Pattern Recognition and Machine Learning
2022 Winter Semester

Report - Lab Assignment 10

Ayush Abrol B20AI052

Question - 1

Reading Data and Preprocessing

- Downloaded the spambase dataset which contained spambase.data, spambase.names and spambase.DOCUMENTATION. We read the data form spambase.data and got the attribute header and other information from spambase.names
- Created the pandas Dataframe and gave header names.
- Found the value of each class where
    - 0 (Non - spam) = 2788
    - 1 (Spam) = 1813

Normalizing the data

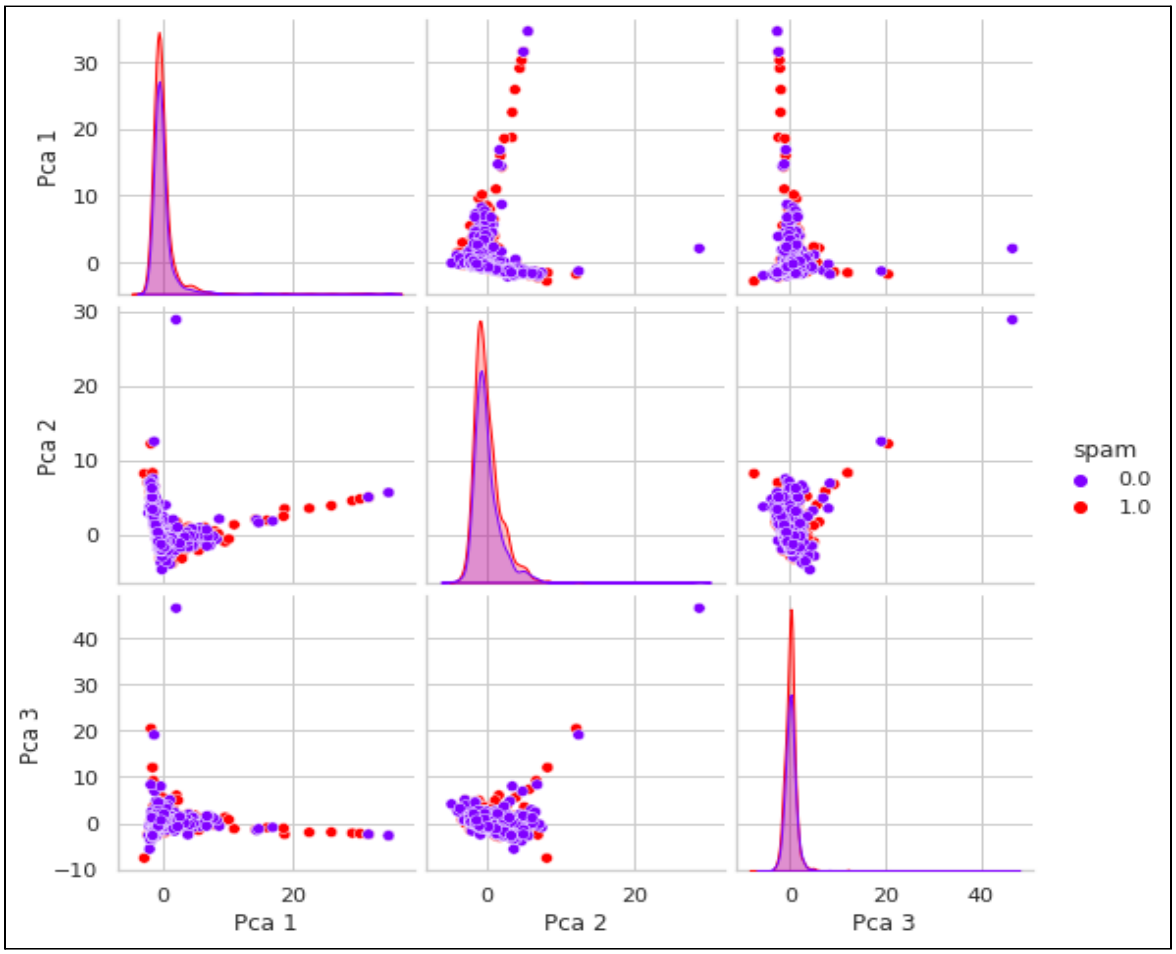
- Used StandardScalar of sklearn.preprocessing and scaled all the continuous variables because SVM requires the data to be scaled uniformly before the model needs to be fitted.

Splitting the dataset into 70:30 ratio

- Used the train_test_split of sklearn.model_selection to split the data randomly into training and testing in the ratio 70-30 respectively.

PCA for better visualization of the dataset

- Used PCA (Principal Component Analysis) of sklearn.decomposition with n_components = 3 so that we can visualize our dataset better with fewer features.
- Used seaborn library to plot a pairplot between the 3 principal components obtained.
- Got the following pairplot as a result:



Used the Support Vector Machine to classify our email data into spam and non-spam categories.

Methodology

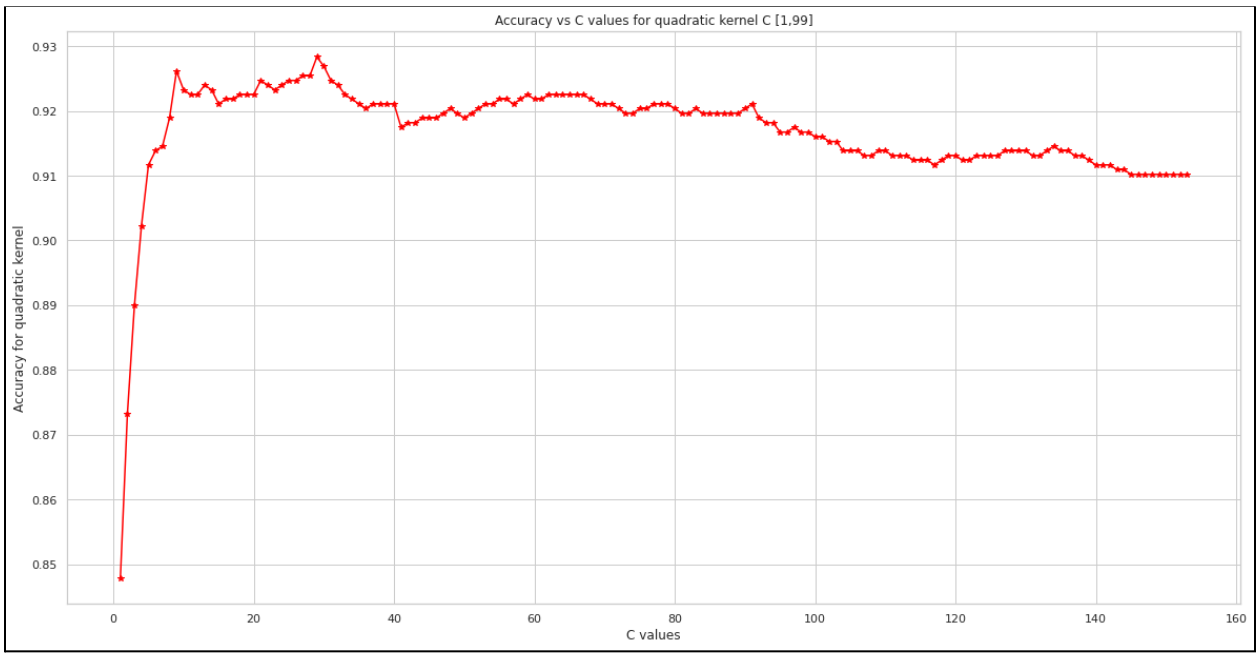
- Used the SVC (C - Support Vector Classification) class of the sklearn.svm package.
- SVC returns the probability of the sample for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute classes_. The probability model is created using cross validation, so the results can be slightly different than those obtained by predict.
- The algorithm creates a line or a hyperplane which separates the data into classes.

Experimental Results

Using kernel = poly with degree = 2 (Quadratic) for finding the best fit line for best accuracy

- Iterated through the values of C from 1 to 100 and fitted the model on X_train and y_train and got predictions on testing data for every value of C.
- Used the predictions, found accuracy score for every value of C from 1 to 100 on y_test.

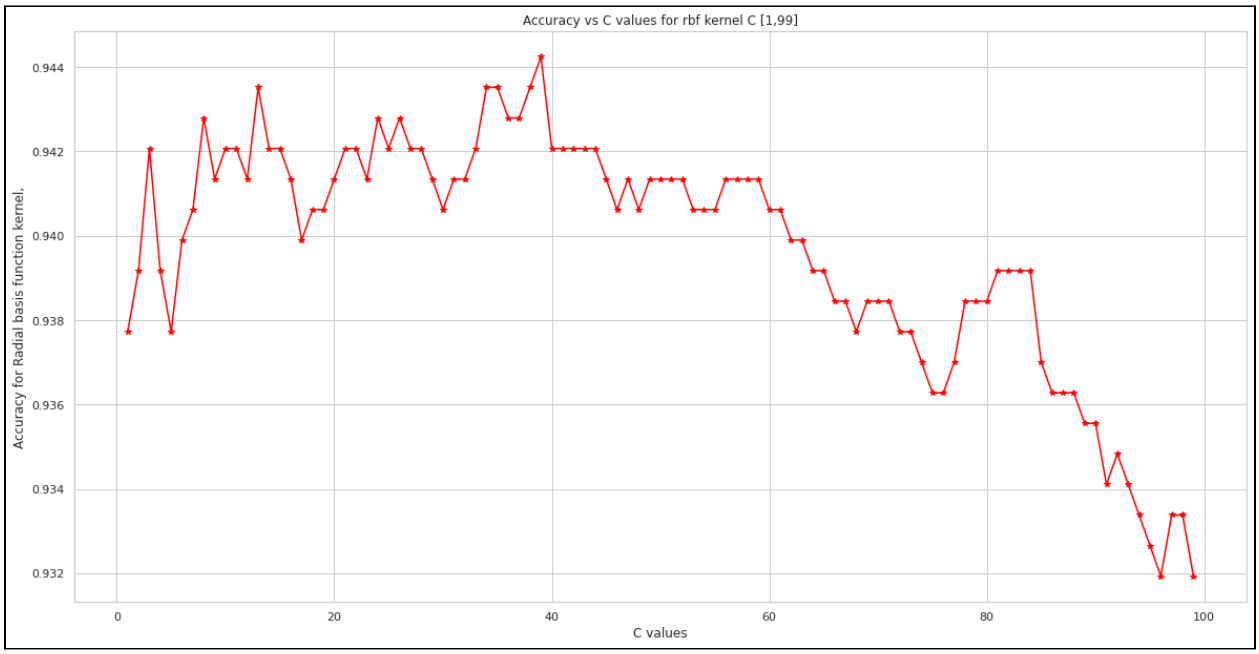
- Created an array of all C values and corresponding accuracies obtained.
- Plotted the graph between C values and corresponding accuracies using matplotlib.pyplot.
- Got the following graph as the result:



- The best value of C obtained for the quadratic kernel is 29 with an accuracy of 92.83128167994207.

Using kernel = rbf (Radial Basis Function) for finding the best fit line for best accuracy

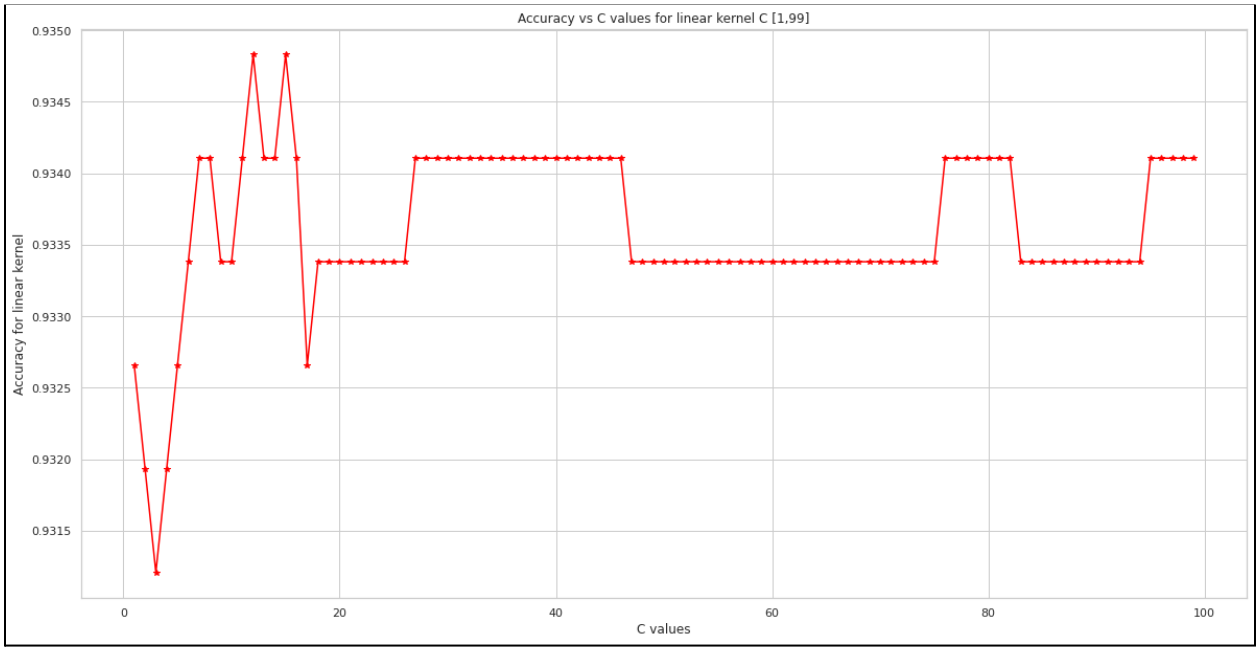
- Iterated through the values of C from 1 to 100 and fitted the model on X_train and y_train and got predictions on testing data for every value of C.
- Used the predictions, found accuracy score for every value of C from 1 to 100 on y_test.
- Created an array of all C values and corresponding accuracies obtained.
- Plotted the graph between C values and corresponding accuracies using matplotlib.pyplot.
- Got the following graph as the result:



- The best value of C obtained for the quadratic kernel is 39 with an accuracy of 94.4243309155105.

Using kernel = linear for finding the best fit line for best accuracy

- Iterated through the values of C from 1 to 100 and fitted the model on X_train and y_train and got predictions on testing data for every value of C.
- Used the predictions, found accuracy score for every value of C from 1 to 100 on y_test.
- Created an array of all C values and corresponding accuracies obtained.
- Plotted the graph between C values and corresponding accuracies using matplotlib.pyplot.
- Got the following graph as the result:



- The best value of C obtained for the quadratic kernel is 12 with an accuracy of 93.48298334540188.

Created a comparison table for the accuracies for best C values by using the three different kernels

- Got the following comparison table:

	Kernel	C	Accuracy
0	Linear	12	0.934830
1	Quadratic	29	0.928313
2	rbf	39	0.944243