predictedvalue + intercept
print("The predicted y value is: " + str(predictedvalue))
return model_best
```

Equation obtained :

Price(y) = (-458.64762287) * carat + (74.36809593) * depth + (501.75988087) * table + (-68.47734902) * x + (222.31481412)* y + (-1333.9900337)* z + (-208.10437575)* clarity + (201.39843689)* cut + (5109.53168875)* color

The predicted price ˆy for a 0.25 carat, 3 cut, 3 color, 5 clarity, 60 depth, 55 table, 4 x, 3 y, 2 z diamond is shown in the image below. This output was obtained using the code snippet above.

```
The predicted y value is: 3435.4267170717444
```

Therefore, we can say that the predicted price ˆy for a 0.25 carat, 3 cut, 3 color, 5 clarity, 60 depth, 55 table, 4 x, 3 y, 2 z diamond is 3435.4267170717444