Support Vector Machines Lab Manual

Ahsan Saadat, PhD

1 Introduction

I have created this document to summarize my expectations from students by providing detailed aspects of how to perform the lab tasks. The purpose of this lab is to provide you with a practical learning experience, enabling you to gain firsthand knowledge of different artificial intelligence techniques and accomplish various objectives.

- Gain knowledge about SVM model.
- Learn about different kernels used in SVM model.
- Learn to implement svm model and perform multi-class classification.

2 Lab Setup

In this lab, the implementation will be done using Python language. Different IDEs can be used for implementing and executing the code. Following are some options.

- Google Colab (https://colab.research.google.com/) (Preferable)
- Jupyter Notebook (https://jupyter.org/)
- Visual Code Studio (https://code.visualstudio.com/)

2.1 Starting with Google Colab

This section is a guide for getting started with Google Colab.

- Sign into your Google account and open the Google Colab website.
- Create a new notebook from the drop-down menu after clicking the files in the menu bar.
- After creating a new notebook, write code in a cell and execute it by clicking the run button at the left side of the cell or by pressing the shift+enter keys.
- Rename your file to the name of the lab and download it as .ipynb file for future use.

3 Preliminary Concept

Support Vector Machines (SVMs) are supervised learning models used for classification and regression tasks. SVMs are particularly effective in scenarios where the data points are not linearly separable in the input space.

The main idea behind SVMs is to find an optimal hyperplane that separates the data points of different classes while maximizing the margin between the classes. The hyperplane is defined as the decision boundary that distinguishes one class from another.

4 Kernel Functions

SVMs can utilize kernel functions to map the data into a higher-dimensional feature space. Kernel functions transform the input data without explicitly computing the coordinates of the data points in the higher-dimensional space. This allows SVMs to handle complex non-linear decision boundaries effectively. Commonly used kernel functions include:

- Gaussian Radial Basis Function (RBF) Kernel: It allows for complex decision boundaries by transforming the data into an infinite-dimensional feature space.

5 Tasks

The following tasks shall be performed for achieving the goals of this lab:

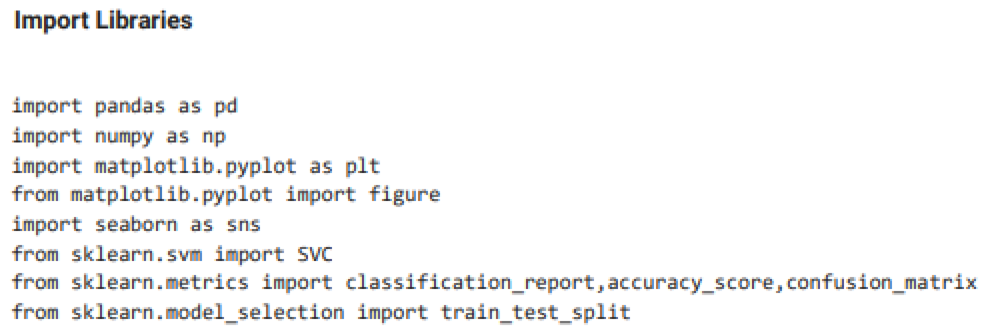
- Implement a svm model to train and test data on the given data set .
  - Divide the data set in training and testing sets - 60% train and 40% test.
  - Use different kernels too see their classification and results.

6 Libraries to Use

- import pandas as pd
- import numpy as np
- import matplotlib.pyplot as plt
- from matplotlib.pyplot import figure
- import seaborn as sns
- from sklearn.svm import SVC
- from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
- from sklearn.model_selection import train_test_split

Help: https://scikit-learn.org/stable/modules/svm.html

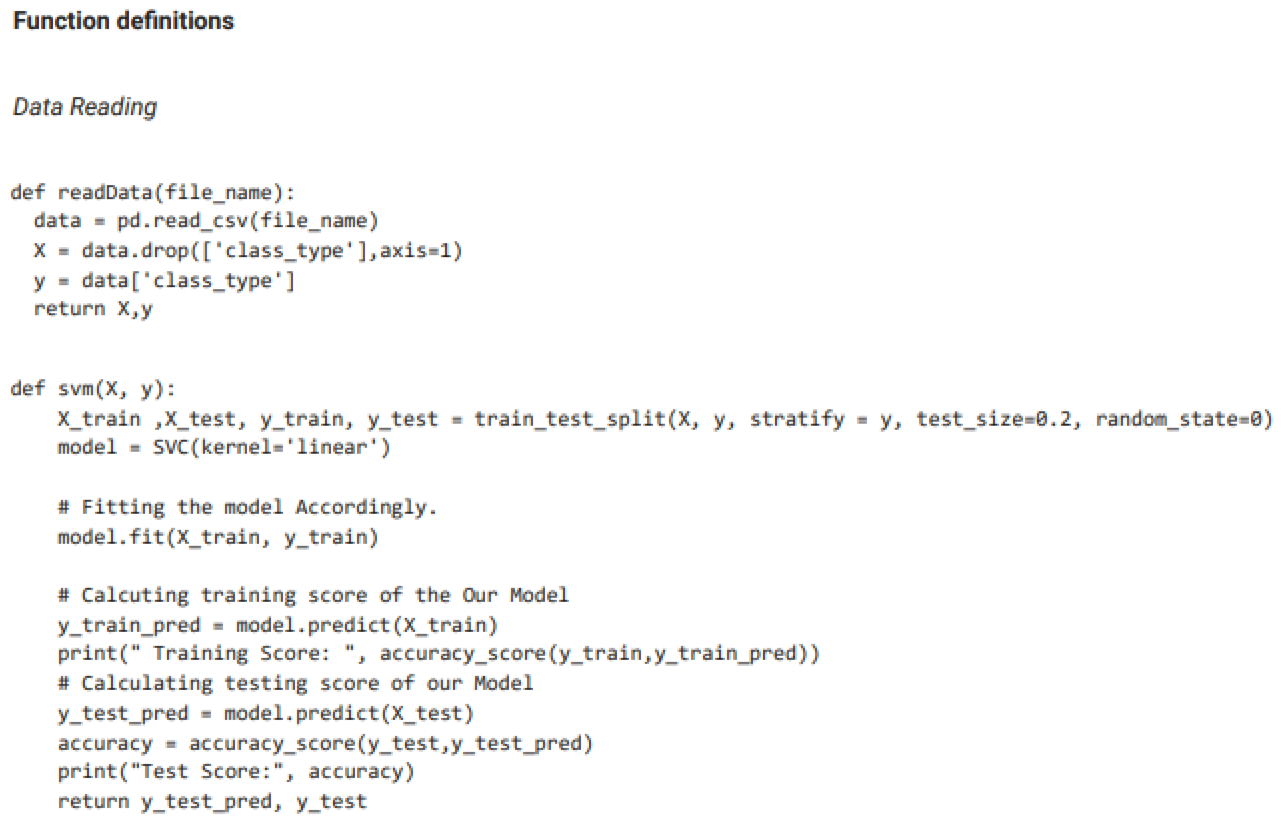
Figure 1: We are importing all the libraries that will be required for performing various tasks such as SVM classification, data split, reading files, plotting maps, etc.



Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
import seaborn as sns
from sklearn.svm import SVC
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
from sklearn.model_selection import train_test_split
```

Figure 2: We are writing a function to read the csv file using pandas library and its read file function. In the next code block, we created a function for training and testing the svm model using the sckit-learn library. The data is divided into training and testing data sets and is used accordingly.



Function definitions

Data Reading

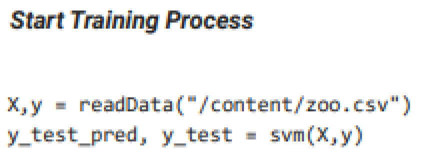
```
def readData(file_name):
  data = pd.read_csv(file_name)
  X = data.drop(['class_type'],axis=1)
  y = data['class_type']
  return X,y


def svm(X, y):
    X_train ,X_test, y_train, y_test = train_test_split(X, y, stratify = y, test_size=0.2, random_state=0)
    model = SVC(kernel='linear')

    # Fitting the model Accordingly.
    model.fit(X_train, y_train)

    # Calcuting training score of the Our Model
    y_train_pred = model.predict(X_train)
    print(" Training Score: ", accuracy_score(y_train,y_train_pred))
    # Calculating testing score of our Model
    y_test_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test,y_test_pred)
    print("Test Score:", accuracy)
    return y_test_pred, y_test
```

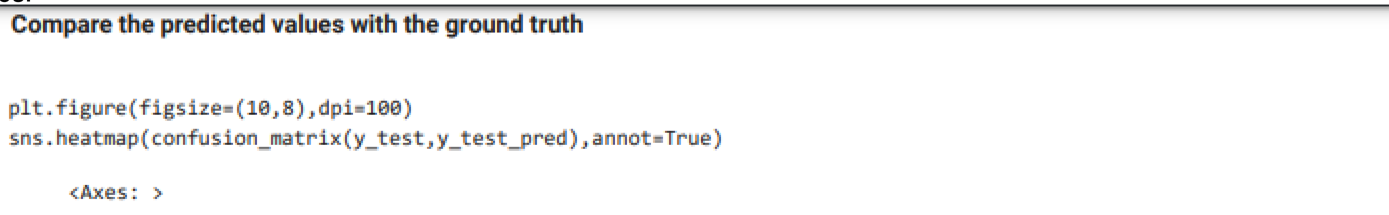
Figure 3: The data file is fed to the read file function and the training and testing process is done by calling the svm function.



Start Training Process

```
X,y = readData("/content/zoo.csv")
y_test_pred, y_test = svm(X,y)
```

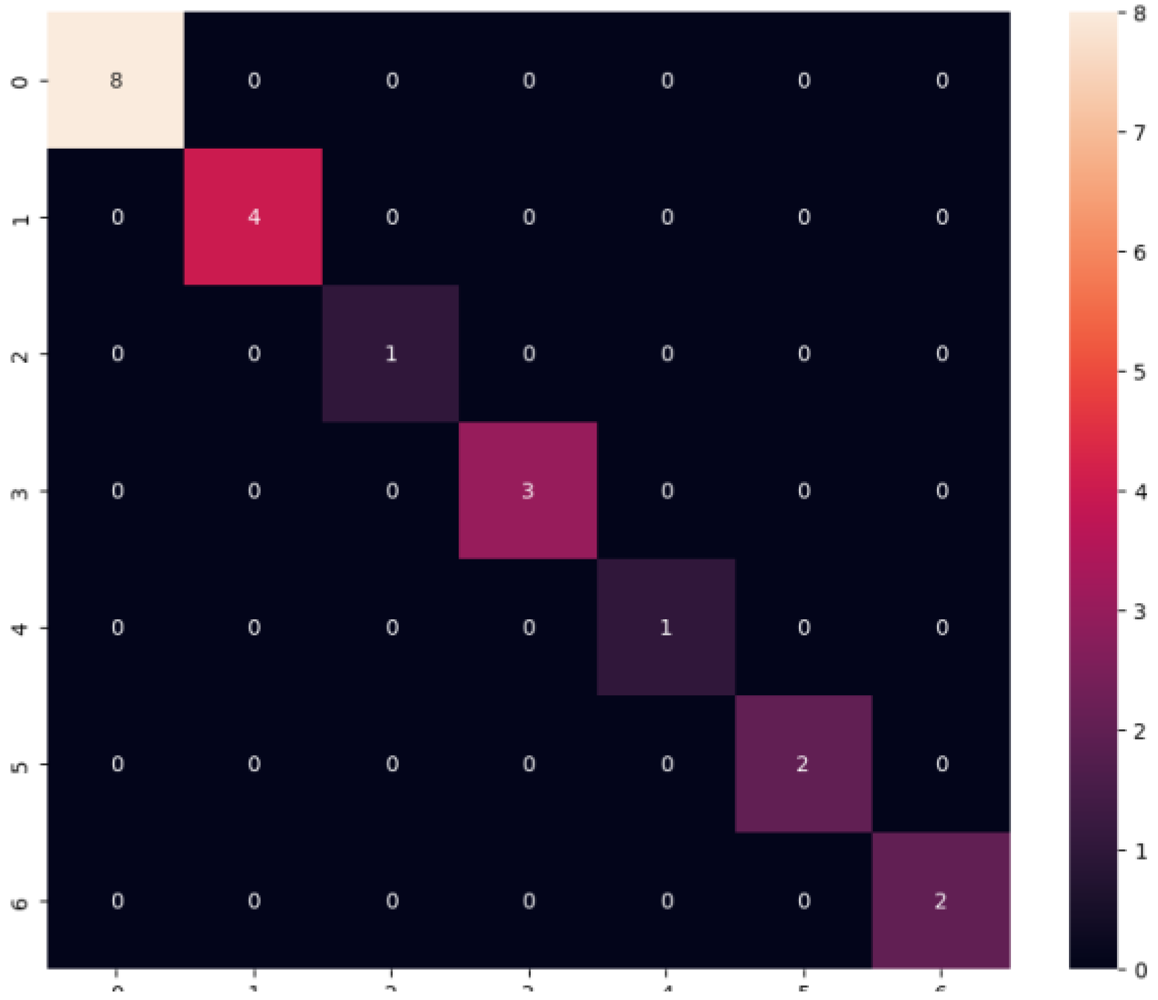
Figure 4: The results are mapped on a confusion matrix to see the prediction accuracy and the predicted values.



Compare the predicted values with the ground truth

```
plt.figure(figsize=(10,8),dpi=100)
sns.heatmap(confusion_matrix(y_test,y_test_pred),annot=True)
```

```
<Axes: >
```



0
1
2
3
4
5
6
8 0 0 0 0 0 0
0 4 0 0 0 0 0
0 0 1 0 0 0 0
0 0 0 3 0 0 0
0 0 0 0 1 0 0
0 0 0 0 0 2 0
0 0 0 0 0 0 2
8
7
6
5
4
3
2
1
0