from sklearn.linear\_model import LinearRegression

from sklearn.svm import SVR

import matplotlib.pyplot as plt

import numpy as np

FILE\_NAME = 'oil\_price.txt'

feature\_list = []

target\_list = []

# Read days as feature, and price as target from an input file

with open(FILE\_NAME, 'r') as f:

    for line in f.readlines():

        day, price = line.strip().split(' ')

        feature\_list.append([int(day)])

        # Support only integer, so, multiply by 100

        target\_list.append(int(float(price) \* 100))

# Create model

linear = LinearRegression().fit(feature\_list, target\_list)

svr\_rbf = SVR(kernel='rbf', C=1e3, gamma=0.0001).fit(feature\_list, target\_list)

# Extend a number of days for forecasting the future

last\_day = len(feature\_list)

for i in range(1, 366):

    feature\_list.append([last\_day + i])

    target\_list.append(np.nan)

# Convert back to float, so, divide every element by 100

linear\_pred = list(map(lambda x: float(x) / 100, linear.predict(feature\_list)))

svr\_rbf\_pred = list(map(lambda x: float(x) / 100, svr\_rbf.predict(feature\_list)))

target = list(map(lambda x: float(x) / 100, target\_list))

# Display the prediction at 1 day, 30 days, and 365 days after the last available data

print('==== Linear regression prediction ====')

print(' - 1 day : ', linear\_pred[last\_day])

print(' - 30 days : ', linear\_pred[last\_day + 29])

print(' - 365 days : ', linear\_pred[last\_day + 364])

print('==== Support vector regression prediction ====')

print(' - 1 day : ', svr\_rbf\_pred[last\_day])

print(' - 30 days : ', svr\_rbf\_pred[last\_day + 29])

print(' - 365 days : ', svr\_rbf\_pred[last\_day + 364])

# Plot data out

plt.hold('on')

plt.plot(feature\_list, target, color='black', label='Stock Price')

plt.plot(feature\_list, linear\_pred, color='blue', label='Linear Regressoin')

plt.plot(feature\_list, svr\_rbf\_pred, color='red', label='Support Vector Regression RBF')

plt.xlabel('A number of days since Jan 1, 2016')

plt.ylabel('Oil Price (USD)')

plt.gca().set\_xlim(left = 0)

plt.gca().set\_xlim(right = 1000)

plt.gca().set\_ylim(bottom = 0)

plt.xticks(np.arange(0, 1001, 60))

plt.legend()

plt.show()