DA HW#7

Q1:

Import model and dataset

```
from PIL import Image
import numpy
from numpy import array
import numpy as np
from tkinter import _flatten
from sklearn.model_selection import train_test_split
from sklearn.preprocessing  import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model  import LogisticRegression
from sklearn import svm
```

✓ 3.4s Python

```
image_matrix = np.zeros((400, 2576))
gender = np.array([])
for j in range(0, 40):
    for i in range(0, 10):
        image = Image.open(r"/Users/4yo/Desktop/NTU_Class/Data_Analyze_Method/ORL_Faces/%s_%s.png" %(j+1, i+1))
        image_array = array(image)
        image_matrix[i+j*10] = image_array.flatten()
gender = [10*[0],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[0],10*[1],10*[0]
         ,10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1]
         ,10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1]
         ,10*[1],10*[0],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1],10*[1]]
gender = list(_flatten(gender))
```

✓ 0.2s Python

Logistic Regression (LR) = 96.25%

```
X = image_matrix
y = gender

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

sc=StandardScaler()
sc.fit(X_train)
X_train_nor=sc.transform(X_train)
X_test_nor=sc.transform(X_test)
```

✓ 0.0s Python

```
lr=LogisticRegression()
lr.fit(X_train_nor,y_train)
print("Accuracy of LR model: ",100*lr.score(X_test_nor, y_test),"%")
```

✓ 0.1s Python

```
Accuracy of LR model:  96.25 %
```

k-Nearest Neighbors (*k*NN) = 97.5% (Usually set default neighbors = 5, if we set k = 6, we can get approximately 98.5% result.)

```
knn = KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train_nor,y_train)
print("Accuracy of KNN model: ",100*knn.score(X_test_nor, y_test),"%")
```

✓ 0.0s Python

```
Accuracy of KNN model:  97.5 %
```

Support Vector Machine (SVM) = 88.75%

```
clf=svm.SVC(kernel='linear',C=1)
clf.fit(X_train,y_train)
print("Accuracy of SVM model: ",100*clf.score(X_test_nor, y_test),"%")
```

✓ 0.1s Python

```
Accuracy of SVM model:  88.75 %
```

In this case, *k*-Nearest Neighbors has better prediction result than LR and SVM while n_neighbor = 5. The result might be different while the parameters are different.

Q2:

Import 1_1.png and using LASSO to find the important pixels.

```
from sklearn.linear_model import Lasso
import matplotlib.pyplot as plt
import math
```

✓ 0.4s Python

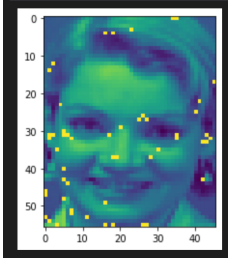
```
lasso = Lasso(alpha = 0.001,normalize = True)
lasso.fit(X, y)
```

✓ 0.1s Python

```
Lasso(alpha=0.001, normalize=True)
```

```
important_pixels = []
for i in range(len(lasso.coef_)):
    if lasso.coef_[i] != 0:
        important_pixels.append(i)
image = Image.open(r"/Users/4yo/Desktop/NTU_Class/Data_Analyze_Method/ORL_Faces/1_1.png")
img_array = np.array(image)
for i in range(0, len(important_pixels)):
    col = math.floor(important_pixels[i]/46)
    row = important_pixels[i]-46*col
    #print("(",col,",", row,")")
    img_array[int(col)][int(row)]=255
plt.imshow(img_array, interpolation='nearest')
plt.show()
```

✓ 0.2s Python




```
X_important = np.zeros((400, len(important_pixels)))
X_important = X.T[important_pixels]
X_important = X_important.T
```

✓ 0.0s Python

Split and Normalize the important pixels.

```
# Split data
X_important_train, X_important_test, y_train, y_test = train_test_split(X_important, y, test_size=0.2, random_state=1)

# Normalize
sc=StandardScaler()
sc.fit(X_important_train)
X_important_train_nor=sc.transform(X_important_train)
X_important_test_nor=sc.transform(X_important_test)
```

✓ 0.1s Python

+ 程式碼 + Markdown

Logistic Regression (LR) = 97.5% (After selecting the important pixels by using Lasso)

```
lr=LogisticRegression()
lr.fit(X_important_train_nor,y_train)
print("Accuracy of LR model",100*lr.score(X_important_test_nor, y_test),"%")
```

✓ 0.1s

```
Accuracy of LR model 97.5 %
```

k-Nearest Neighbors (kNN) = 97.5% (After selecting the important pixels by using Lasso)

Support Vector Machine (SVM) = 96.25% (After selecting the important pixels by using Lasso)

In this case, compared to the result generated from EX1, the accuracy of LR model is improved from 96.25% to 97.5%, and the accuracy of SVM model is improved from 88.75% to 96.25%.

Q3:

Import model and dataset

```python
import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the dataset
auto = pd.read_csv('/Users/4yo/Desktop/NTU_Class/Data_Analyze_Method/DA_Demo.csv')
```

✓ 0.8s Python

```python
# Create X and y
X = auto[['mpg', 'cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'model year']]
y = auto['origin']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

✓ 0.0s Python

Logistic Regression (LR) = 73.72%

```python
# Create and fit the model
lr = LogisticRegression(multi_class='multinomial', solver='newton-cg')
lr.fit(X_train, y_train)

# Make predictions on the test set
lr_pred = lr.predict(X_test)

# Evaluate the model
lr_accuracy = accuracy_score(y_test, lr_pred)
print('Accuracy of LR model: ', 100*lr_accuracy,"%")
```

✓ 0.2s Python

```
Accuracy of LR model:  73.72881355932203 %
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-packages/sklearn/utils/optimize.py:203: ConvergenceWarning: newton-cg failed to converg
  "number of iterations.", ConvergenceWarning)
```

k-Nearest Neighbors (kNN) = 64.4%

```python
# Create and fit the model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Make predictions on the test set
knn_pred = knn.predict(X_test)

# Evaluate the model
knn_accuracy = accuracy_score(y_test, knn_pred)
print('Accuracy of KNN model: ', 100*knn_accuracy,"%")
```

✓ 0.0s Python

```
Accuracy of KNN model:  64.40677966101694 %
```

Support Vector Machine (SVM) = 71.18%

```python
# Create and fit the model
svc = SVC(kernel='linear', C=1, decision_function_shape='ovr')
svc.fit(X_train, y_train)

# Make predictions on the test set
svc_pred = svc.predict(X_test)

# Evaluate the model
svc_accuracy = accuracy_score(y_test, svc_pred)
print('Accuracy of SVM model: ', 100*svc_accuracy,"%")
```

✓ 4.6s Python

```
Accuracy of SVM model:  71.1864406779661 %
```

In this case, LR and SVM models have the better performance than KNN model while k = 5. LR model has the highest accuracy which is 73.72%.