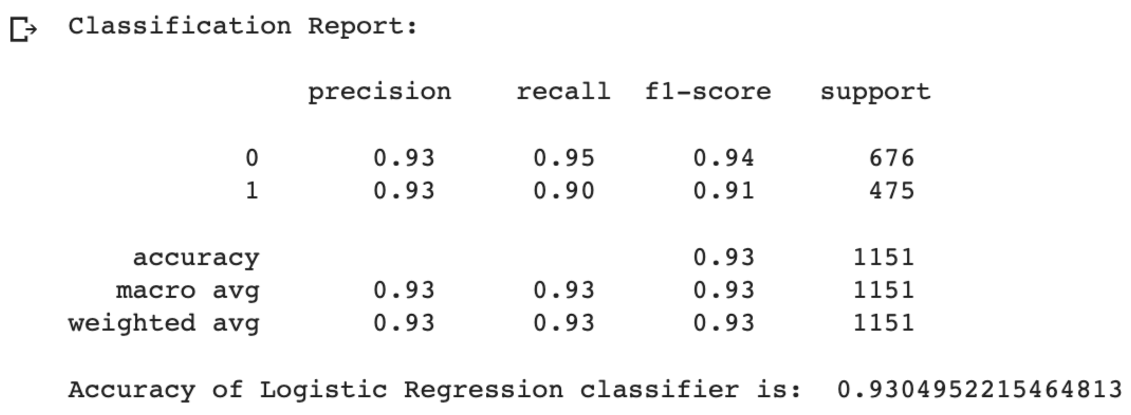
(2022W) COMP-5313-WDE/WB - Artificial Intelligence Computer Science

Project 2

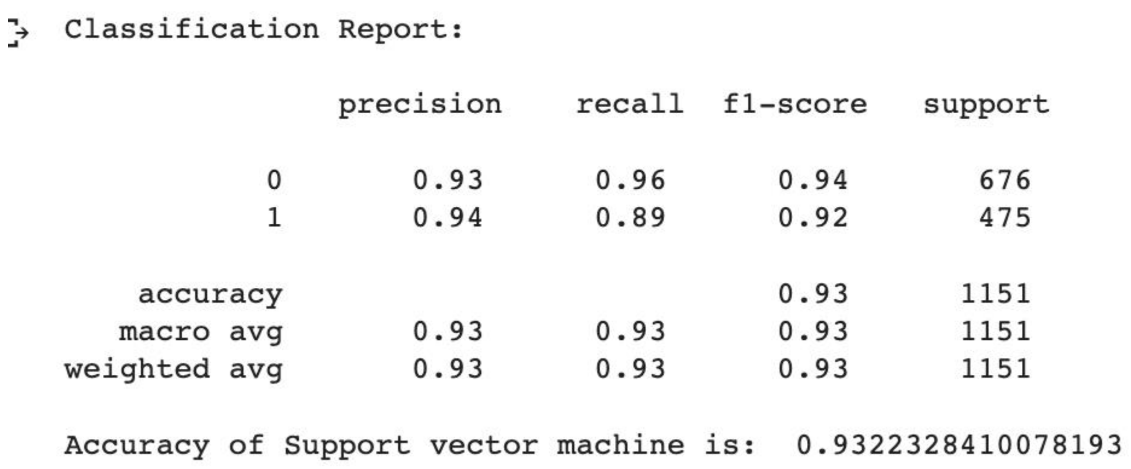
In this project Ihave imported all the libraries first. After that load the dataset and check that whether any null values is present or not and used Traditional Machine Learning Classifiers from SKLearn for classify that message is a spam or not.

LogisticRegression



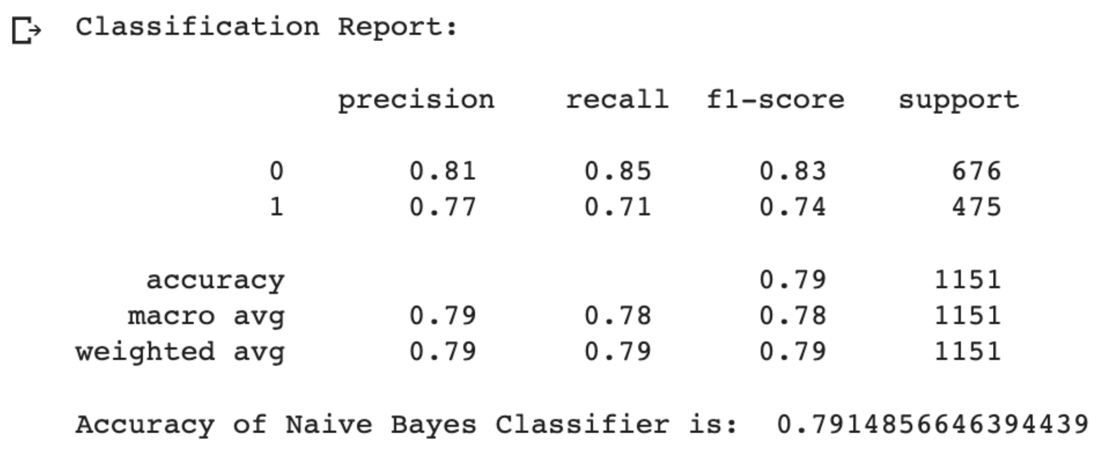
```
Classification Report:

              precision    recall  f1-score   support

           0       0.93      0.95      0.94       676
           1       0.93      0.90      0.91       475

    accuracy                           0.93      1151
   macro avg       0.93      0.93      0.93      1151
weighted avg       0.93      0.93      0.93      1151

Accuracy of Logistic Regression classifier is:  0.9304952215464813
```

SVC



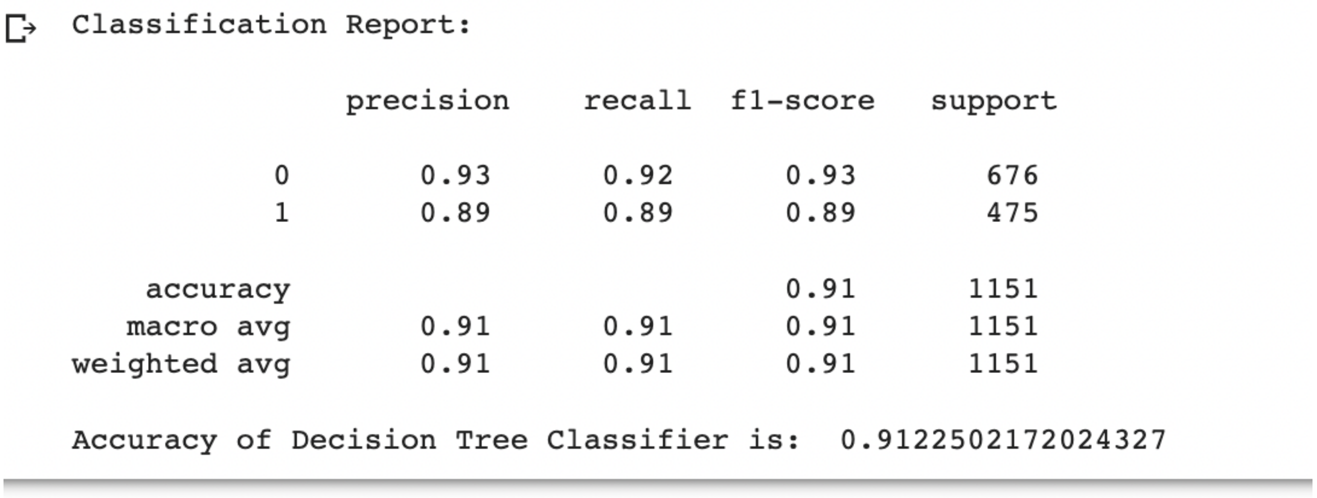
```
Classification Report:

              precision    recall  f1-score   support

           0       0.93      0.96      0.94       676
           1       0.94      0.89      0.92       475

    accuracy                           0.93      1151
   macro avg       0.93      0.93      0.93      1151
weighted avg       0.93      0.93      0.93      1151

Accuracy of Support vector machine is:  0.9322328410078193
```

MultinomialNB



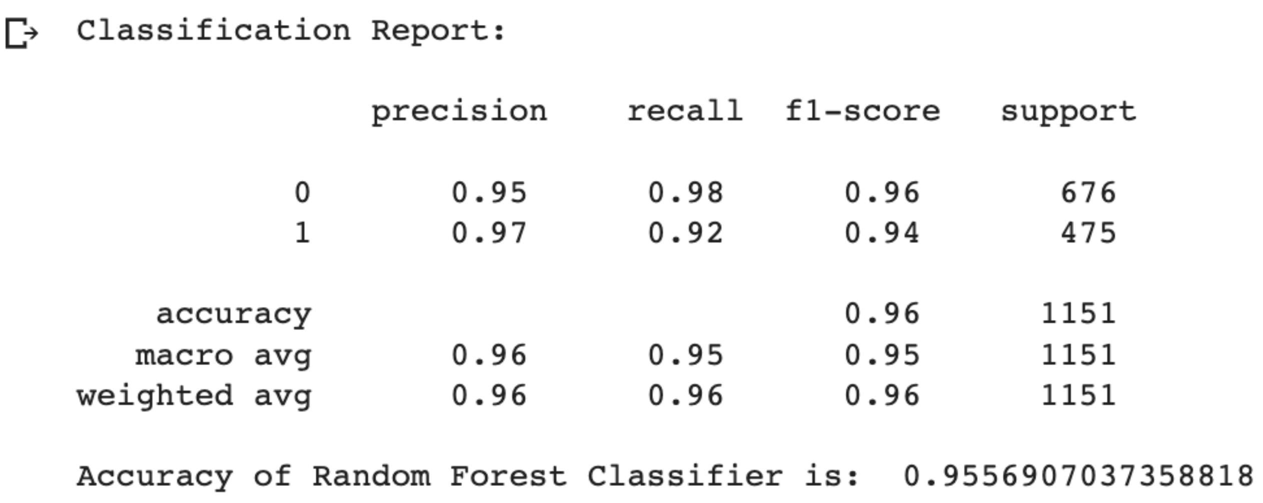
```
Classification Report:

              precision    recall  f1-score   support

           0       0.81      0.85      0.83       676
           1       0.77      0.71      0.74       475

    accuracy                           0.79      1151
   macro avg       0.79      0.78      0.78      1151
weighted avg       0.79      0.79      0.79      1151

Accuracy of Naive Bayes Classifier is:  0.7914856646394439
```

DecisionTreeClassifier



```
Classification Report:

              precision    recall  f1-score   support

           0       0.93      0.92      0.93       676
           1       0.89      0.89      0.89       475

    accuracy                           0.91      1151
   macro avg       0.91      0.91      0.91      1151
weighted avg       0.91      0.91      0.91      1151

Accuracy of Decision Tree Classifier is:  0.9122502172024327
```

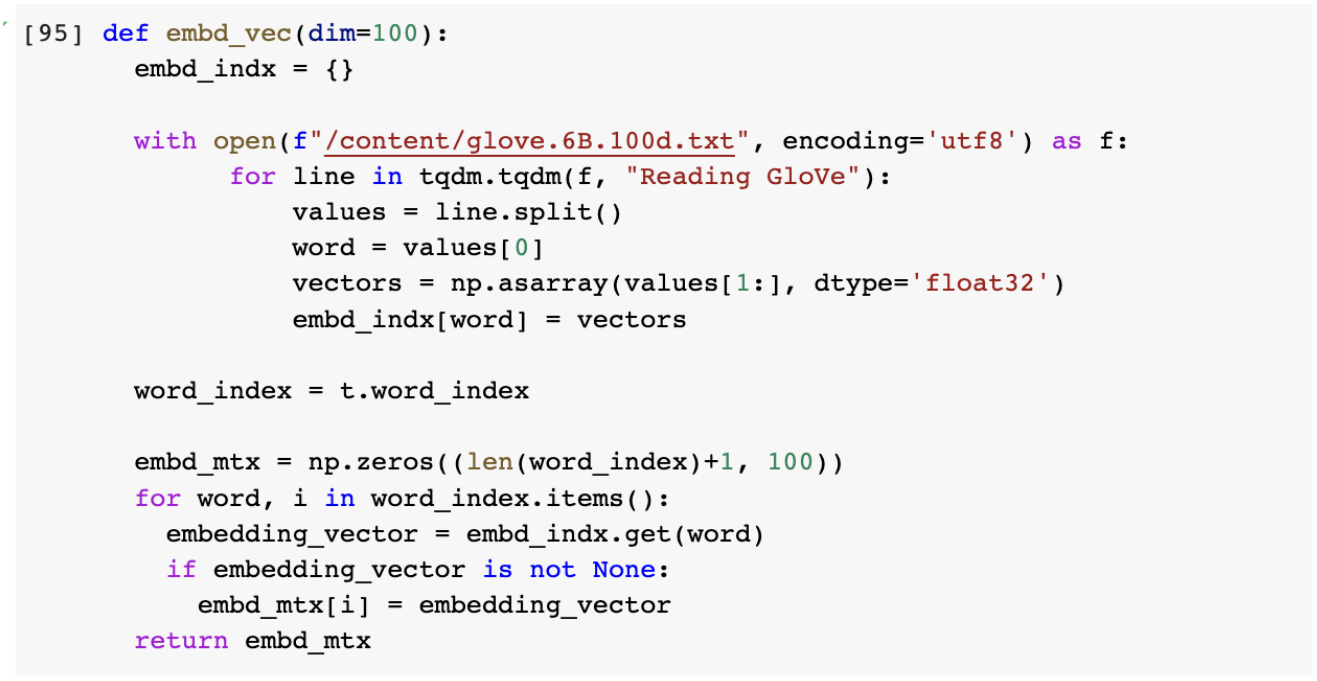
RandomForestClassifier



```
Classification Report:

              precision    recall  f1-score   support

           0       0.95      0.98      0.96       676
           1       0.97      0.92      0.94       475

    accuracy                           0.96      1151
   macro avg       0.96      0.95      0.95      1151
weighted avg       0.96      0.96      0.96      1151

Accuracy of Random Forest Classifier is:  0.9556907037358818
```

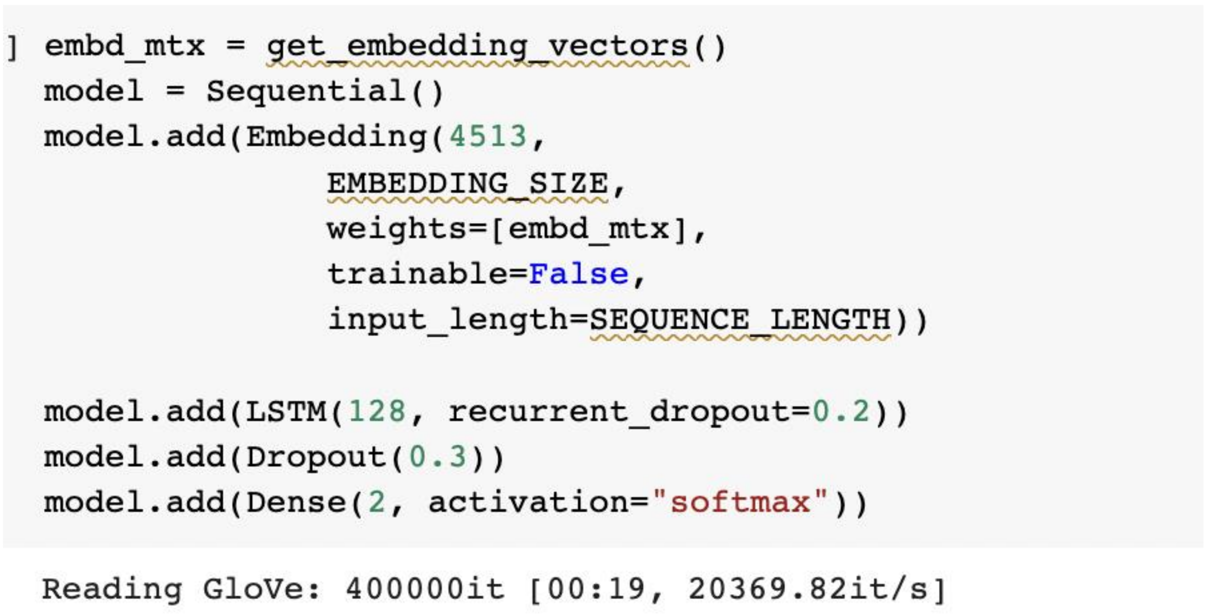
In the second part I did word embedding with deep learning using the glove model first we imported the dataset after that we used the token sizer for it and then split the dataset.

Afterthat we used the word embedding method and import the glove model in it.



```
[95] def embd_vec(dim=100):
       embd_indx = {}

       with open(f"/content/glove.6B.100d.txt", encoding='utf8') as f:
             for line in tqdm.tqdm(f, "Reading GloVe"):
                 values = line.split()
                 word = values[0]
                 vectors = np.asarray(values[1:], dtype='float32')
                 embd_indx[word] = vectors

       word_index = t.word_index

       embd_mtx = np.zeros((len(word_index)+1, 100))
       for word, i in word_index.items():
         embedding_vector = embd_indx.get(word)
         if embedding_vector is not None:
           embd_mtx[i] = embedding_vector
       return embd_mtx
```

And then we added the lstm model in it



```
] embd_mtx = get_embedding_vectors()
  model = Sequential()
  model.add(Embedding(4513,
                  EMBEDDING_SIZE,
                  weights=[embd_mtx],
                  trainable=False,
                  input_length=SEQUENCE_LENGTH))

  model.add(LSTM(128, recurrent_dropout=0.2))
  model.add(Dropout(0.3))
  model.add(Dense(2, activation="softmax"))

  Reading GloVe: 400000it [00:19, 20369.82it/s]
```

The final accuracy which we got is around 83.49%

```
[100] # get the loss and metrics
      output = model.evaluate(testX, testY)
      # extract those
      loss = output[0]
      accuracy = output[1]

      print(f"[+] Accuracy: {accuracy*100:.2f}%")
```

```
36/36 [==============================] - 1s 34ms/step - loss: 0.3751 - accuracy:
[+] Accuracy: 83.49%
```