We kernelize ridge regression and train on the auto_mpg dataset. Code is as follows:

```
“import numpy as np
import pandas as pd
from sklearn.svm import LinearSVC
from sklearn.metrics import mean_squared_error
from math import sqrt

lamb = 1

dftr=pd.read_csv('auto_mpg_train.csv', sep=' ',header=None)
datavalstr=dftr.values
traindata=datavalstr.astype(np.float)
trainlabels = traindata[:,0]
trainfeatures = traindata[:,1:]

dftest=pd.read_csv('auto_mpg_test.csv', sep=' ',header=None)
datavalstest=dftest.values
testdata=datavalstest.astype(np.float)
testlabels = testdata[:,0]
testfeatures = testdata[:,1:]

Preds1 = np.zeros((testlabels.shape[0], 1))
Preds2 = np.zeros((testlabels.shape[0], 1))
Preds3 = np.zeros((testlabels.shape[0], 1))
Preds4 = np.zeros((testlabels.shape[0], 1))

def k1(x):
 K = np.zeros((x.shape[0], x.shape[0]))
 for i in range(x.shape[0]):
  for j in range(x.shape[0]):
   K[i][j] = (np.dot(x[i],x[j]) + 1)**2
 return K

def k2(x):
 K = np.zeros((x.shape[0], x.shape[0]))
 for i in range(x.shape[0]):
  for j in range(x.shape[0]):
   K[i][j] = (np.dot(x[i],x[j]) + 1)**4
 return K

def k3(x):
 K = np.zeros((x.shape[0], x.shape[0]))
 for i in range(x.shape[0]):
  for j in range(x.shape[0]):
   K[i][j] = (np.dot(x[i],x[j]) + 1)**8
 return K

def k4(x):
```

```
 K = np.zeros((x.shape[0], x.shape[0]))
 for i in range(x.shape[0]):
  for j in range(x.shape[0]):
   K[i][j] = np.exp(-1* (np.linalg.norm(x[i]-x[j])**2) / 2)
 return K

def k1Prod(x, y):
 K = np.zeros(x.shape[0])
 for i in range(x.shape[0]):
  K[i] = (np.dot(x[i], y) + 1)**2
 return K

def k2Prod(x, y):
 K = np.zeros(x.shape[0])
 for i in range(x.shape[0]):
   K[i] = (np.dot(x[i], y) + 1)**4
 return K

def k3Prod(x, y):
 K = np.zeros(x.shape[0])
 for i in range(x.shape[0]):
   K[i] = (np.dot(x[i], y) + 1)**8
 return K

def k4Prod(x, y):
 K = np.zeros(x.shape[0])
 for i in range(x.shape[0]):
   K[i] = np.exp(-1* (np.linalg.norm(x[i]-y)**2) / 2)
 return K

K1 = k1(trainfeatures)
K2 = k2(trainfeatures)
K3 = k3(trainfeatures)
K4 = k4(trainfeatures)

for i in range(testlabels.shape[0]):
 Preds1[i] = (1 / lamb) * (np.transpose(trainlabels) @ (np.identity(trainlabels.shape[0]) - (K1 @
np.linalg.inv((lamb * np.identity(trainlabels.shape[0]) + K1)))) @ (k1Prod(trainfeatures,
testfeatures[i,:])))
for i in range(testlabels.shape[0]):
 Preds2[i] = (1 / lamb) * (np.transpose(trainlabels) @ (np.identity(trainlabels.shape[0]) - (K2 @
np.linalg.inv((lamb * np.identity(trainlabels.shape[0]) + K2)))) @ (k2Prod(trainfeatures,
testfeatures[i,:])))
for i in range(testlabels.shape[0]):
 Preds3[i] = (1 / lamb) * (np.transpose(trainlabels) @ (np.identity(trainlabels.shape[0]) - (K3 @
np.linalg.inv((lamb * np.identity(trainlabels.shape[0]) + K3)))) @ (k3Prod(trainfeatures,
testfeatures[i,:])))
for i in range(testlabels.shape[0]):
```

```
 Preds4[i] = (1 / lamb) * (np.transpose(trainlabels) @ (np.identity(trainlabels.shape[0]) - (K4 @ np.linalg.inv((lamb * np.identity(trainlabels.shape[0]) + K4)))) @ (k4Prod(trainfeatures, testfeatures[i,:])))

Error1=sqrt(mean_squared_error(testlabels, Preds1))
Error2=sqrt(mean_squared_error(testlabels, Preds2))
Error3=sqrt(mean_squared_error(testlabels, Preds3))
Error4=sqrt(mean_squared_error(testlabels, Preds4))

print(“Error 1 is: ”)
print(Error1)
print(“Error 2 is: ”)
print(Error2)
print(“Error 3 is: ”)
print(Error3)
print(“Error 4 is: ”)
print(Error4)

”
```

Our output is:

“Error 1 is:
2.907323802187438
Error 2 is:
2.906433119938579
Error 3 is:
3.8079908461433214
Error 4 is:
3.1954742162078924”