Assignment 3

- Donwload ziptrain.csv and ziptest.csv datasets from https://github.com/vahidpartovinia/ycbs255/ (https://github.com/vahidpartovinia/ycbs255/)

Submission note

Please fill this jupyter notebook. Extract the pdf file as follows. On Jupyter manue go to File/Print Preview, then on Browser menu go to File/Print.

Only PDF Submissions will be graded

1- Differentiate digit 2 from Digit 7

1.1- Two principal components

- Select only digit 2, and digit 7 from ziptrain data set.
- Project ziprain onto two principal components
- Make a scatterplot to confirm wheather or not only two principal components separates digit 2 from digit 7.

In [211]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.decomposition import PCA
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

path = "Desktop/"
ziptrain_file = path + "ziptrain.csv"
ziptest_file = path + "ziptest.csv"

ziptrain = pd.read_csv(ziptrain_file, sep=" ", header = None, na_filter=True)
ziptest = pd.read_csv(ziptest_file, sep=" ", header = None, na_filter=True)
```

In [212]:

```
zipdata = np.loadtxt(ziptrain_file)
zipdata2 = zipdata[zipdata[:, 0] == 2]
zipdata7 = zipdata[zipdata[:, 0] == 7]

ziptest = np.loadtxt(ziptest_file)
ziptest2 = ziptest[ziptest[:, 0] == 2]
ziptest7 = ziptest[ziptest[:, 0] == 7]

zipdata27 = np.vstack([zipdata2, zipdata7])
```

In [213]:

```
pca = PCA(n_components=2)
pca.fit(zipdata27[:, 1:])
```

Out[213]:

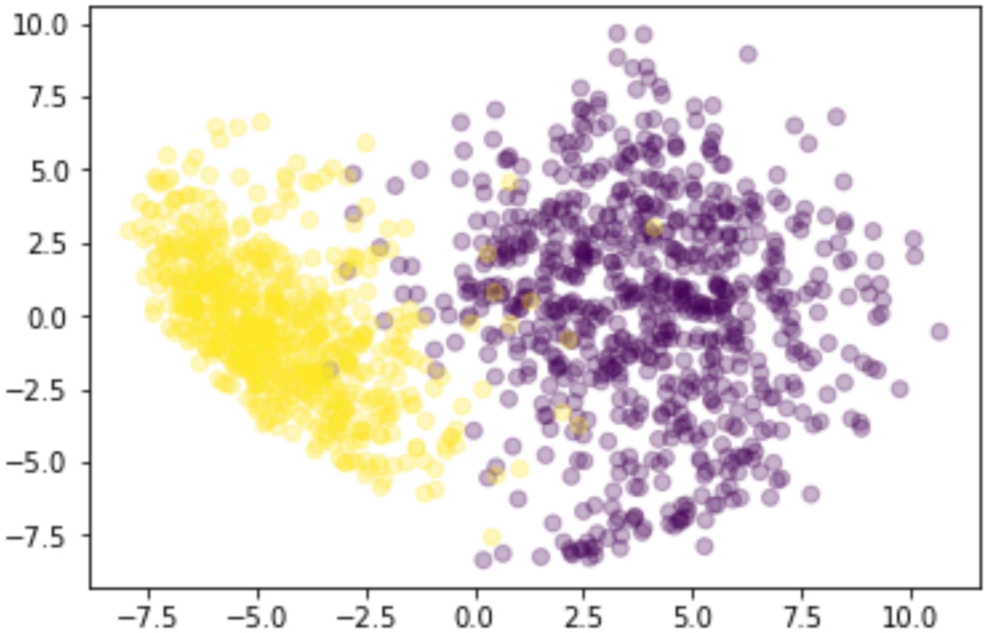
```
PCA(copy=True, iterated_power='auto', n_components=2, random_state=N
one,
  svd_solver='auto', tol=0.0, whiten=False)
```

In [214]:

```
Z = pca.transform(zipdata27[:,1:])
```

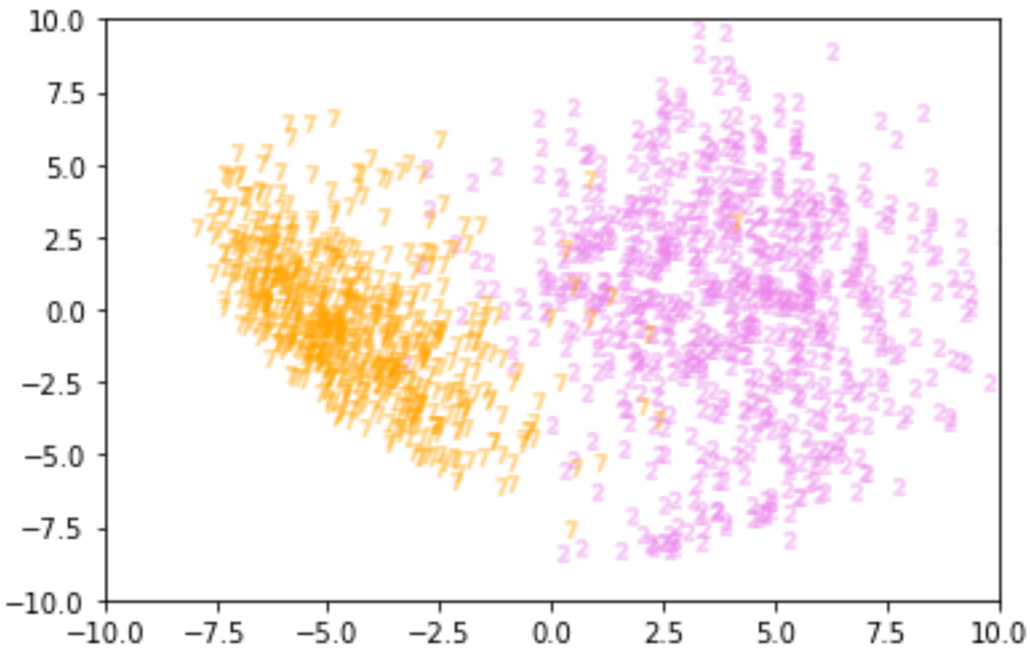
In [215]:

```
plt.scatter(Z[:,0], Z[:,1], c= zipdata27[:,0], alpha=0.3);
plt.show()
```



In [216]:

```
#To better represent the graph

plt.scatter(Z[zipdata27[:,0]==2,0], Z[zipdata27[:,0]==2,1], marker='$2$',
            color='violet', alpha = 0.3);

plt.scatter(Z[zipdata27[:,0]==7,0], Z[zipdata27[:,0]==7,1], marker='$7$',
            color='orange', alpha = 0.3);

plt.xlim([-10,10])
plt.ylim([-10,10])
plt.show()
```



Answer of question number 1.1:

Using two principal components separates marjority of the digit 2 from digit 7. However, a minority of the plots still mixed up from one another.

1.2- Logistic regression

- Fit a logistic regression to separate digit 2 from digit 7 over the projected 2 principal components. Remember in logistic regression, classes are differentiated using 0 and 1 (and not 2 or 7).
- Build the confusion matrix on ziptest and check how well the model works on the test data.

In [217]:

```
X_data = zipdata27[:, 1:]
y_data = zipdata27[:, 0]
X_test = ziptest27[:, 1:]
y_test = ziptest27[:, 0]
```

In [218]:

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_data)

lr.fit(X_pca,y_data)
```

Out[218]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_interce
pt=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jo
bs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0
.0001,
          verbose=0, warm_start=False)
```

In [219]:

```
X_pca_test = pca.fit_transform(X_test)
y_test_pred = lr.predict(X_pca_test)
print(confusion_matrix(y_test, y_test_pred))
```

```
[[187  11]
 [ 14 133]]
```

Answer of question number 1.2:

From the confusion metrix, the number of false positive is 11 and false negative is 14. Comparing with the correctly predicted amount (187+133), we can conclude that the performance of this model is good.

2 -Multiple principal components

- Project train data onto "m = 2, 3, ..." principal components.
- Choose an "m" so that the classification of digit 2 and 7 is the most precise on ziptest.

In [220]:

```python
#The range of m is set to 2 to 10 for the analysis
from sklearn.metrics import classification_report
for m in range(2, 10):
    lr = LogisticRegression()
    pca = PCA(n_components=m)
    X_pca = pca.fit_transform(X_data)
    lr.fit(X_pca,y_data)
    X_pca_test = pca.fit_transform(X_test)
    y_test_pred = lr.predict(X_pca_test)
    print('The components ' + str(m) + ' the confusion matrix is:')
    print( confusion_matrix(y_test, y_test_pred))
    print('The classification report is:')
    print(classification_report(y_test, y_test_pred))
```

```
The components 2 the confusion matrix is:
[[187  11]
 [ 14 133]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.93      0.94      0.94       198
        7.0       0.92      0.90      0.91       147

avg / total       0.93      0.93      0.93       345

The components 3 the confusion matrix is:
[[186  12]
 [ 14 133]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.93      0.94      0.93       198
        7.0       0.92      0.90      0.91       147

avg / total       0.92      0.92      0.92       345

The components 4 the confusion matrix is:
[[188  10]
 [ 11 136]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.94      0.95      0.95       198
        7.0       0.93      0.93      0.93       147

avg / total       0.94      0.94      0.94       345

The components 5 the confusion matrix is:
[[186  12]
 [ 15 132]]
The classification report is:
```

```
             precision    recall  f1-score   support

        2.0       0.93      0.94      0.93       198
        7.0       0.92      0.90      0.91       147

avg / total       0.92      0.92      0.92       345

The components 6 the confusion matrix is:
[[176  22]
 [ 23 124]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.88      0.89      0.89       198
        7.0       0.85      0.84      0.85       147

avg / total       0.87      0.87      0.87       345

The components 7 the confusion matrix is:
[[179  19]
 [ 23 124]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.89      0.90      0.90       198
        7.0       0.87      0.84      0.86       147

avg / total       0.88      0.88      0.88       345

The components 8 the confusion matrix is:
[[176  22]
 [ 22 125]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.89      0.89      0.89       198
        7.0       0.85      0.85      0.85       147

avg / total       0.87      0.87      0.87       345

The components 9 the confusion matrix is:
[[177  21]
 [ 22 125]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.89      0.89      0.89       198
        7.0       0.86      0.85      0.85       147

avg / total       0.88      0.88      0.88       345
```

Answer of question number 2:

-If only consider m in a range in 2 to 10. From the precision indicated above, the precision 0.94 is the highest, when m = 4.

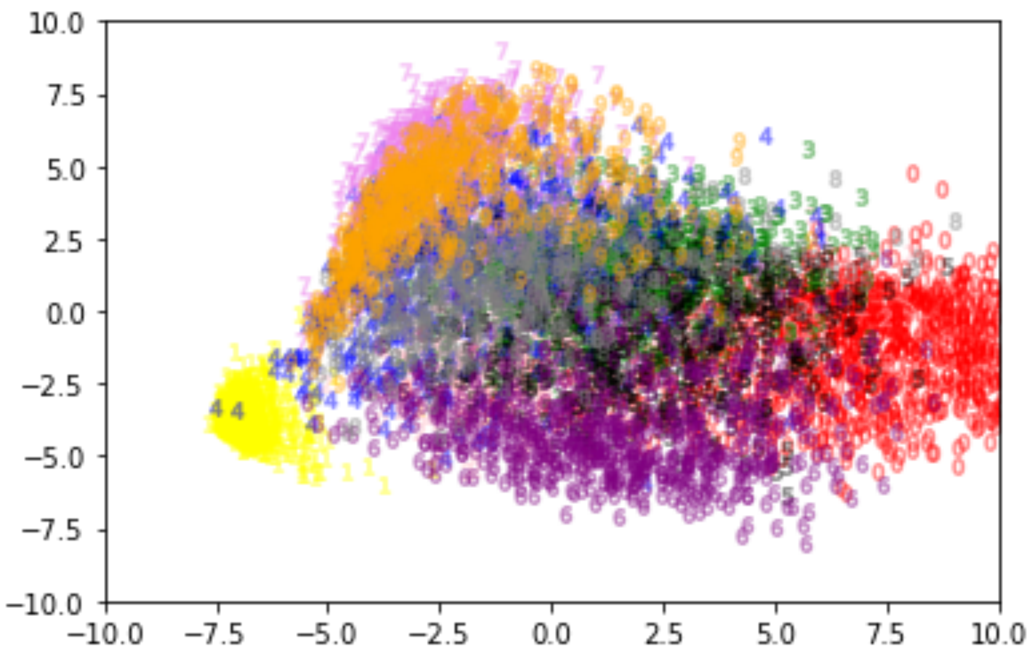
3-Differentiate all digits

- Project ziprain onto two principal components
- Make a scatterplot to confirm wheather or not only two principal components separates all digits properly.
- Use linear discriminant on ziptrain over 256 original pixels and build the confusion matrix of this model over ziptrain
- Use linear disciminant over "m" projected principal components, with the appropriate choice of "m" (where the precision of prediction maximizes over ziptest data set).

In [221]:

```
pca = PCA(n_components=2)

zipdata = np.loadtxt(ziptrain_file)
zipdata0 = zipdata[zipdata[:, 0] == 0]
zipdata1 = zipdata[zipdata[:, 0] == 1]
zipdata2 = zipdata[zipdata[:, 0] == 2]
zipdata3 = zipdata[zipdata[:, 0] == 3]
zipdata4 = zipdata[zipdata[:, 0] == 4]
zipdata5 = zipdata[zipdata[:, 0] == 5]
zipdata6 = zipdata[zipdata[:, 0] == 6]
zipdata7 = zipdata[zipdata[:, 0] == 7]
zipdata8 = zipdata[zipdata[:, 0] == 8]
zipdata9 = zipdata[zipdata[:, 0] == 9]

zipdataAll = np.vstack([zipdata0,zipdata1,zipdata2, zipdata3,zipdata4,zipdata5,z
ipdata6,zipdata7,
                       zipdata8,zipdata9])
pca = PCA(n_components=2)
pca.fit(zipdataAll[:, 1:])

Zall = pca.transform(zipdataAll[:,1:])

plt.scatter(Zall[zipdataAll[:,0]==0,0], Zall[zipdataAll[:,0]==0,1], marker='$0$'
,
            color='red', alpha = 0.3);

plt.scatter(Zall[zipdataAll[:,0]==1,0], Zall[zipdataAll[:,0]==1,1], marker='$1$'
,
            color='yellow', alpha = 0.3);

plt.scatter(Zall[zipdataAll[:,0]==2,0], Zall[zipdataAll[:,0]==2,1], marker='$2$'
,
            color='pink', alpha = 0.3);
```

```
plt.scatter(Zall[zipdataAll[:,0]==3,0], Zall[zipdataAll[:,0]==3,1], marker='$3$'
,
            color='green', alpha = 0.3);

plt.scatter(Zall[zipdataAll[:,0]==4,0], Zall[zipdataAll[:,0]==4,1], marker='$4$'
,
            color='blue', alpha = 0.3);

plt.scatter(Zall[zipdataAll[:,0]==5,0], Zall[zipdataAll[:,0]==5,1], marker='$5$'
,
            color='black', alpha = 0.3);

plt.scatter(Zall[zipdataAll[:,0]==6,0], Zall[zipdataAll[:,0]==6,1], marker='$6$'
,
            color='purple', alpha = 0.3);

plt.scatter(Zall[zipdataAll[:,0]==7,0], Zall[zipdataAll[:,0]==7,1], marker='$7$'
,
            color='violet', alpha = 0.3);

plt.scatter(Zall[zipdataAll[:,0]==8,0], Zall[zipdataAll[:,0]==8,1], marker='$8$'
,
            color='grey', alpha = 0.3);

plt.scatter(Zall[zipdataAll[:,0]==9,0], Zall[zipdataAll[:,0]==9,1], marker='$9$'
,
            color='orange', alpha = 0.3);

plt.xlim([-10,10])
plt.ylim([-10,10])
plt.show()
```



Answer of question number 3.1:

Using on two principle components, we can visualize that there are separations of digits with different colors. However, the separations are not perfectly clear and the there are digits that mixed up together.

In [222]:

```
lda = LinearDiscriminantAnalysis()
pca = PCA(n_components=2)
```

In [223]:

```
X_pca = pca.fit_transform(X_data)
lda.fit(X_pca, y_data)
```

Out[223]:

```
LinearDiscriminantAnalysis(n_components=None, priors=None, shrinkage
=None,
              solver='svd', store_covariance=False, tol=0.0001)
```

In [224]:

```
X_pca_test = pca.fit_transform(X_test)
y_pred_lda = lda.predict(X_pca_test)
```

In [225]:

```
print(confusion_matrix(y_test, y_pred_lda))
```

```
[[181  17]
 [ 11 136]]
```

In [226]:

```
from sklearn.metrics import classification_report
for m in range(2, 10):
    lda = LinearDiscriminantAnalysis()
    pca = PCA(n_components=m)
    X_pca = pca.fit_transform(X_data)
    lda.fit(X_pca, y_data)
    X_pca_test = pca.fit_transform(X_test)
    y_pred_lda = lda.predict(X_pca_test)

    print('The components ' + str(m) + ' the confusion matrix is:')
    print( confusion_matrix(y_test, y_pred_lda))
    print('The classification report is:')
    print(classification_report(y_test, y_pred_lda))
```

```
The components 2 the confusion matrix is:
[[181  17]
 [ 11 136]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.94      0.91      0.93       198
        7.0       0.89      0.93      0.91       147
```

```
avg / total       0.92      0.92      0.92       345

The components 3 the confusion matrix is:
[[182  16]
 [ 10 137]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.95      0.92      0.93       198
        7.0       0.90      0.93      0.91       147

avg / total       0.93      0.92      0.92       345

The components 4 the confusion matrix is:
[[187  11]
 [  9 138]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.95      0.94      0.95       198
        7.0       0.93      0.94      0.93       147

avg / total       0.94      0.94      0.94       345

The components 5 the confusion matrix is:
[[190   8]
 [  8 139]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.96      0.96      0.96       198
        7.0       0.95      0.95      0.95       147

avg / total       0.95      0.95      0.95       345

The components 6 the confusion matrix is:
[[180  18]
 [  9 138]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.95      0.91      0.93       198
        7.0       0.88      0.94      0.91       147

avg / total       0.92      0.92      0.92       345

The components 7 the confusion matrix is:
[[182  16]
 [  9 138]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.95      0.92      0.94       198
```

```
        7.0       0.90      0.94      0.92       147

avg / total       0.93      0.93      0.93       345

The components 8 the confusion matrix is:
[[182  16]
 [ 10 137]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.95      0.92      0.93       198
        7.0       0.90      0.93      0.91       147

avg / total       0.93      0.92      0.92       345

The components 9 the confusion matrix is:
[[181  17]
 [ 10 137]]
The classification report is:
             precision    recall  f1-score   support

        2.0       0.95      0.91      0.93       198
        7.0       0.89      0.93      0.91       147

avg / total       0.92      0.92      0.92       345
```

Answer of question number 3:

The precision get maximized when the number of principle components is 5, with the precision 0.95.