Introduction to Machine Learning (CSCI-UA.473): Homework 3

Instructor: Lerrel Pinto

September 20, 2022

Submission Instructions

You must typeset the answers using LaTeX and compile them into a single PDF file. Name the pdf file as ⟨Your-NetID⟩_hw3.pdf and the notebook containing the coding portion as ⟨Your-NetID⟩_hw3.ipynb. The PDF file should contain solutions to both the theory portion and the coding portion. Submit the files through the following Google Form - https://forms.gle/ud89h4dESqEDVg8S9 The due date is **October 18, 2022, 11:59 PM**. You may discuss the questions with each other but each student must provide their own answer to each question.

Assignment Tasks

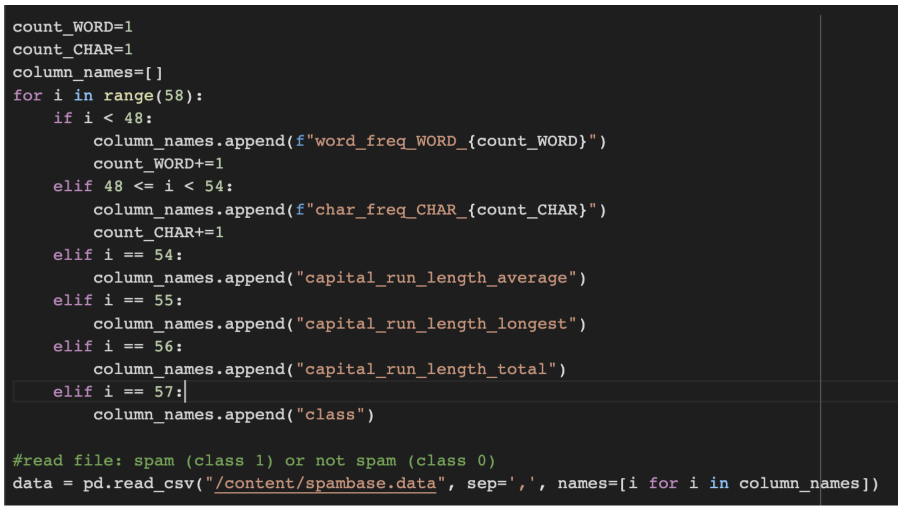
Methodology

1. Mention the libraries used in your solution.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn
import sklearn.preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
```

2. Explain the details of the methodology used to solve the homework - how did you read the data, scaler/normalizer used for the data, the SVM package used, data split, kernel functions, etc.

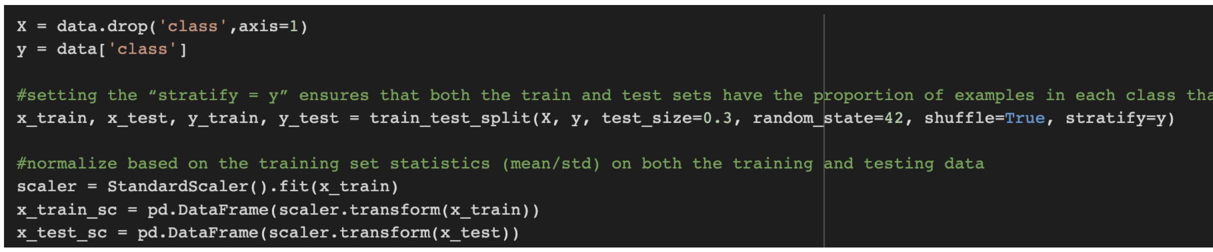
   I use Google Colab for this assignment. I first read the spambase.data file into a dataframe and create column names for each column.



```
count_WORD=1
count_CHAR=1
column_names=[]
for i in range(58):
    if i < 48:
        column_names.append(f"word_freq_WORD_{count_WORD}")
        count_WORD+=1
    elif 48 <= i < 54:
        column_names.append(f"char_freq_CHAR_{count_CHAR}")
        count_CHAR+=1
    elif i == 54:
        column_names.append("capital_run_length_average")
    elif i == 55:
        column_names.append("capital_run_length_longest")
    elif i == 56:
        column_names.append("capital_run_length_total")
    elif i == 57:
        column_names.append("class")

#read file: spam (class 1) or not spam (class 0)
data = pd.read_csv("/content/spambase.data", sep=',', names=[i for i in column_names])
```

Then, I extract the features column into X and the output into y. I use `train_test_split` to randomly split 0.7 of the data as training set (3220 samples) and 0.3 of the data as testing set (1381 samples). I also normalize based on the training set on both the training and testing data using StandardScaler.

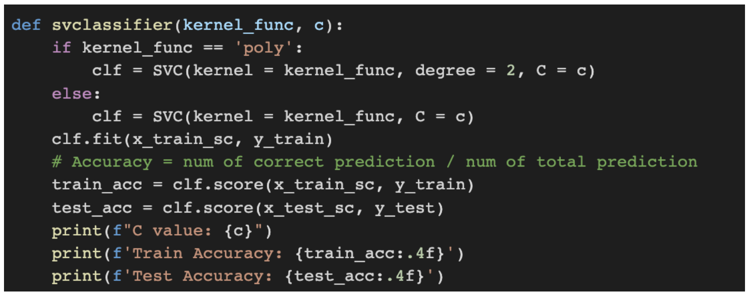


```
X = data.drop('class',axis=1)
y = data['class']

#setting the “stratify = y” ensures that both the train and test sets have the proportion of examples in each class tha
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42, shuffle=True, stratify=y)

#normalize based on the training set statistics (mean/std) on both the training and testing data
scaler = StandardScaler().fit(x_train)
x_train_sc = pd.DataFrame(scaler.transform(x_train))
x_test_sc = pd.DataFrame(scaler.transform(x_test))
```

Next, I write a definition of svclassifier with different functions and different c values to train the data and print out the train accuracy and test accuracy.

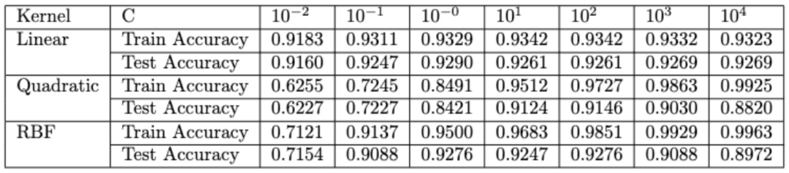


```
def svclassifier(kernel_func, c):
    if kernel_func == 'poly':
        clf = SVC(kernel = kernel_func, degree = 2, C = c)
    else:
        clf = SVC(kernel = kernel_func, C = c)
    clf.fit(x_train_sc, y_train)
    # Accuracy = num of correct prediction / num of total prediction
    train_acc = clf.score(x_train_sc, y_train)
    test_acc = clf.score(x_test_sc, y_test)
    print(f"C value: {c}")
    print(f'Train Accuracy: {train_acc:.4f}')
    print(f'Test Accuracy: {test_acc:.4f}')
```

Experimental Results

1. You have to use each of the following three kernel functions (a) Linear, (b) Quadratic, (c) RBF.

2. For each of the kernels, you have to report training and test set classification accuracy for the best value of generalization constant C. The best C value is the one that provides the best test set accuracy that you have found out by trial of different values of C. Report accuracies in the form of a comparison table, along with the values of C.



Kernel	C	10^{-2}	10^{-1}	10^{-0}	10^{1}	10^{2}	10^{3}	10^{4}
Linear	Train Accuracy	0.9183	0.9311	0.9329	0.9342	0.9342	0.9332	0.9323
	Test Accuracy	0.9160	0.9247	0.9290	0.9261	0.9261	0.9269	0.9269
Quadratic	Train Accuracy	0.6255	0.7245	0.8491	0.9512	0.9727	0.9863	0.9925
	Test Accuracy	0.6227	0.7227	0.8421	0.9124	0.9146	0.9030	0.8820
RBF	Train Accuracy	0.7121	0.9137	0.9500	0.9683	0.9851	0.9929	0.9963
	Test Accuracy	0.7154	0.9088	0.9276	0.9247	0.9276	0.9088	0.8972

   The best c value for linear kernel function is 1.
   The best c value for quadratic kernel function is 100.
   The best c value for rbf kernel function is 1.

3. Provide an intuition for the results observed for different kernels and different values of C.

   The quadratic kernel function and the rbf kernel function have the problem of overfitting on train set, so the test accuracy is lower than the linear kernel function.

   The lower the c values, it has lower penalty on misclassification, so increasing the c vlues could have a higher accuracy on traning set for quadratic kernel function and rbf kernel function, but their performace on testing set is decreasing due to overfitting. For linear kernel function, if the c is too high, then the model will be too strict with the classification, which will lower the performance on traning set and testing set as the c value geting larger and larger.