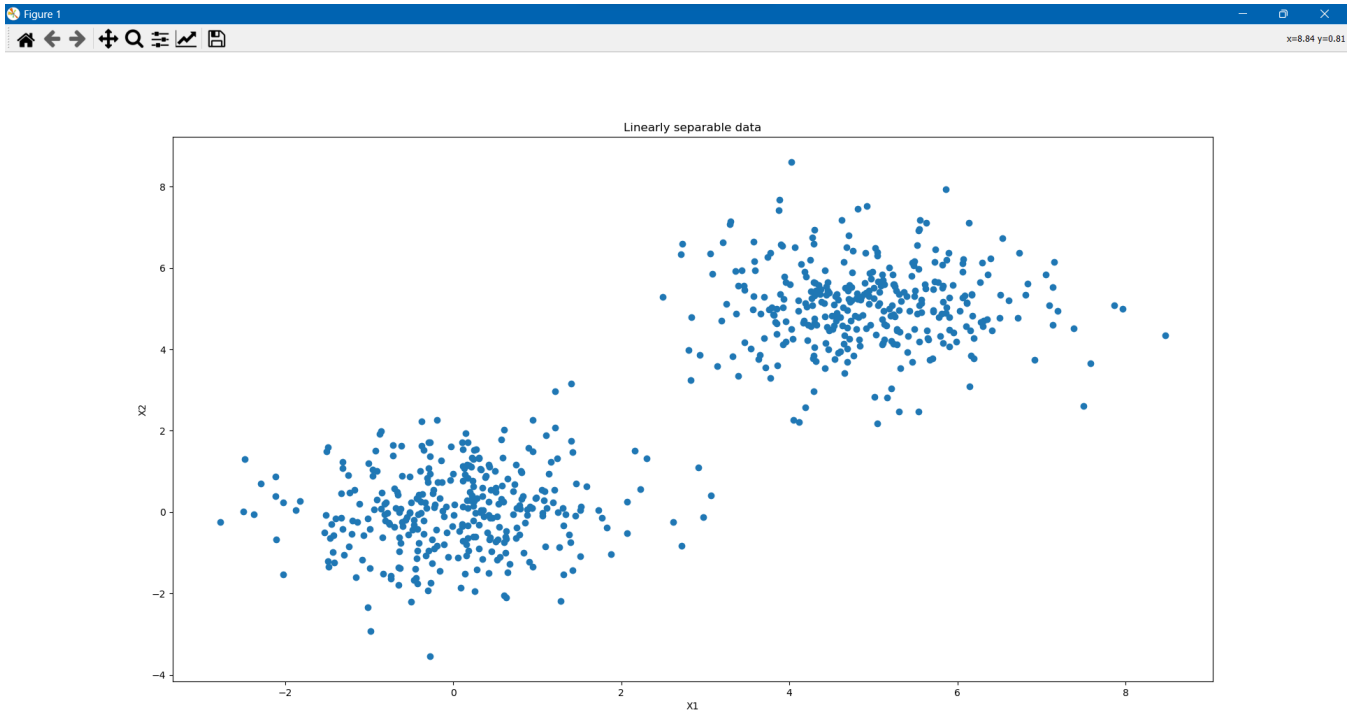
به نام خدا

Part I

در این قسمت باید از سه مدل **svm** استفاده کردیم که در هر یک به جواب های نسبتا خوبی رسیدیم.
Linear, Poly و **rbf** .
در ابتدا داده های رندوم تولید کردیم و به صورت رندوم به آن ها تگ باینری دادیم (به صورت رندوم به بعضی یک و بعضی دیگر صفر) و با نسبت **33** درصد آن ها را به داده های آموزشی و آزمایشی تقسیم کردیم.

```
inputs, targets = make_blobs(n_samples = 1000, centers = [(0,0), (5,5)], n_features = 2, cluster_std = 1)
X_train, X_test, y_train, y_test = train_test_split(inputs, targets, test_size=0.33, random_state=60)

plt.scatter(X_train[:,0], X_train[:,1])
plt.title('Linearly separable data')
plt.xlabel('X1')
plt.ylabel('X2')
plt.show()
```

سپس آن ها را نمایش می دهیم.

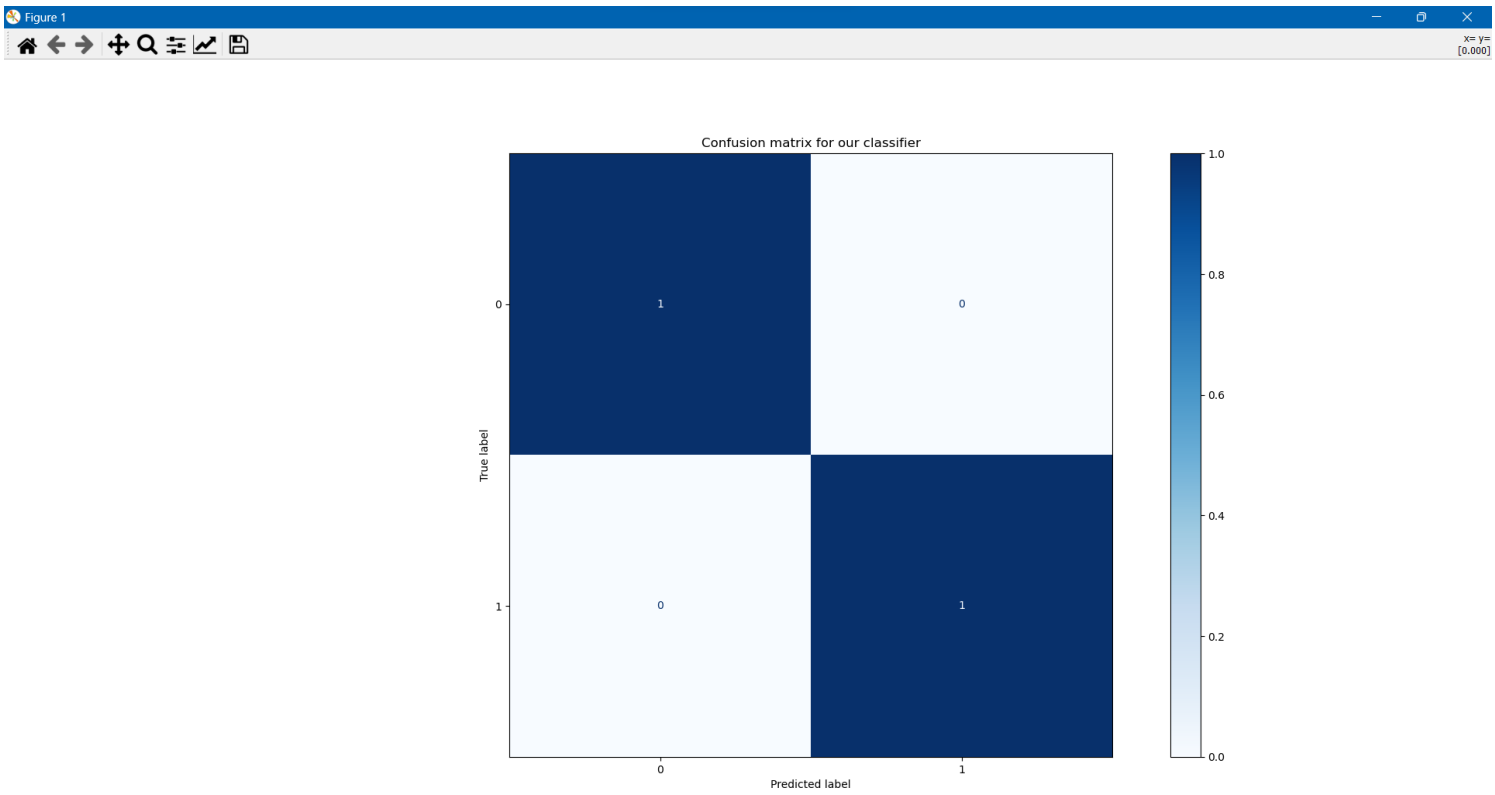


حال از سه مدل svm یکی را برای فیت رو داده ها انتخاب می کنیم و آن ها با داده ها آموزش میدهیم.

```
from sklearn import svm

clf = svm.SVC(kernel='linear')
clf = svm.SVC(kernel='rbf')
clf = svm.SVC(kernel='poly')


clf = clf.fit(X_train, y_train)
```

سپس مدل را روی داده های آزمایشی پیش بینی می کنیم و ماتریس نتیجه را رسم می کنیم.

```
predictions = clf.predict(X_test)
from sklearn.metrics import plot_confusion_matrix

predictions = clf.predict(X_test)


matrix = plot_confusion_matrix(clf, X_test, y_test,
                               cmap=plt.cm.Blues,
                               normalize='true')
plt.title('Confusion matrix for our classifier')

plt.show()
```



حال بردارهای پشتیبان و نحوه ی جداشدگی خطی داده ها را نمایش می دهیم.

```
support_vectors = clf.support_vectors_
plt.scatter(X_train[:,0], X_train[:,1])
plt.scatter(support_vectors[:,0], support_vectors[:,1], color='red')
plt.title('Linearly separable data with support vectors')
plt.xlabel('X1')
plt.ylabel('X2')
plt.show()
```

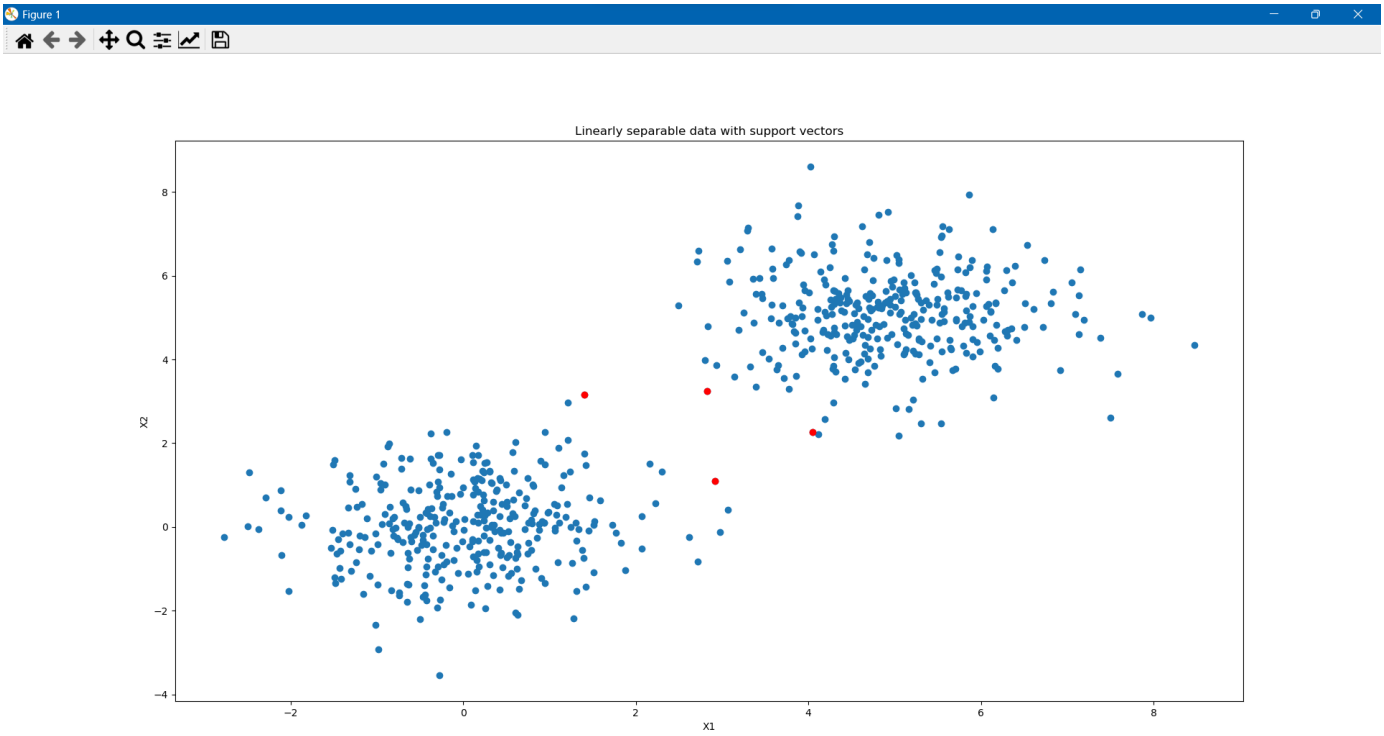
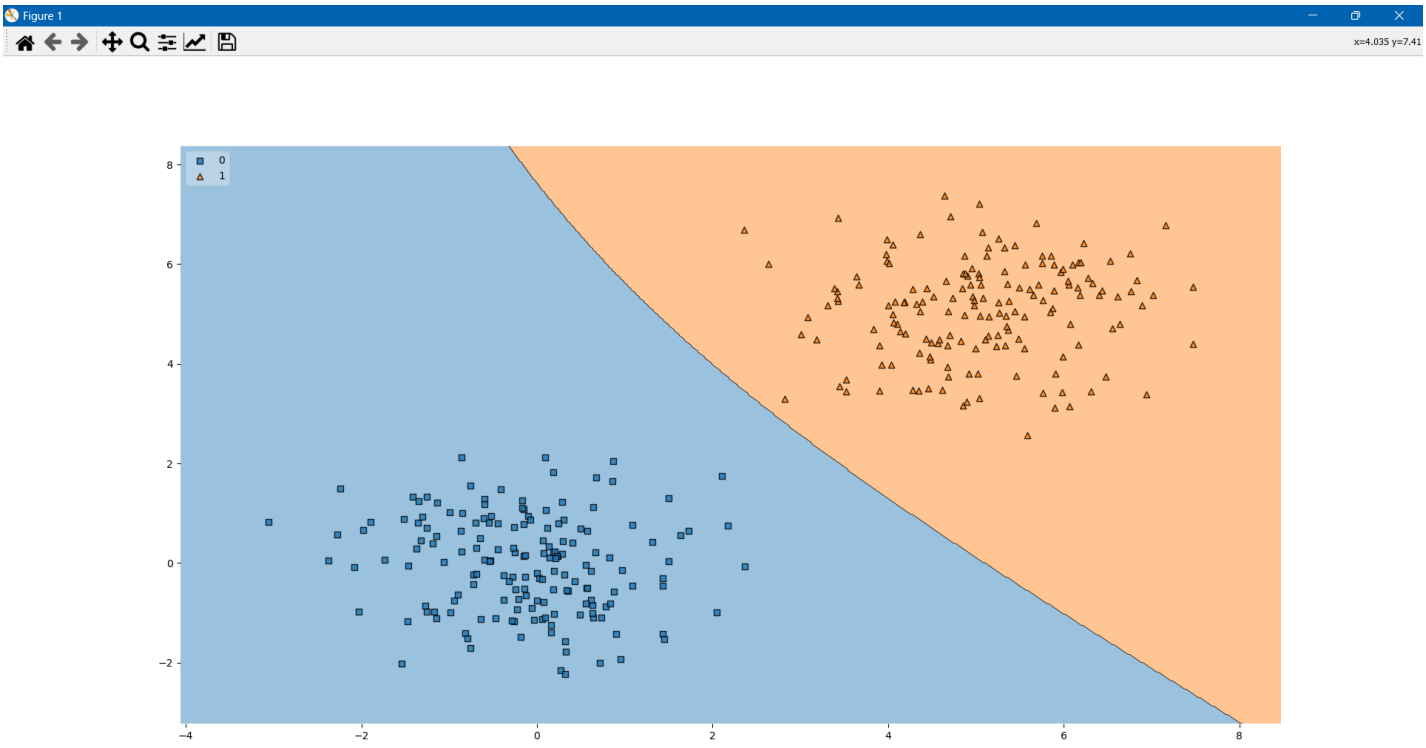


Figure 1
Linearly separable data with support vectors
X2
X1
8
6
4
2
0
−2
−4
−2
0
2
4
6
8

```
from mlxtend.plotting import plot_decision_regions

plot_decision_regions(X_test, y_test, clf=clf, legend=2)
plt.show()
```



Part II

برای این قسمت باید از svm برای categorical classification بهره بگیریم.
دیتاست مربوط به قسمت پنجم شبکه عصبی را استخراج و با عملیات زیر به داده های آموزشی و آزمایشی مطلوب تبدیل می کنیم.

```
local_zip = '/content/USPS_images.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/content/trainntest')

train_dir = '/content/trainntest/train'
validation_dir = '/content/trainntest/test'
y_train = []
y_test = []
for path in os.listdir(train_dir):
    if os.path.isfile(os.path.join(train_dir, path)):
        y_train.append(int(path[0]))

for path in os.listdir(validation_dir):
  if os.path.isfile(os.path.join(validation_dir, path)):
      y_test.append(int(path[0]))

x_train = []
for path in os.listdir(train_dir):
  if os.path.isfile(os.path.join(train_dir,path)):
    x_train.append(cv2.cvtColor(cv2.imread(f"{train_dir}/{path}"), cv2.COLOR_RGB2GRAY))

x_test = []
for path in os.listdir(validation_dir):
  if os.path.isfile(os.path.join(validation_dir,path)):
    x_test.append(cv2.cvtColor(cv2.imread(f"{validation_dir}/{path}"), cv2.COLOR_RGB2GRAY))

x_train = np.array(x_train)
x_test = np.array(x_test)
y_train = np.array(y_train)
y_test = np.array(y_test)


x_train_final = x_train.reshape(-1 ,16*16) / 255
x_test_final = x_test.reshape(-1 ,16*16) / 255
```

تابع pick با گرفتن یک عدد کسری به عنوان پارامتر تقسیم، داده ها را به آموزش و آزمایش تقسیم می کند.

```
train, trlab, test, tslab  = x_train_final, y_train, x_test_final, y_test
```

```
train.shape
```

```
(7291, 256)
```

```
def pick(train, trlab, test, tslab, percentage):
  train = train[:int(len(train)*percentage)]
  test = test[:int(len(test)*percentage)]
  trlab = trlab[:int(len(trlab)*percentage)]
  tslab = tslab[:int(len(tslab)*percentage)]
  return train, trlab, test, tslab
```

```
train, trlab, test, tslab = pick(train, trlab, test, tslab, 0.05)
```

حال به LinearSVC ، مدل svm را روی داده های آموزشی ، آموزش می دهیم و ضرایب ابرصفحه را به دست می آوریم.
و در آخر با استفاده از تابع accuracy_score ، دقت مدل را به دست می آوریم.

```
    svm = LinearSVC(dual=False, verbose=1)
    svm.fit(train, trlab)
```

```
[LibLinear]
```

```
LinearSVC(dual=False, verbose=1)
```

```
    svm.coef_
    svm.intercept_
```

```
array([-0.71078573, -0.23419634, -0.16636508, -0.76613492, -0.01311585,
       -0.62888266, -0.28157165, -0.01484649, -0.55359038, -0.76468539])
```

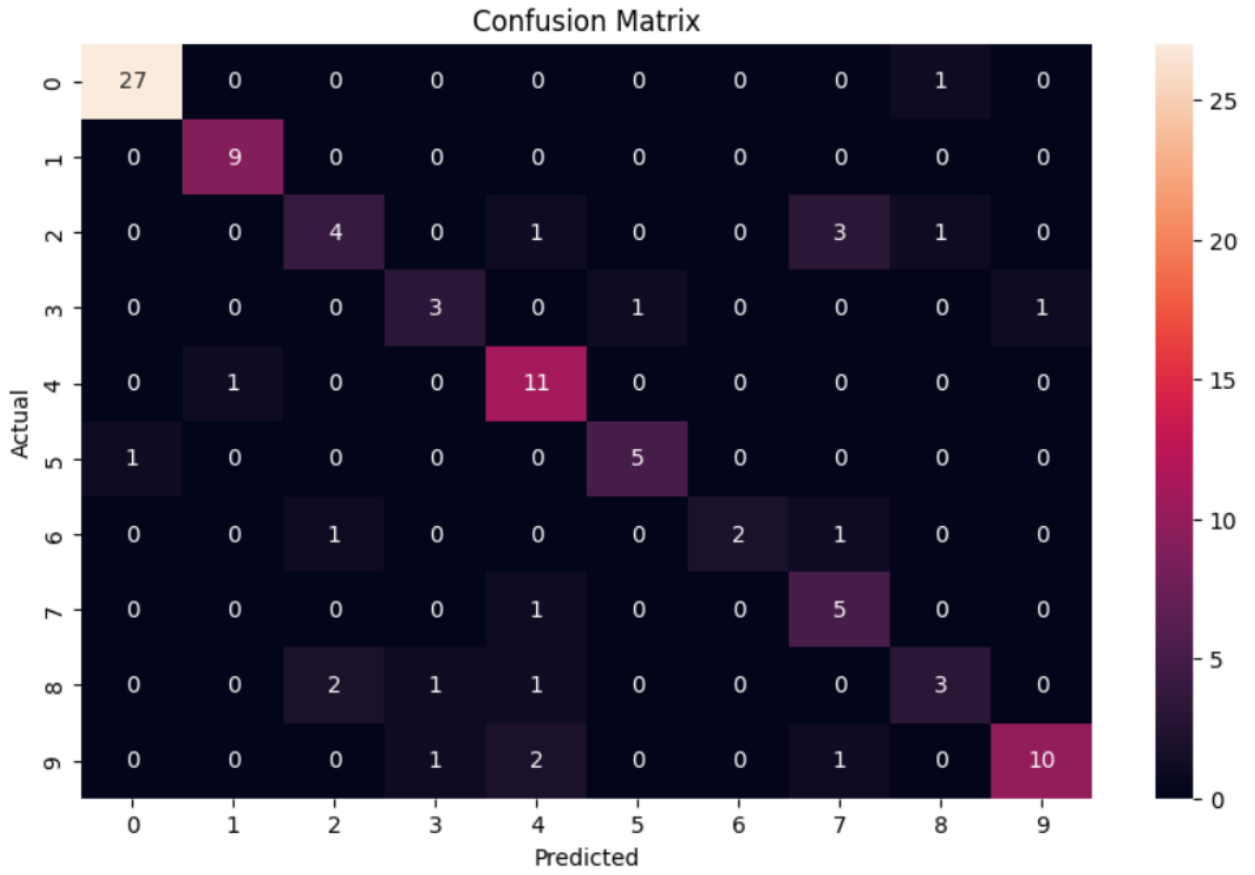
```
    pred = svm.predict(test)
```

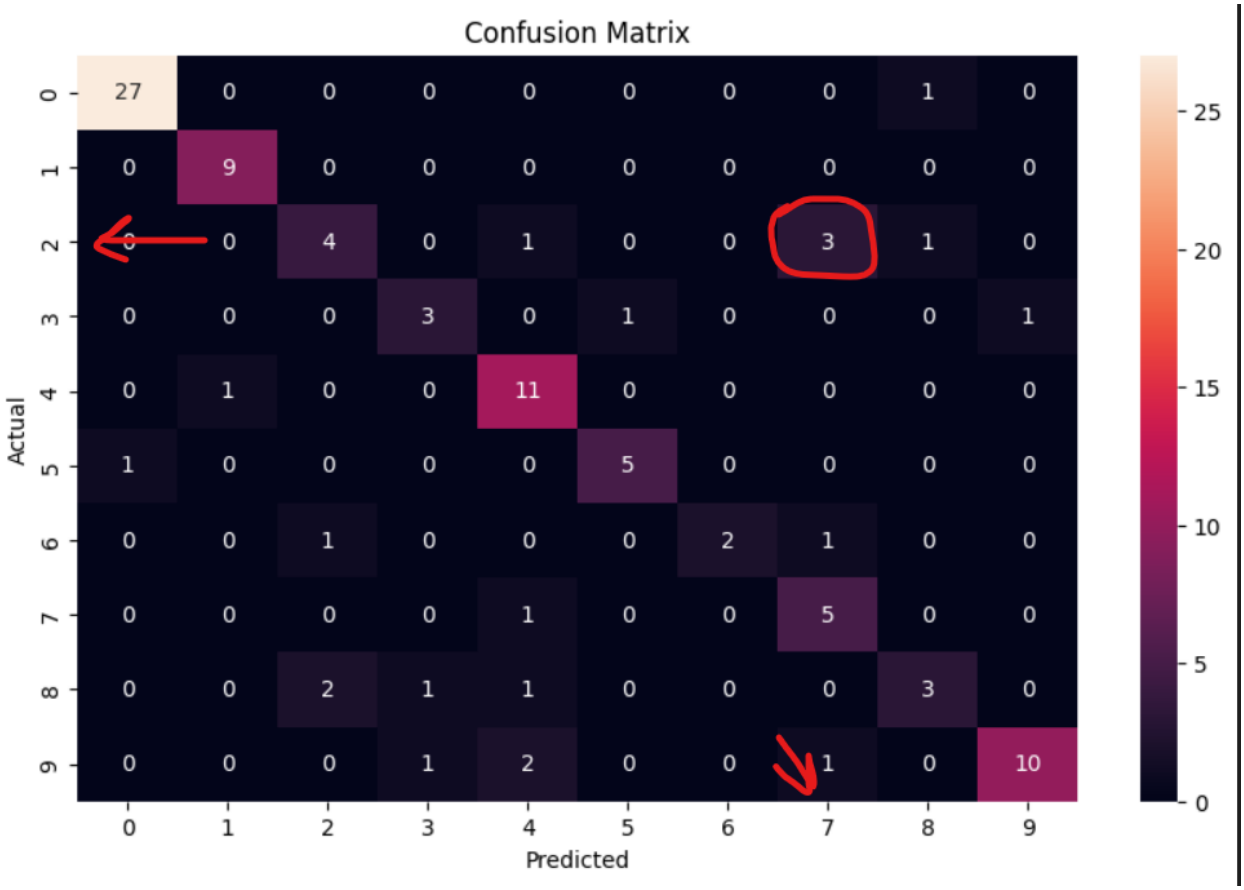
```
    accuracy_score(tslab, pred) # Accuracy
```

```
0.79
```

و در نهایت کانفیوژن ماتریس را رسم می کنیم.
که برای مثال ضعف مدل svm در تمایز بین دو و هفت را می توان مشاهده کرد.

```
cm = confusion_matrix(tslab, pred)
matplot.subplots(figsize=(10, 6))
sb.heatmap(cm, annot = True, fmt = 'g')
matplot.xlabel("Predicted")
matplot.ylabel("Actual")
matplot.title("Confusion Matrix")
matplot.show()
```





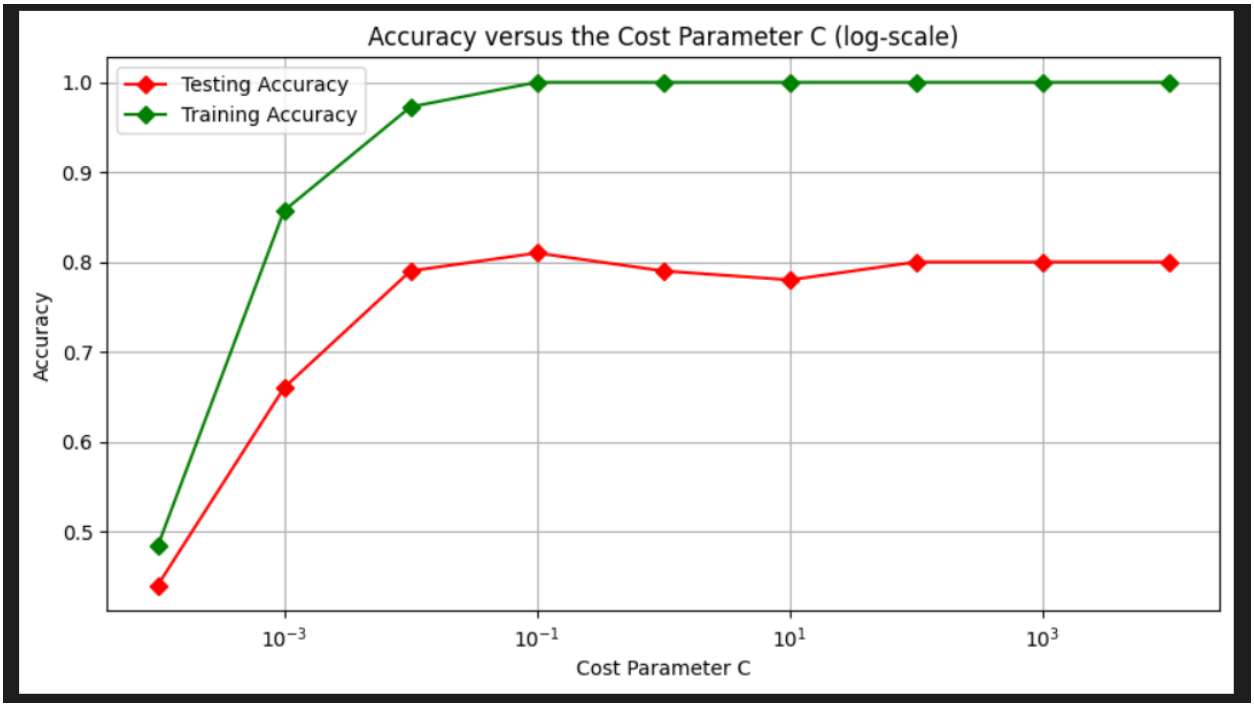
حال ضرایب ابر صفحه را در حالتی که l=2 می باشد،(از اختلاف توان دو می گیرد.) برای ضرایب c مختلف به دست می آوریم و ذخیره می کنیم و نمودار آن را بر روی داده های آموزشی و آزمایشی رسم میکنیم.

```
    acc = []
    acc_tr = []
    coefficient = []
    for c in [0.0001,0.001,0.01,0.1,1,10,100,1000,10000]:
        svm = LinearSVC(dual=False, C=c)
        svm.fit(train, trlab)
        coef = svm.coef_

        p_tr = svm.predict(train)
        a_tr = accuracy_score(trlab, p_tr)

        pred = svm.predict(test)
        a = accuracy_score(tslab, pred)

        coefficient.append(coef)
        acc_tr.append(a_tr)
        acc.append(a)
        print(f"done for {c}")
```

[140]

```
done for 0.0001
done for 0.001
done for 0.01
done for 0.1
done for 1
done for 10
done for 100
done for 1000
done for 10000
```

```
c = [0.0001,0.001,0.01,0.1,1,10,100,1000,10000]

matplot.subplots(figsize=(10, 5))
matplot.semilogx(c, acc,'-gD' ,color='red' , label="Testing Accuracy")
matplot.semilogx(c, acc_tr,'-gD' , label="Training Accuracy")
#matplot.xticks(L,L)
matplot.grid(True)
matplot.xlabel("Cost Parameter C")
matplot.ylabel("Accuracy")
matplot.legend()
matplot.title('Accuracy versus the Cost Parameter C (log-scale)')
matplot.show()
```



که بهترین نتیجه برای حالتی به دست آمد که c برابر یک باشد.

در اینجا ضرایب ابر صفحه برای هر کلاس (0 تا 9) برای c=1 ترسیم می کنیم.

c = 1 : the Best

```
svm_coef = coefficient[4]
svm_coef.shape
```

42]

```
(10, 256)
```

```
matplot.subplots(2,5, figsize=(24,10))
for i in range(10):
    l1 = matplot.subplot(2, 5, i + 1)
    l1.imshow(svm_coef[i].reshape(16, 16), cmap=matplot.cm.RdBu)
    l1.set_xticks(())
    l1.set_yticks(())
    l1.set_xlabel('Class %i' % i)
matplot.suptitle('Class Coefficients')
matplot.show()
```

67]

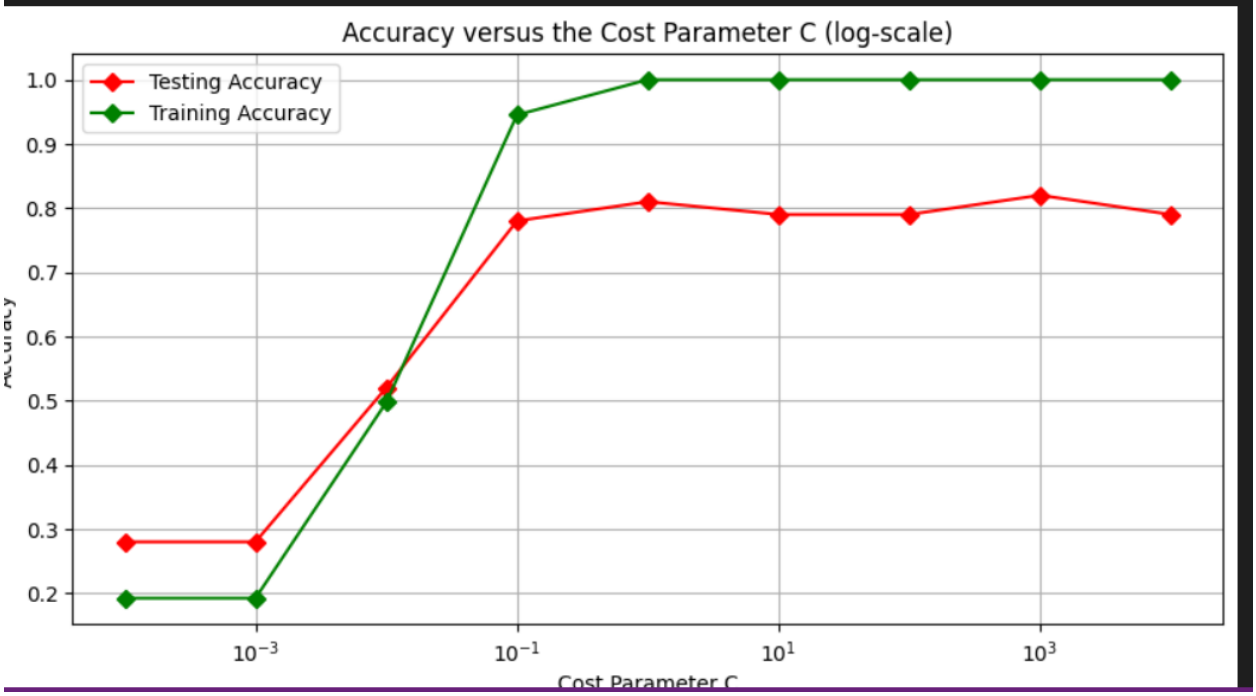

```
matplot.subplots(2,5, figsize=(24,10))
for i in range(10):
    l1 = matplot.subplot(2, 5, i + 1)
    l1.imshow(svm_coef[i].reshape(16, 16), cmap=matplot.cm.RdBu)
    l1.set_xticks(())
    l1.set_yticks(())
    l1.set_xlabel('Class %i' % i)
matplot.suptitle('Class Coefficients')
matplot.show()
```

حال ضرایب ابر صفحه را در حالتی که l=1 می باشد،(از اختلاف قدر مطلق می گیرد.) برای ضرایب c مختلف به دست می آوریم و ذخیره می کنیم و نمودار آن را بر روی داده های آموزشی و آزمایشی رسم میکنیم.

```
acc = []
acc_tr = []
coefficient = []
for c in [0.0001,0.001,0.01,0.1,1,10,100,1000,10000]:
    svm = LinearSVC(dual=False, C=c, penalty='l1')
    svm.fit(train, trlab)
    coef = svm.coef_

    p_tr = svm.predict(train)
    a_tr = accuracy_score(trlab, p_tr)

    pred = svm.predict(test)
    a = accuracy_score(tslab, pred)

    coefficient.append(coef)
    acc_tr.append(a_tr)
    acc.append(a)
```

و برای c های مختلف ضرایب را به دست می آوریم و ذخیره می کنیم و درنهایت نمودار آن هارا روی داده های آموزشی و آزمایشی رسم می کنیم که باز هم بهترین نتیجه وقتی c برابر یک باشد ، میدهد.

```
c = [0.0001,0.001,0.01,0.1,1,10,100,1000,10000]

matplot.subplots(figsize=(10, 5))
matplot.semilogx(c, acc,'-gD' ,color='red' , label="Testing Accuracy")
matplot.semilogx(c, acc_tr,'-gD' , label="Training Accuracy")
#matplot.xticks(L,L)
matplot.grid(True)
matplot.xlabel("Cost Parameter C")
matplot.ylabel("Accuracy")
matplot.legend()
matplot.title('Accuracy versus the Cost Parameter C (log-scale)')
matplot.show()
```

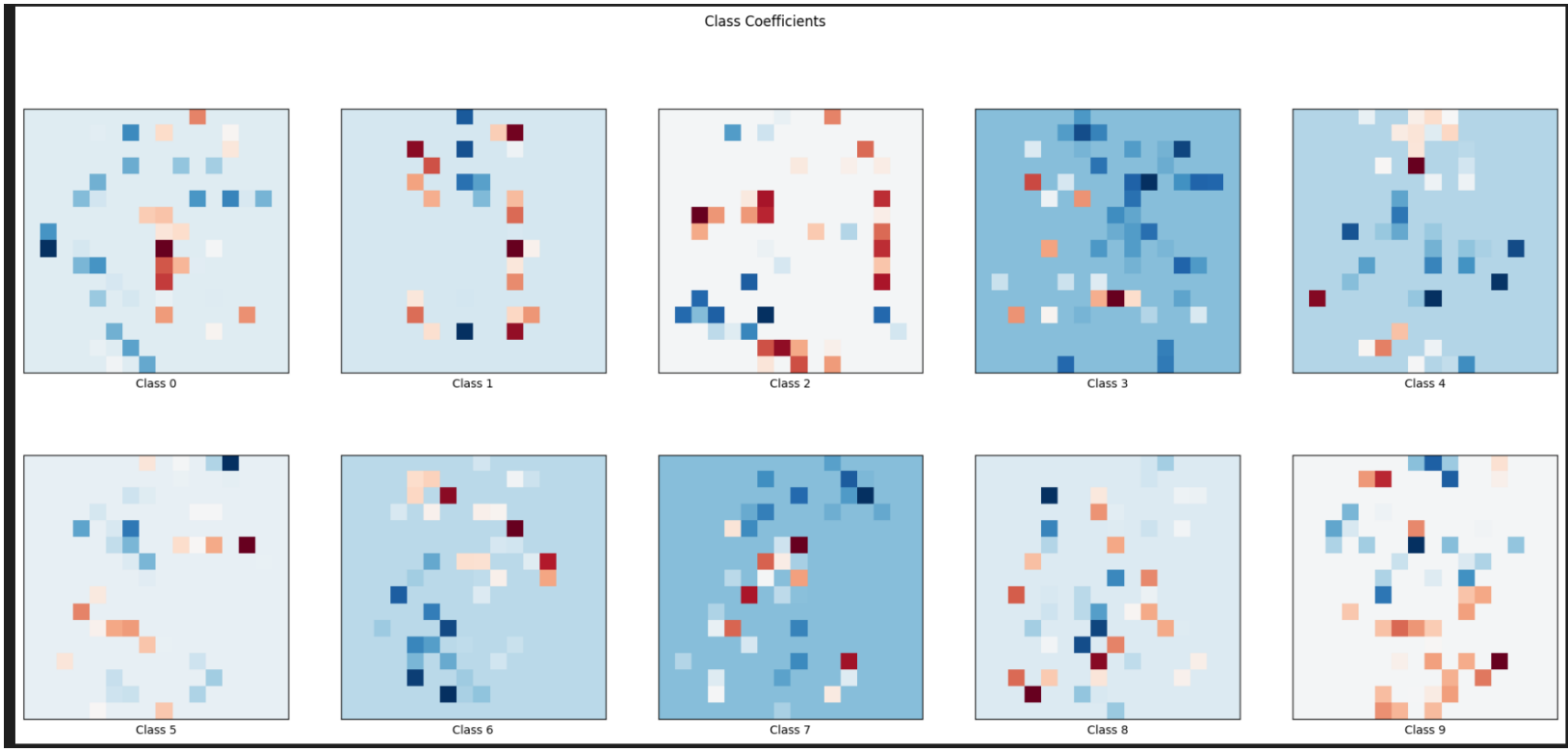


c = 1 : the Best

```
svm_coef = coefficient[4]
svm_coef.shape
```

```
(10, 256)
```

```
matplot.subplots(2,5, figsize=(24,10))
for i in range(10):
    l1 = matplot.subplot(2, 5, i + 1)
    l1.imshow(svm_coef[i].reshape(16, 16), cmap=matplot.cm.RdBu)
    l1.set_xticks(())
    l1.set_yticks(())
    l1.set_xlabel('Class %i' % i)
matplot.suptitle('Class Coefficients')
matplot.show()
```



حال kernel rbf را برای مدل svm انتخاب میکنیم و این بار داده های رندوم از داده های آموزشی و آزمایشی را برای آموزش و آزمایش انتخاب می کنیم و آن ها را نمایش می دهیم و چون آموزش بسیار زمان بر خواهد بود درصد بالایی از داده ها را انتخاب نمی کنیم.

SVM RBF Kernel

generate a random sample of the data and check how the distribution is compared to the origin

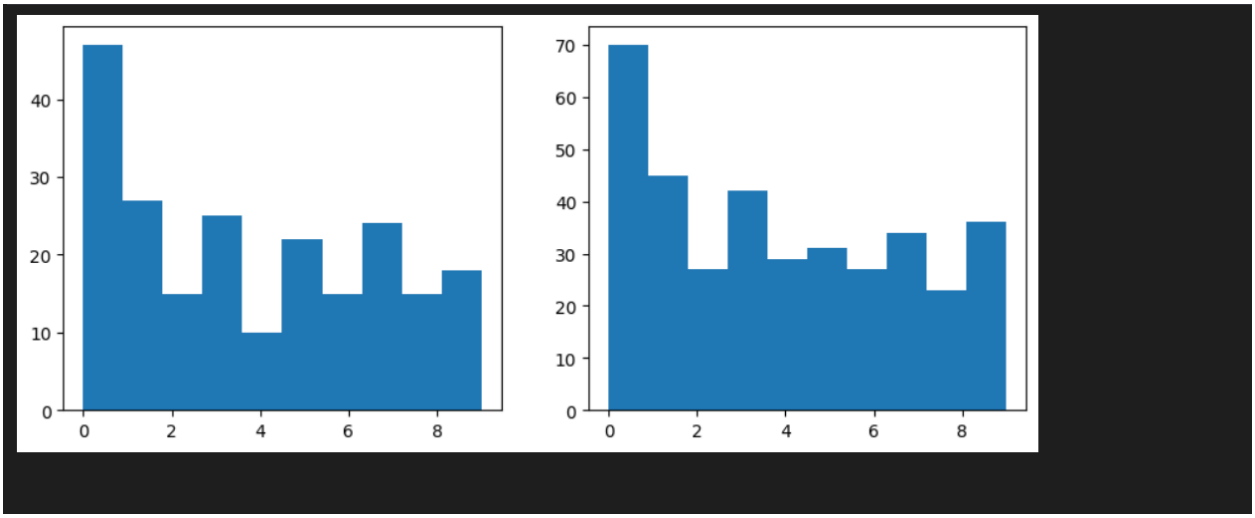
```
[48]
seq = np.random.randint(0,len(train),int(0.6*(len(train))))
train_samp = train[seq]
trlab_samp = trlab[seq]

train_samp.shape
trlab_samp.shape
```

```
(218,)
```

```
[49]
seq = np.random.randint(0,len(test),int(0.6*(len(test))))
test_samp = test[seq]
tslab_samp = tslab[seq]

test_samp.shape
tslab_samp.shape
```

```
(60,)
```

```
fig, ax = matplot.subplots(1,2, figsize=(10,4))
ax[0].hist(trlab_samp)
ax[1].hist(trlab)
fig.show
matplot.show()
```



حال svm را با c های مختلف مدل می کنیم.

Running SVC for multiple cost factor(s) C and Gamma

```
coefficient = []
n_supp = []
sup_vec = []
i = 0
df = pd.DataFrame(columns = ['c','gamma','train_acc','test_acc'])
for c in [0.01, 0.1, 1, 10, 100]:
    for g in [0.01, 0.1, 1, 10, 100]:
        svm = SVC(kernel='rbf', C=c, gamma=g)
        model = svm.fit(train_samp, trlab_samp)
        globals()['model%s' % i] = model
        d_coef = svm.dual_coef_
        support = svm.n_support_
        sv = svm.support_

        p_tr = svm.predict(train_samp)
        a_tr = accuracy_score(trlab_samp, p_tr)

        pred = svm.predict(test_samp)
        a = accuracy_score(tslab_samp, pred)

        coefficient.append(d_coef)
        n_supp.append(support)
        sup_vec.append(sv)
        df.loc[i] = [c,g,a_tr,a]
        i=i+1
```

```
df
```

	c	gamma	train_acc	test_acc
0	0.01	0.01	0.215596	0.266667
1	0.01	0.10	0.215596	0.266667
2	0.01	1.00	0.215596	0.266667
3	0.01	10.00	0.215596	0.266667
4	0.01	100.00	0.215596	0.266667
5	0.10	0.01	0.339450	0.333333
6	0.10	0.10	0.330275	0.316667
7	0.10	1.00	0.215596	0.266667
8	0.10	10.00	0.215596	0.266667
9	0.10	100.00	0.215596	0.266667
10	1.00	0.01	0.981651	0.683333
11	1.00	0.10	1.000000	0.516667
12	1.00	1.00	1.000000	0.283333
13	1.00	10.00	1.000000	0.266667
14	1.00	100.00	1.000000	0.266667
15	10.00	0.01	1.000000	0.783333
16	10.00	0.10	1.000000	0.550000
17	10.00	1.00	1.000000	0.300000
18	10.00	10.00	1.000000	0.266667
19	10.00	100.00	1.000000	0.266667
20	100.00	0.01	1.000000	0.783333
21	100.00	0.10	1.000000	0.550000
22	100.00	1.00	1.000000	0.300000

و همین فرایند را برای poly kernel انجام می دهیم و نمودارهای مربوطه را رسم میکنیم.

```
coefficient = []
n_supp = []
sup_vec = []
i = 0
df = pd.DataFrame(columns = ['c','degree','train_acc','test_acc'])
for c in [0.01, 0.1, 1, 10, 100]:
    for d in [2,3,4,5,6]:
        svm = SVC(kernel='poly', C=c, degree=d)
        model = svm.fit(train_samp, trlab_samp)
        globals()['model%s' % i] = model
        d_coef = svm.dual_coef_
        support = svm.n_support_
        sv = svm.support_

        p_tr = svm.predict(train_samp)
        a_tr = accuracy_score(trlab_samp, p_tr)

        pred = svm.predict(test_samp)
        a = accuracy_score(tslab_samp, pred)

        coefficient.append(d_coef)
        n_supp.append(support)
        sup_vec.append(sv)
        df.loc[i] = [c,d,a_tr,a]
        i=i+1
```

```
df
```

	c	degree	train_acc	test_acc
0	0.01	2.0	0.192661	0.250000
1	0.01	3.0	0.211009	0.250000
2	0.01	4.0	0.307339	0.283333

	c	degree	train_acc	test_acc
0	0.01	2.0	0.192661	0.250000
1	0.01	3.0	0.211009	0.250000
2	0.01	4.0	0.307339	0.283333
3	0.01	5.0	0.344037	0.250000
4	0.01	6.0	0.376147	0.266667
5	0.10	2.0	0.752294	0.533333
6	0.10	3.0	0.770642	0.516667
7	0.10	4.0	0.766055	0.516667
8	0.10	5.0	0.729358	0.400000
9	0.10	6.0	0.729358	0.400000
10	1.00	2.0	0.990826	0.816667
11	1.00	3.0	0.967890	0.766667
12	1.00	4.0	0.954128	0.683333
13	1.00	5.0	0.926606	0.583333
14	1.00	6.0	0.926606	0.533333
15	10.00	2.0	1.000000	0.833333
16	10.00	3.0	1.000000	0.816667
17	10.00	4.0	1.000000	0.733333
18	10.00	5.0	1.000000	0.733333
19	10.00	6.0	0.995413	0.683333
20	100.00	2.0	1.000000	0.833333
21	100.00	3.0	1.000000	0.816667
22	100.00	4.0	1.000000	0.733333
23	100.00	5.0	1.000000	0.733333
24	100.00	6.0	1.000000	0.700000

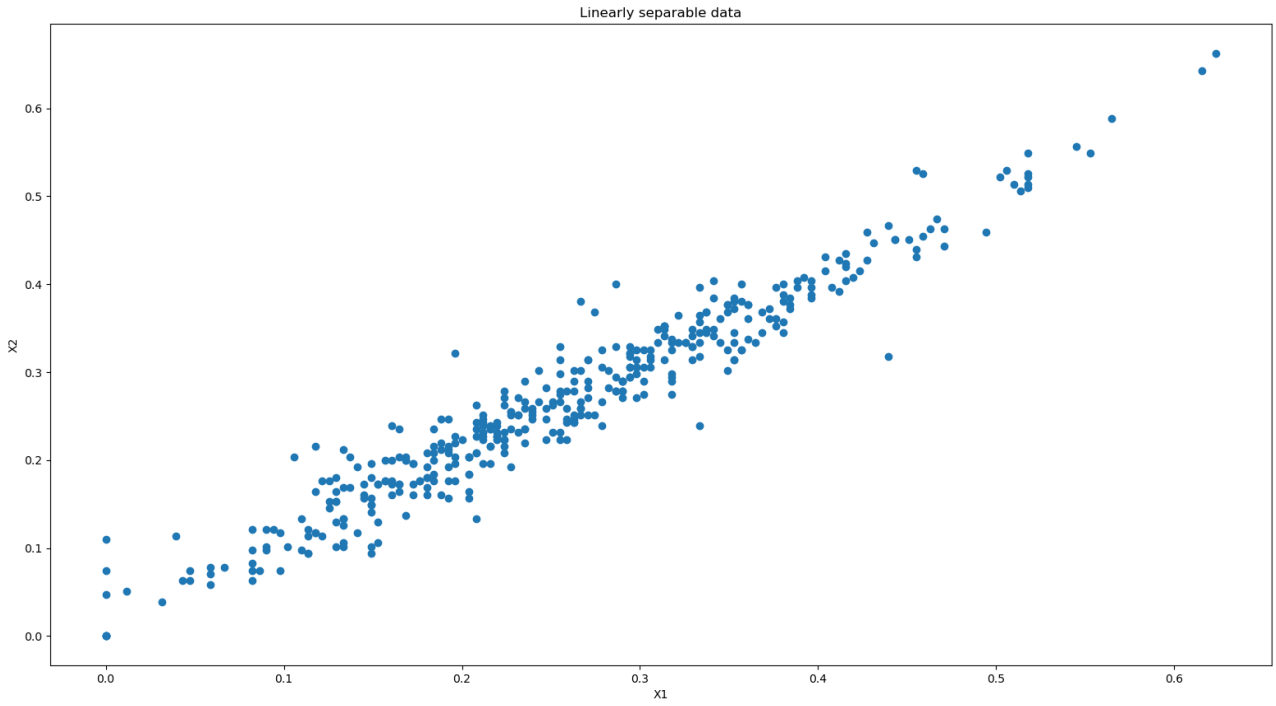
Part 3

در این قسمت به دلیل عدم توانایی و ضعف svm در تشخیص و تمایز بین بعضی اعداد و حروف(همانطور که در بخش قبلی نشان داده شد که در تمایز بین دو و هفت ضعیف عمل می کرد) ، این اعداد و حروف را به صورت دوتایی به مدل svm می دهیم تا روی آن ها آموزش ببیند و بتواند عملکرد بهتری داشته باشد.
ابتدا تصاویر را به صورت آرایه و در نهایت داده های مطلوب جهت آموزش و تست مدل آماده می کنیم.
داده ها را با برچسب گذاری صفر و یک به دو نوع داده ی مجزا تقسیم می کنیم تا با svm آن ها را جدا کنیم.

```
train_dir1 = 'E:\\6\AI\\PJ4\\P3\\2'
train_dir2 = 'E:\\6\\AI\\PJ4\\P3\\7'

x_trainNtest1 = []
x_trainNtest2 = []
for path in os.listdir(train_dir1):
  if os.path.isfile(os.path.join(train_dir1,path)):
    x_trainNtest1.append(cv2.cvtColor(cv2.imread(f"{train_dir1}/{path}"), cv2.COLOR_RGB2GRAY))

for path in os.listdir(train_dir2):
  if os.path.isfile(os.path.join(train_dir2,path)):
    x_trainNtest2.append(cv2.cvtColor(cv2.imread(f"{train_dir2}/{path}"), cv2.COLOR_RGB2GRAY))

y_trainNtest1 = [0]*len(x_trainNtest1)
y_trainNtest2 = [1]*len(x_trainNtest2)

x_trainNtest1 = np.array(x_trainNtest1)
x_trainNtest2 = np.array(x_trainNtest2)
y_trainNtest1 = np.array(y_trainNtest1)
y_trainNtest2 = np.array(y_trainNtest2)

X = np.concatenate((x_trainNtest1,x_trainNtest2))
Y = np.concatenate((y_trainNtest1,y_trainNtest2))
print(X.shape)
X = X.reshape(-1 ,16*16) / 255

X_train, X_test, y_train, y_test = train_test_split( X, Y, test_size=0.33, random_state=42)
```

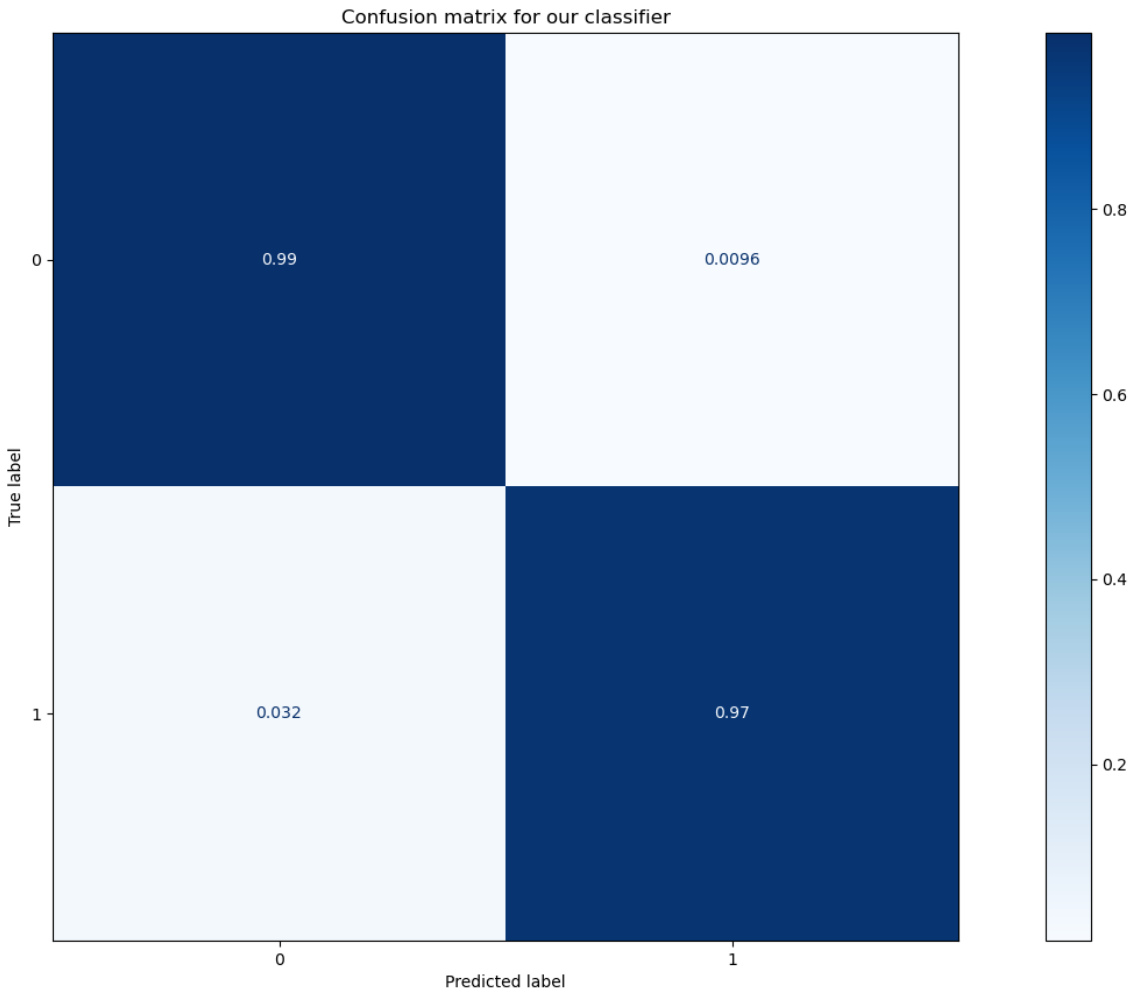
داده ها را نمایش می دهیم.

```
plt.scatter(X_train[:,0], X_train[:,1])
plt.title('Linearly separable data')
plt.xlabel('X1')
plt.ylabel('X2')
plt.show()
```



یک مدل kernel برای svm انتخاب می کنیم(در اینجا rbf) و با آن مدل را روی داده ها آموزش می دهیم و در نهایت آن را روی داده های آزمایشی مدل می کنیم و ماتریس حاصل از نتایج بدست آمده و نتایج واقعی مربوط به svm را نمایش می دهیم.

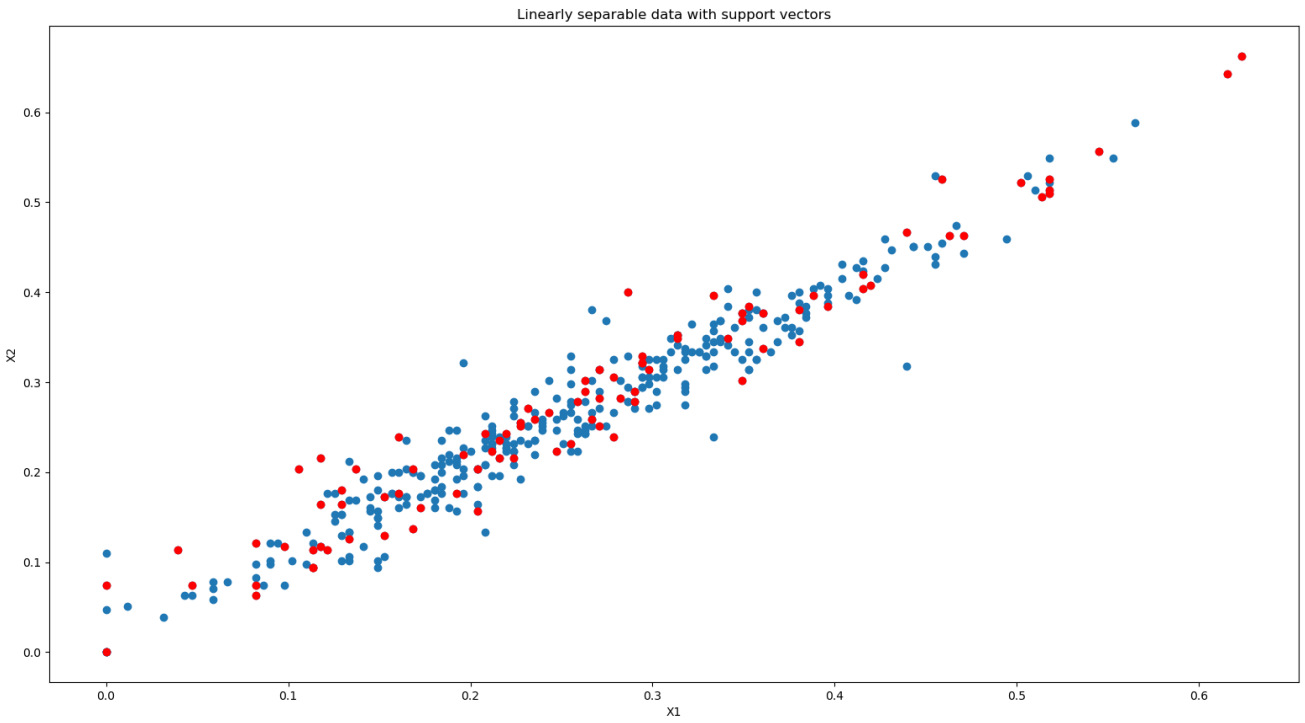
```
# clf = svm.SVC(kernel='linear')
clf = svm.SVC(kernel='rbf')
# clf = svm.SVC(kernel='poly')


clf = clf.fit(X_train, y_train)


predictions = clf.predict(X_test)


matrix = plot_confusion_matrix(clf, X_test, y_test,
                               cmap=plt.cm.Blues,
                               normalize='true')
plt.title('Confusion matrix for our classifier')
plt.show()
```



که همانطور که به دست می آید، نتایج خوبی به دست آمد.

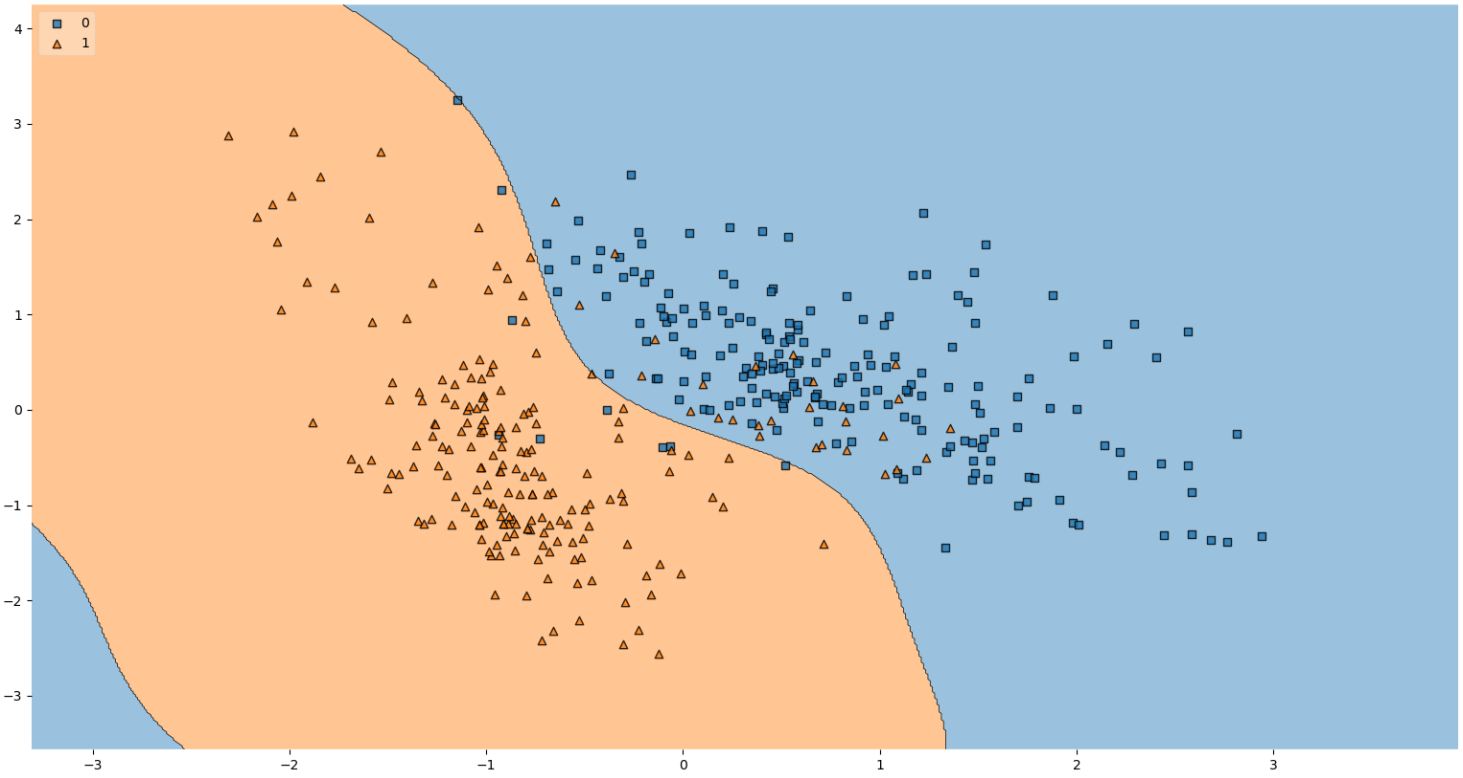
حال بردارهای پشتیبان، SVM را نمایش می دهیم.

```
support_vectors = clf.support_vectors_


plt.scatter(X_train[:,0], X_train[:,1])
plt.scatter(support_vectors[:,0], support_vectors[:,1], color='red')
plt.title('Linearly separable data with support vectors')
plt.xlabel('X1')
plt.ylabel('X2')
plt.show()
```



در نهایت decision boundary را نمایش می دهیم که چون داده ها دو بعدی هستند قبل از نمایش باید یک بعدی شوند، پس از PCA استفاده می کنیم.

```python
from sklearn.decomposition import PCA
from mlxtend.plotting import plot_decision_regions

pca = PCA(n_components = 2)
X_train2 = pca.fit_transform(X_train)
clf.fit(X_train2, y_train)
plot_decision_regions(X_train2, y_train, clf=clf, legend=2)
plt.show()
```



علی شیخ عطار

دکتر عبدی

هوش مصنوعی

دانشکده ی کامپیوتر علم و صنعت

1402