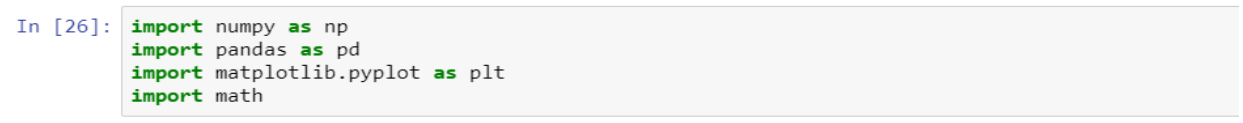
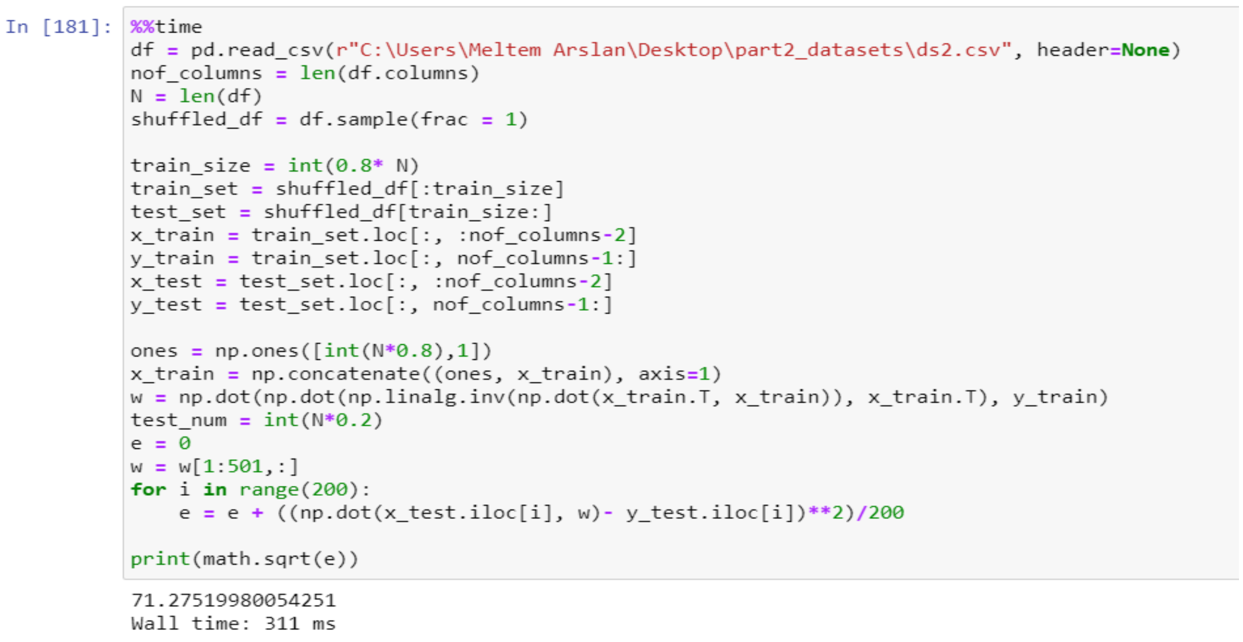
Data set 1 has 1000 samples with 100 independent variables, data set 2 has 1000 samples with 500 independent variables. Since the number of features(independent variables) increases, the time to complete step2 is greater than the time to complete step1. Since the data size is the same, time to complete step3 is nearly the same with the time to complete step2.

I set the lambda value to 500. Although I did not implement cross validation on the submitted python file, I calculated rms error with the following code in another notebook by splitting data into training set and test set and changing the lambda value manually. For lambda values 0.5, 500, 5000, I get the Erms values ~68, ~50, ~75, respectively. So, I used 500 as the lambda value for regularization.



```
In [26]:  import numpy as np
          import pandas as pd
          import matplotlib.pyplot as plt
          import math
```



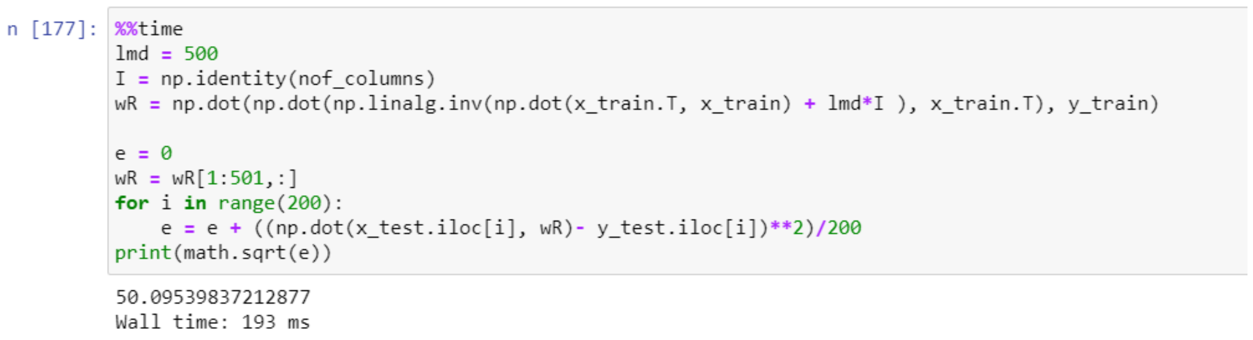
```
In [181]: %%time
          df = pd.read_csv(r"C:\Users\Meltem Arslan\Desktop\part2_datasets\ds2.csv", header=None)
          nof_columns = len(df.columns)
          N = len(df)
          shuffled_df = df.sample(frac = 1)

          train_size = int(0.8* N)
          train_set = shuffled_df[:train_size]
          test_set = shuffled_df[train_size:]
          x_train = train_set.loc[:, :nof_columns-2]
          y_train = train_set.loc[:, nof_columns-1:]
          x_test = test_set.loc[:, :nof_columns-2]
          y_test = test_set.loc[:, nof_columns-1:]

          ones = np.ones([int(N*0.8),1])
          x_train = np.concatenate((ones, x_train), axis=1)
          w = np.dot(np.dot(np.linalg.inv(np.dot(x_train.T, x_train)), x_train.T), y_train)
          test_num = int(N*0.2)
          e = 0
          w = w[1:501,:]
          for i in range(200):
              e = e + ((np.dot(x_test.iloc[i], w)- y_test.iloc[i])**2)/200

          print(math.sqrt(e))

          71.27519980054251
          Wall time: 311 ms
```



```
n [177]:  %%time
          lmd = 500
          I = np.identity(nof_columns)
          wR = np.dot(np.dot(np.linalg.inv(np.dot(x_train.T, x_train) + lmd*I ), x_train.T), y_train)

          e = 0
          wR = wR[1:501,:]
          for i in range(200):
              e = e + ((np.dot(x_test.iloc[i], wR)- y_test.iloc[i])**2)/200
          print(math.sqrt(e))

          50.09539837212877
          Wall time: 193 ms
```