Chunyuan Shen

ID:801322013

Homework 4

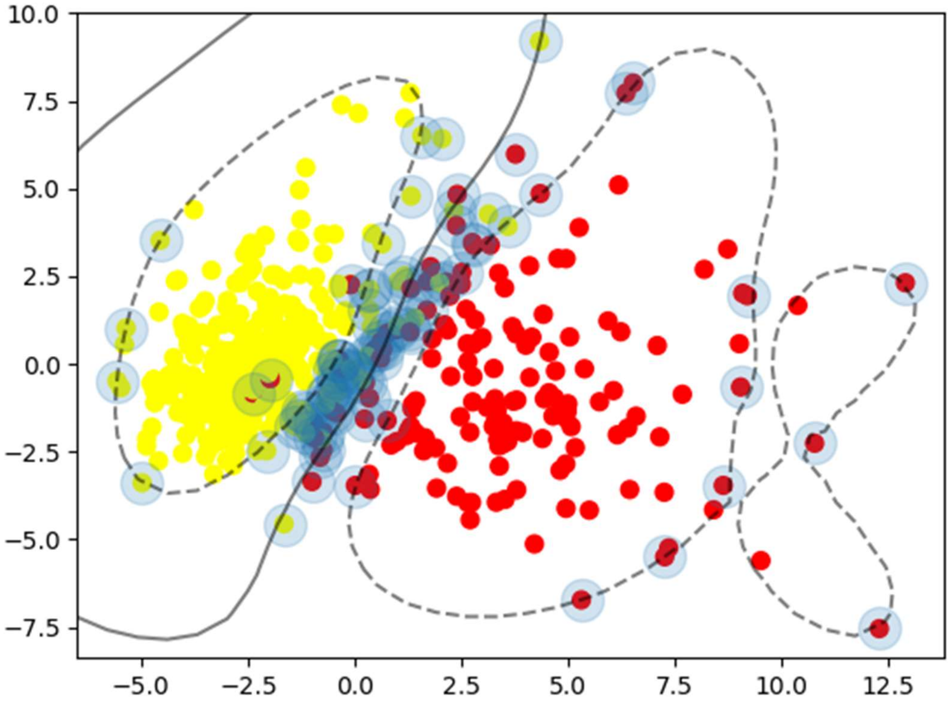
https://github.com/cryyin/ECGR5105/tree/main/homework4

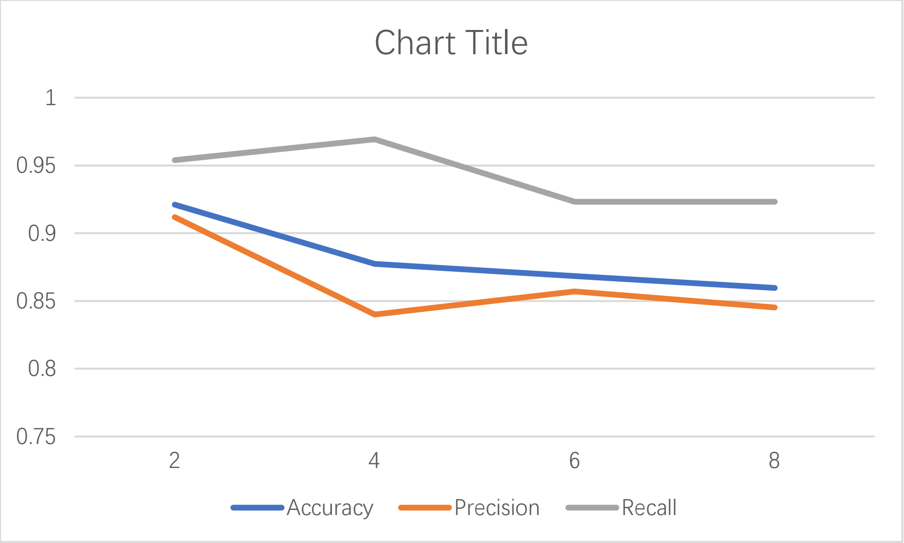
Problem 1 (50pts):

Use the cancer dataset to build an SVM classifier to classify the type of cancer (Malignant vs. benign). Use the PCA feature extraction for your training. Perform N number of independent training (N=1, …, K).

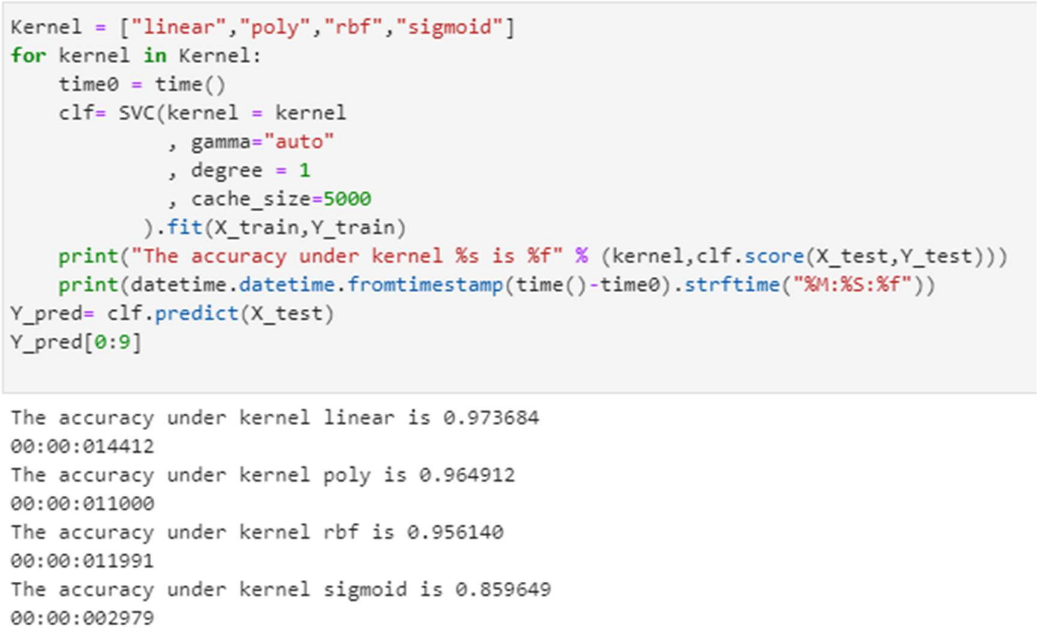
1. Identify the optimum number of K, principal components that achieve the highest classification accuracy.
2. Plot your classification accuracy, precision, and recall over a different number of Ks.
3. Explore different kernel tricks to capture non-linearities within your data. Plot the results and compare the accuracies for different kernels.
4. Compare your results against the logistic regression that you have done in homework 3.

Make sure to explain and elaborate your results.





As we can see, when K=2 the model reach the best result, maybe because it only affected by a few main factors, others are interference



```
Kernel = ["linear","poly","rbf","sigmoid"]
for kernel in Kernel:
    time0 = time()
    clf= SVC(kernel = kernel
             , gamma="auto"
             , degree = 1
             , cache_size=5000
            ).fit(X_train,Y_train)
    print("The accuracy under kernel %s is %f" % (kernel,clf.score(X_test,Y_test)))
    print(datetime.datetime.fromtimestamp(time()-time0).strftime("%M:%S:%f"))
Y_pred= clf.predict(X_test)
Y_pred[0:9]
```

```
The accuracy under kernel linear is 0.973684
00:00:014412
The accuracy under kernel poly is 0.964912
00:00:011000
The accuracy under kernel rbf is 0.956140
00:00:011991
The accuracy under kernel sigmoid is 0.859649
00:00:002979
```

Linear has the best accuracy, poly has the fastest train speed.

In last homework when k=9 it has the best performance:

Accuracy: 0.9736842105263158
Precision: 0.9558823529411765
Recall: 1.0

It is batter then this times.

Problem 2 (50pts):

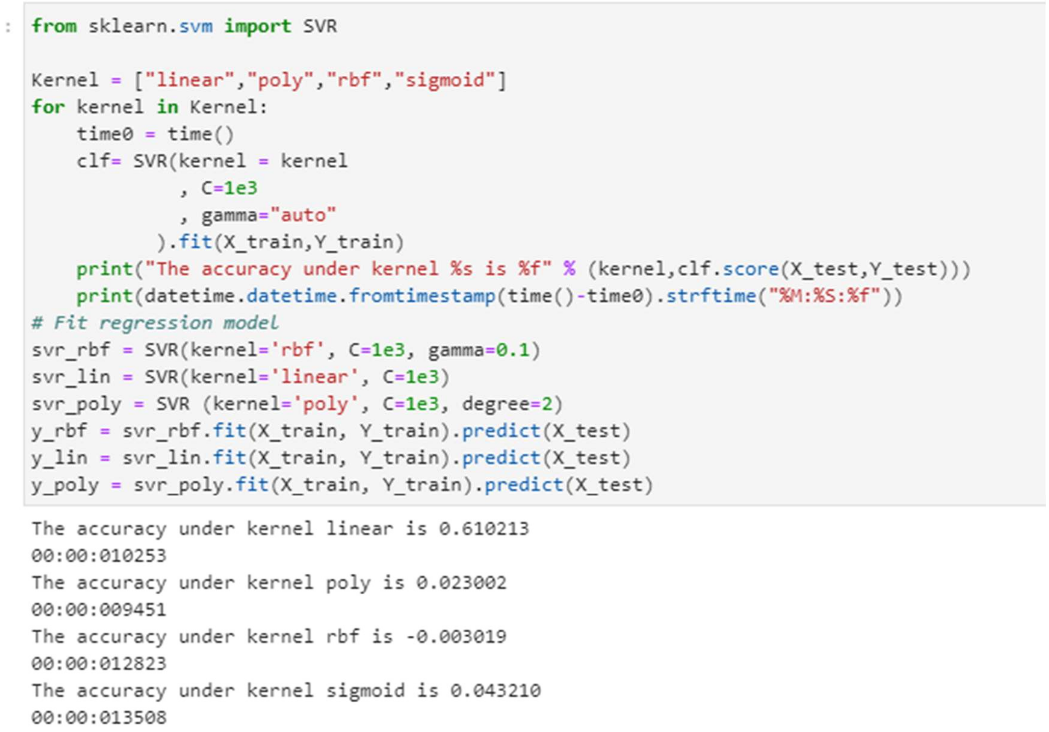
Develop a SVR regression model that predicts housing price based on the following input variables:

Area, bedrooms, bathrooms, stories, mainroad, guestroom, basement, hotwaterheating, airconditioning, parking, prefarea

1. Plot your regression model for SVR similar to the sample code provided on Canvas.
2. Compare your results against linear regression with regularization loss that you already did in homework1.
3. Use the PCA feature extraction for your training. Perform N number of independent training (N=1, …, K). Identify the optimum number of

K, principal components that achieve the highest regression accuracy.

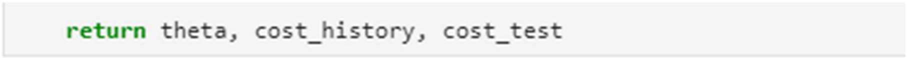
4. Explore different kernel tricks to capture non-linearities within your data. Plot the results and compare the accuracies for different kernels.



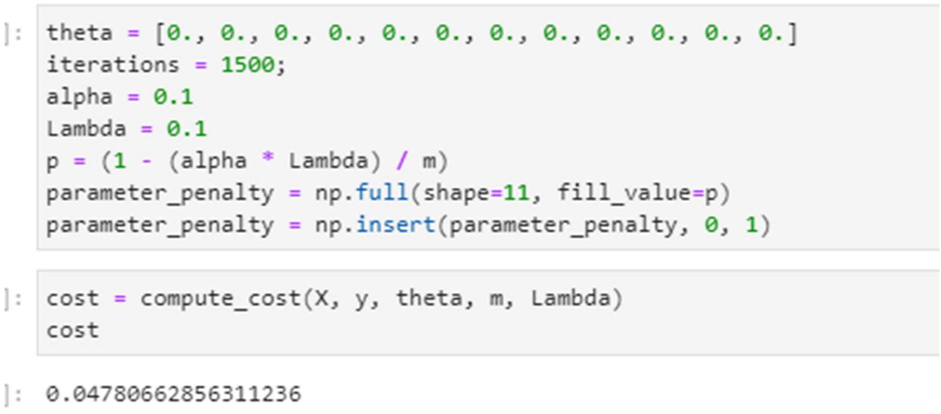
```
from sklearn.svm import SVR

Kernel = ["linear","poly","rbf","sigmoid"]
for kernel in Kernel:
    time0 = time()
    clf= SVR(kernel = kernel
             , C=1e3
             , gamma="auto"
           ).fit(X_train,Y_train)
    print("The accuracy under kernel %s is %f" % (kernel,clf.score(X_test,Y_test)))
    print(datetime.datetime.fromtimestamp(time()-time0).strftime("%M:%S:%f"))
# Fit regression model
svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)
svr_lin = SVR(kernel='linear', C=1e3)
svr_poly = SVR (kernel='poly', C=1e3, degree=2)
y_rbf = svr_rbf.fit(X_train, Y_train).predict(X_test)
y_lin = svr_lin.fit(X_train, Y_train).predict(X_test)
y_poly = svr_poly.fit(X_train, Y_train).predict(X_test)
```

```
The accuracy under kernel linear is 0.610213
00:00:010253
The accuracy under kernel poly is 0.023002
00:00:009451
The accuracy under kernel rbf is -0.003019
00:00:012823
The accuracy under kernel sigmoid is 0.043210
00:00:013508
```



```
    return theta, cost_history, cost_test
```

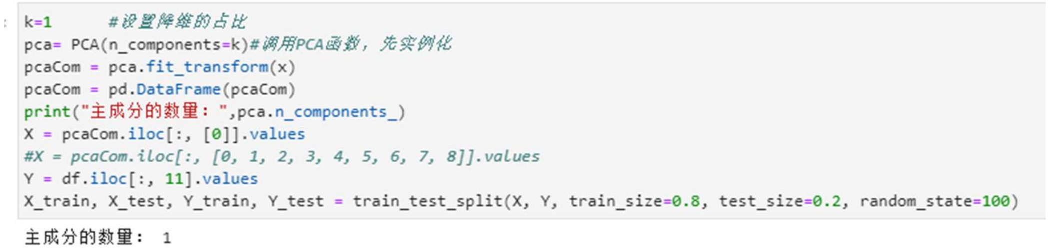


```
theta = [0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]
iterations = 1500;
alpha = 0.1
Lambda = 0.1
p = (1 - (alpha * Lambda) / m)
parameter_penalty = np.full(shape=11, fill_value=p)
parameter_penalty = np.insert(parameter_penalty, 0, 1)
```

```
cost = compute_cost(X, y, theta, m, Lambda)
cost
```

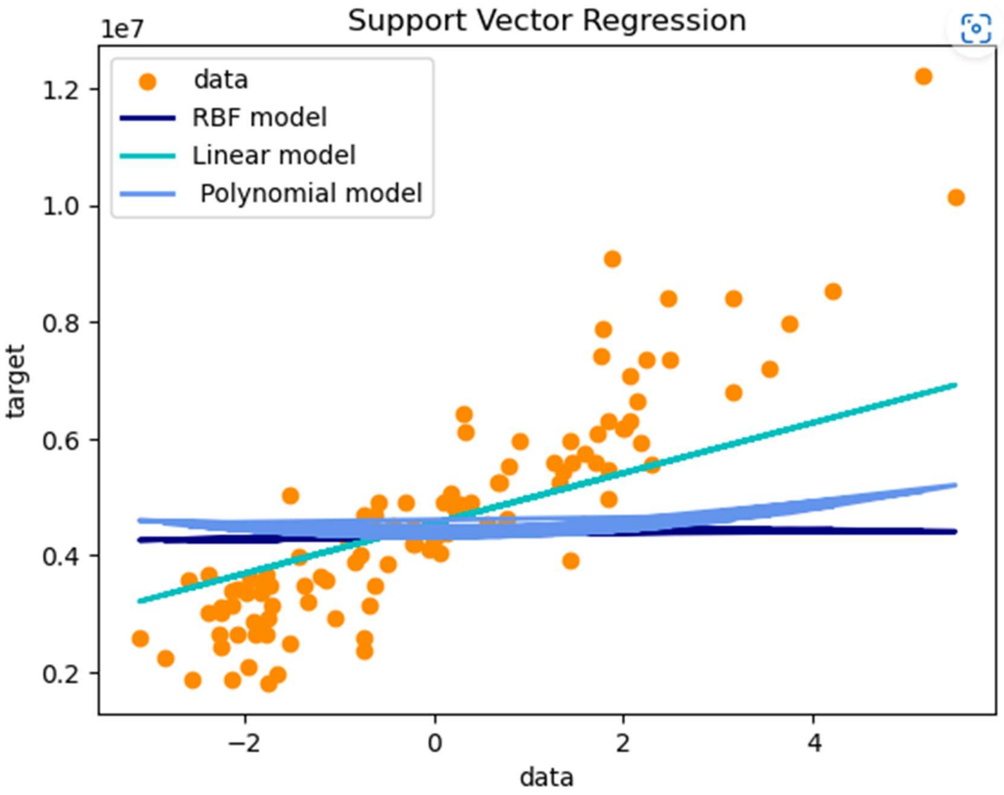
```
0.04780662856311236
```

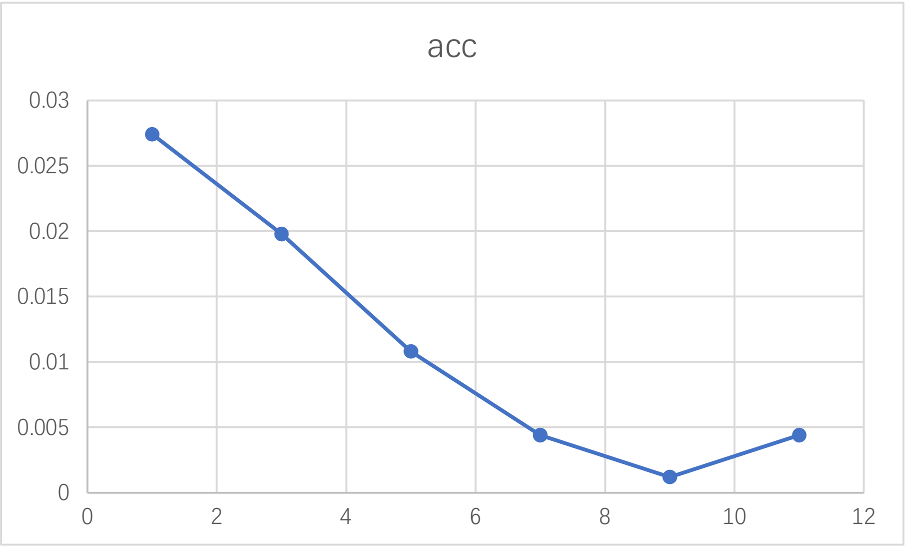
The lose in homework 1 is 0.04780662856311236, poly and rbf are smaller than it.



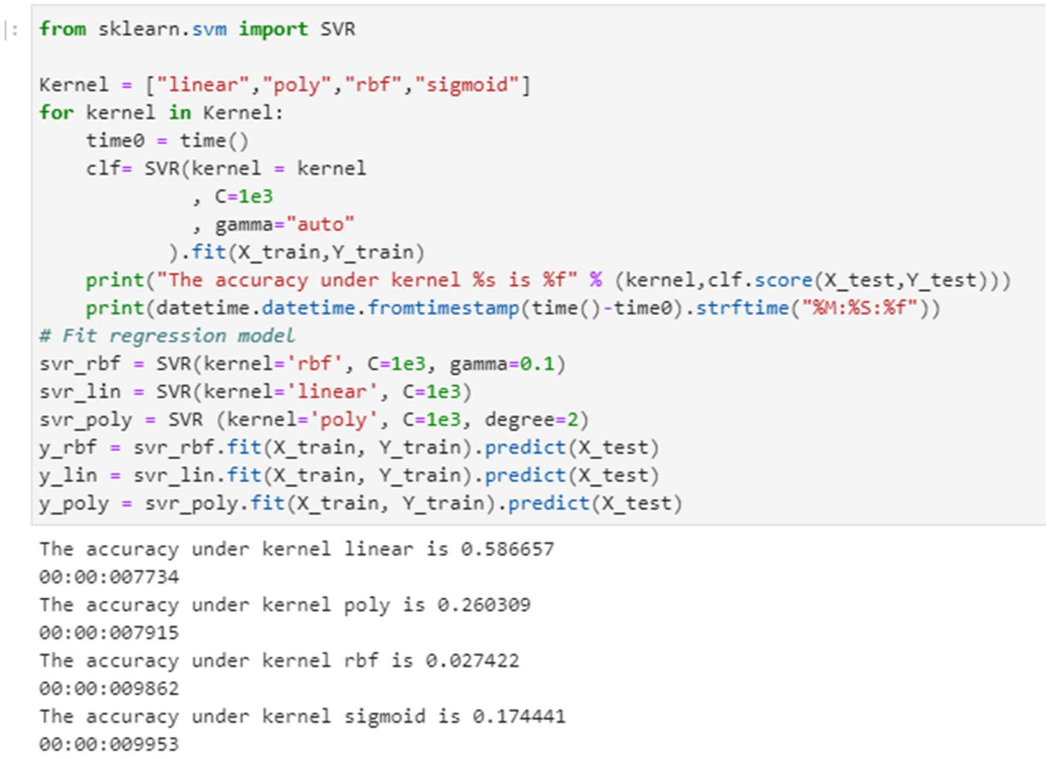
```
k=1       #设置降维的占比
pca= PCA(n_components=k)#调用PCA函数，先实例化
pcaCom = pca.fit_transform(x)
pcaCom = pd.DataFrame(pcaCom)
print("主成分的数量：",pca.n_components_)
X = pcaCom.iloc[:, [0]].values
#X = pcaCom.iloc[:, [0, 1, 2, 3, 4, 5, 6, 7, 8]].values
Y = df.iloc[:, 11].values
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size=0.8, test_size=0.2, random_state=100)
```

```
主成分的数量： 1
```





This is the plot for rbf kernel, we can see when k=9 it got the best result.



```python
from sklearn.svm import SVR

Kernel = ["linear","poly","rbf","sigmoid"]
for kernel in Kernel:
    time0 = time()
    clf= SVR(kernel = kernel
             , C=1e3
             , gamma="auto"
           ).fit(X_train,Y_train)
    print("The accuracy under kernel %s is %f" % (kernel,clf.score(X_test,Y_test)))
    print(datetime.datetime.fromtimestamp(time()-time0).strftime("%M:%S:%f"))
# Fit regression model
svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)
svr_lin = SVR(kernel='linear', C=1e3)
svr_poly = SVR (kernel='poly', C=1e3, degree=2)
y_rbf = svr_rbf.fit(X_train, Y_train).predict(X_test)
y_lin = svr_lin.fit(X_train, Y_train).predict(X_test)
y_poly = svr_poly.fit(X_train, Y_train).predict(X_test)
```

```
The accuracy under kernel linear is 0.586657
00:00:007734
The accuracy under kernel poly is 0.260309
00:00:007915
The accuracy under kernel rbf is 0.027422
00:00:009862
The accuracy under kernel sigmoid is 0.174441
00:00:009953
```

For different kernels, rbf is the best one.