B10703049 財金二 柯宥圻

Homework 5 Question 1

The first question ask us to practice and discuss the results of LR, KNN and SVM on the ORLface dataset. The preprocessing is performed several times, so I won't discuss it again.

```
from PIL import Image
import numpy as np
import pandas as pd
data = []
for i in range(1, 41):
    for j in range(1,11):
        image_dir = f"C:/Users/user/Desktop/課程資料/1a DA/ORL faces/{i}_{j}.png"
        img = Image.open(image_dir)
        img_array = np.asarray(img)
        data.append(img_array.flatten())
data = np.array(data)

gender = 
[0,1,1,1,1,1,1,0,1,0,1,0,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1]
genders = []
for i in gender:
    for _ in range(10):
        genders.append(i)
```

After finishing the data preprocessing process, we use train_test_split to split the training set and testing set, and we choose a random state to ensure that the training data and testing data be the same whenever the code is performed.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data, genders, test_size=0.2, random_state=212)
```

The first classfier is logistic regression, we take the penalty l2 to get better results, and define the max iteration in order to avoid the warning message. The message tells you that the model reached the upper limit of iterations during training. This may be because the model is experiencing convergence issues, it may be due to insufficient data or more tuning is required. Therefore, I should define the max iteration to get rid of these issues. We can see that the Accuracy is rather high.

```
'''Classfier 1: Logistic regression'''
from sklearn.linear_model import LogisticRegression
classifier_LR = 
LogisticRegression(max_iter=5000,penalty='l2',random_state=0)
classifier_LR.fit(X_train, y_train)
accuracy = classifier_LR.score(X_test, y_test)
print("Logistic regression Accuracy: {:.2f}%".format(accuracy * 100))
```

```
Logistic regression Accuracy: 97.50%
```

The second classfier is KNN. we take n_neighbors be 5. The accuracy is higher.

```
'''Classfier 2: KNN'''
from sklearn.neighbors import KNeighborsClassifier
classifier_KNN = KNeighborsClassifier(n_neighbors=5)
classifier_KNN.fit(X_train, y_train)
accuracy = classifier_KNN.score(X_test, y_test)
print("KNN Accuracy: {:.2f}%".format(accuracy * 100))
```

```
KNN Accuracy: 98.75%
```

The third classfier is SVC, we use kernal=linear to this data. There are many kernal to choose in SVC model, but in this case, linear model is great enough and better describe the ORLFace dataset. We choose a random state to ensure the same result, again. We can see the Accuracy is as high as the LR methods.

```
'''Classfier 3: SVM(linear)'''
from sklearn.svm import SVC
classifier_SVC = SVC(kernel='linear', random_state=622)
classifier_SVC.fit(X_train, y_train)
accuracy = classifier_SVC.score(X_test, y_test)
print("SVM Accuracy: {:.2f}%".format(accuracy * 100))
```

```
SVM Accuracy: 97.50%
```

Homework 5 Question 2

In this problem, we have to consider the parsimonious principle in modelling. Therefore, we should take a few variables to get as close as possible to the result in EX1. Since the details are not defined, I use my own way to define the meaning of "getting closer".

When building a model, feature selection can be used to select the most important variables, thereby simplifying the model.According to the notions in HW1, we can use correlation coefficient analysis and tree model for feature selection. Specific steps are as follows:

1. Convert data and genders to a DataFrame (using pandas)
2. Computes the correlation coefficient between each feature and the target variable.
3. Select features with high correlation coefficients.In this case, we selected the features with the absolute value of the correlation coefficient greater than 0.1 as the selected features.

```
df = pd.DataFrame(data)
df['genders'] = genders
correlations = df.corr()['genders'].drop('genders')
selected_features = correlations[abs(correlations) >
0.1].index.tolist()
X = df[selected_features]
y = df['genders']
```

Then in these selected features, we try to calculate the importance of each feature using a decision tree model. After that, we select features with higher importance, agian. We selected features with feature importance greater than 0.1 as selected features.

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X, y)
importance = model.feature_importances_
selected_features = X.columns[importance > 0.1].tolist()
X = df[selected_features]
y = df['genders']

print(selected_features)
print(len(selected_features))
```

```
[1618, 2301]
2
```

We can see only two features are selected, which are really a few. We can use the same ways in EX1 to check the accuracy of each classifiers

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=666)
```

```
'''Classfier 1: Logistic regression'''
from sklearn.linear_model import LogisticRegression
classifier_LR =
LogisticRegression(max_iter=5000,penalty='l2',random_state=0)
classifier_LR.fit(X_train, y_train)
accuracy = classifier_LR.score(X_test, y_test)
print("Feature_Selection - Logistic regression Accuracy: {:.2f}
%".format(accuracy * 100))
```

```
Feature_Selection - Logistic regression Accuracy: 95.00%
```

```
'''Classfier 2: KNN'''
from sklearn.neighbors import KNeighborsClassifier
classifier_KNN = KNeighborsClassifier(n_neighbors=5)
classifier_KNN.fit(X_train, y_train)
accuracy = classifier_KNN.score(X_test, y_test)
print("KNN Accuracy: {:.2f}%".format(accuracy * 100))
```

```
KNN Accuracy: 96.25%
```

```
'''Classfier 3: SVM(linear)'''
from sklearn.svm import SVC
classifier_SVC = SVC(kernel='linear', random_state=622)
classifier_SVC.fit(X_train, y_train)
accuracy = classifier_SVC.score(X_test, y_test)
print("SVM Accuracy: {:.2f}%".format(accuracy * 100))
```

```
SVM Accuracy: 95.00%
```

We can see that although there are only two variables, the accuracy are still high. Precisely, the accuracy of each classifiers just fall by 2-3%, which still remain great results.

Homework 5 Question 3

The question ask us to looj for the multiclass classfier in LR, Knn and SVM. Apply them to analyze AutoMPG and discuss the results. The target is to classify the "origin" of the car and "mpg" can be included in the X.

The "origin" variable is a nominal scales, which are 1, 2, 3, depending on the car's origin countries or regions. Therefore, we can use multiple classfier to classify the orgin with 7 distinct independent variables, including MPG.

In the case, I have found several multiple classifier to solve the problem the codes are shown below. We skip the discussion of data preprocessing, which is spoken before.

```python
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import numpy as np
data = pd.read_csv("C:/Users/user/Desktop/autompg.csv")
data.replace('?', np.nan, inplace=True)
data.dropna(inplace=True)

# Define the independent and dependent variables
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=121)
```

I thought that performing data normalization can help improve the training performance of the model and reduce the possibility of convergence problems. In Python, you can use the StandardScaler from the Scikit-learn library for data normalization.

However, after testing, I found the result become worse in all classfiers.

```python
# # Standardize (all worsen)
# from sklearn.preprocessing import StandardScaler
# scaler = StandardScaler()
# X_train = scaler.fit_transform(X_train)
# X_test = scaler.transform(X_test)
```

In the code above, we first create a StandardScaler object, then use the fit_transform method to normalize the training data and store it in X_train_scaled. Next, we apply the same normalizer object to the test data using the transform method and store it in X_test_scaled.

I think the reason why it become worse is that we use some inappropriate normalization method: If the normalization method chosen is not suitable for a particular dataset or model, it may lead to poor performance. In the dataset, it should not be useful to standarization the data.

In LR method, I found that newton-cg is a great solver. Al

```
"""Logistic regression"""
model = LogisticRegression(multi_class='multinomial', solver='newton-
cg',max_iter=5000,penalty='l2')
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("LogisticRegression Accuracy: {:.2f}%".format(accuracy * 100))
```

```
LogisticRegression Accuracy: 86.08%
```

```
"""KNN"""
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("KNN Accuracy: {:.2f}%".format(accuracy * 100))
```

```
KNN Accuracy: 78.48%
```

```
"""SVM"""
from sklearn.svm import SVC
model = SVC(kernel='linear')
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("SVM Accuracy: {:.2f}%".format(accuracy * 100))
```

```
SVM Accuracy: 88.61%
```