For CountVectorizer + LogisticRegression :

```
 Result
Accuracy        : 0.9663
Precision (avg): 0.9685
Recall (avg)    : 0.9644
F1-score (avg)  : 0.9659
              precision    recall  f1-score   support

     business       0.93      0.98      0.95       102
entertainment       0.97      0.99      0.98        77
     politics       0.99      0.90      0.94        84
        sport       0.98      1.00      0.99       102
         tech       0.97      0.95      0.96        80

     accuracy                           0.97       445
    macro avg       0.97      0.96      0.97       445
 weighted avg       0.97      0.97      0.97       445
```

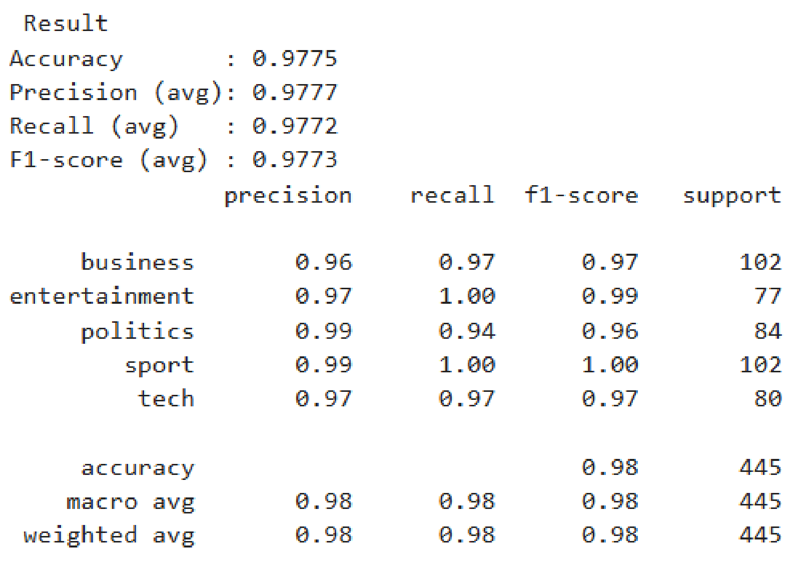
For TFIDF + LogisticRegression

```
 Result
Accuracy        : 0.9663
Precision (avg): 0.9685
Recall (avg)    : 0.9644
F1-score (avg)  : 0.9659
              precision    recall  f1-score   support

     business       0.93      0.98      0.95       102
entertainment       0.97      0.99      0.98        77
     politics       0.99      0.90      0.94        84
        sport       0.98      1.00      0.99       102
         tech       0.97      0.95      0.96        80

     accuracy                           0.97       445
    macro avg       0.97      0.96      0.97       445
 weighted avg       0.97      0.97      0.97       445
```

For TFIDF + MLP (NN) :

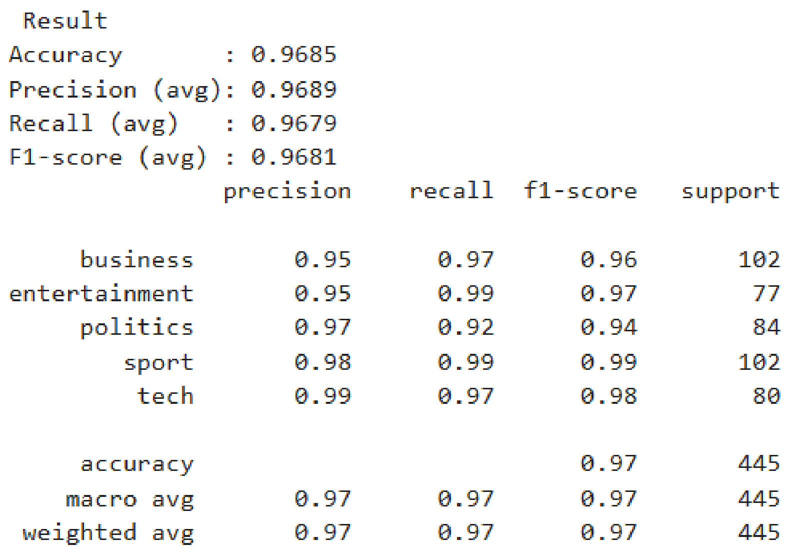


```
 Result
Accuracy        : 0.9775
Precision (avg): 0.9777
Recall (avg)    : 0.9772
F1-score (avg)  : 0.9773
               precision    recall  f1-score   support

     business       0.96      0.97      0.97       102
entertainment       0.97      1.00      0.99        77
     politics       0.99      0.94      0.96        84
        sport       0.99      1.00      1.00       102
         tech       0.97      0.97      0.97        80

     accuracy                           0.98       445
    macro avg       0.98      0.98      0.98       445
 weighted avg       0.98      0.98      0.98       445
```

For TFIDF + Dense model (Keras fully connected NN) :

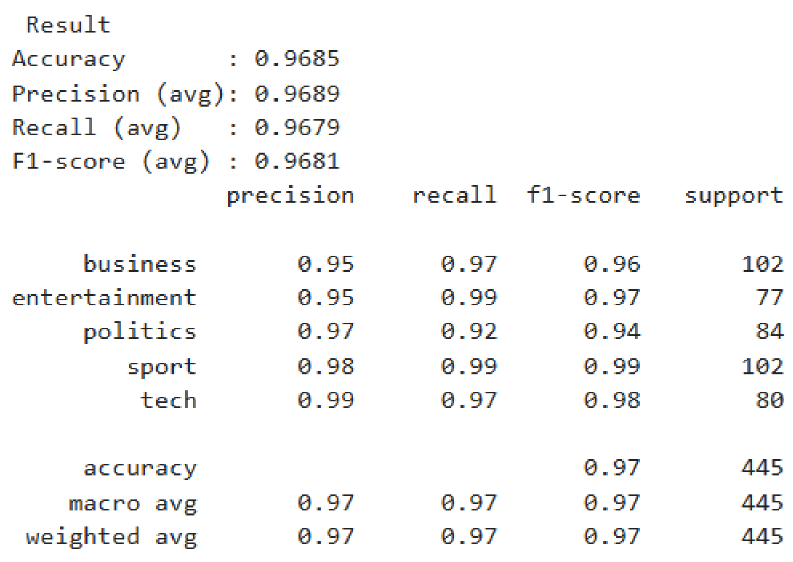


```
 Result
Accuracy        : 0.9685
Precision (avg): 0.9689
Recall (avg)    : 0.9679
F1-score (avg)  : 0.9681
               precision    recall  f1-score   support

     business       0.95      0.97      0.96       102
entertainment       0.95      0.99      0.97        77
     politics       0.97      0.92      0.94        84
        sport       0.98      0.99      0.99       102
         tech       0.99      0.97      0.98        80

     accuracy                           0.97       445
    macro avg       0.97      0.97      0.97       445
 weighted avg       0.97      0.97      0.97       445
```

For CountVectorizer + Dense model (Keras fully connected NN)

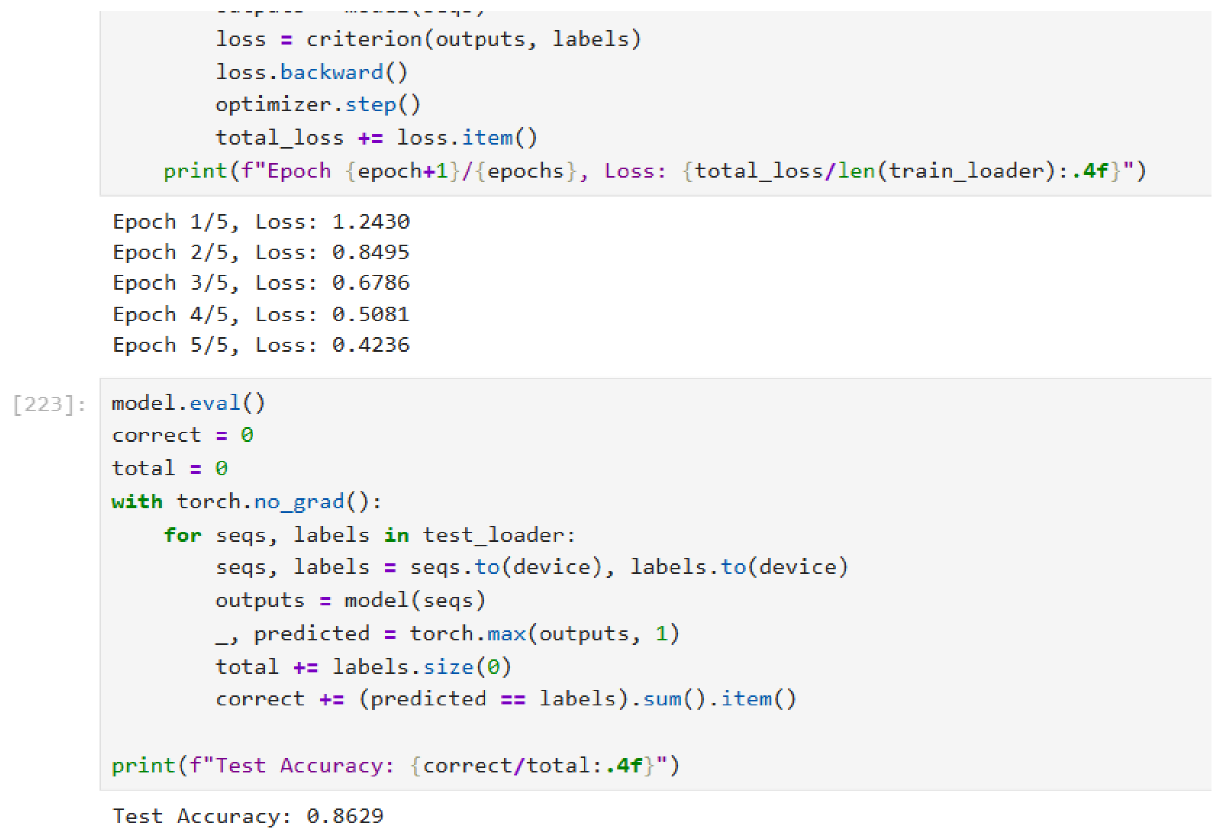


```
 Result
Accuracy        : 0.9685
Precision (avg): 0.9689
Recall (avg)    : 0.9679
F1-score (avg) : 0.9681
               precision    recall  f1-score   support

     business       0.95      0.97      0.96       102
entertainment       0.95      0.99      0.97        77
     politics       0.97      0.92      0.94        84
        sport       0.98      0.99      0.99       102
         tech       0.99      0.97      0.98        80

     accuracy                           0.97       445
    macro avg       0.97      0.97      0.97       445
 weighted avg       0.97      0.97      0.97       445
```

For Word2Vec + BiLSTM :



```
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
    print(f"Epoch {epoch+1}/{epochs}, Loss: {total_loss/len(train_loader):.4f}")
```

```
Epoch 1/5, Loss: 1.2430
Epoch 2/5, Loss: 0.8495
Epoch 3/5, Loss: 0.6786
Epoch 4/5, Loss: 0.5081
Epoch 5/5, Loss: 0.4236
```

```
[223]: model.eval()
correct = 0
total = 0
with torch.no_grad():
    for seqs, labels in test_loader:
        seqs, labels = seqs.to(device), labels.to(device)
        outputs = model(seqs)
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

print(f"Test Accuracy: {correct/total:.4f}")
```

```
Test Accuracy: 0.8629
```

For TFIDF + LinearSVC :

```
 Result
Accuracy        : 0.9596
Precision (avg): 0.9616
Recall (avg)    : 0.9579
F1-score (avg) : 0.9591
               precision    recall  f1-score   support

     business       0.93      0.97      0.95       102
entertainment       0.94      0.97      0.96        77
     politics       0.99      0.90      0.94        84
        sport       0.97      0.99      0.98       102
         tech       0.99      0.95      0.97        80

     accuracy                           0.96       445
    macro avg       0.96      0.96      0.96       445
 weighted avg       0.96      0.96      0.96       445
```

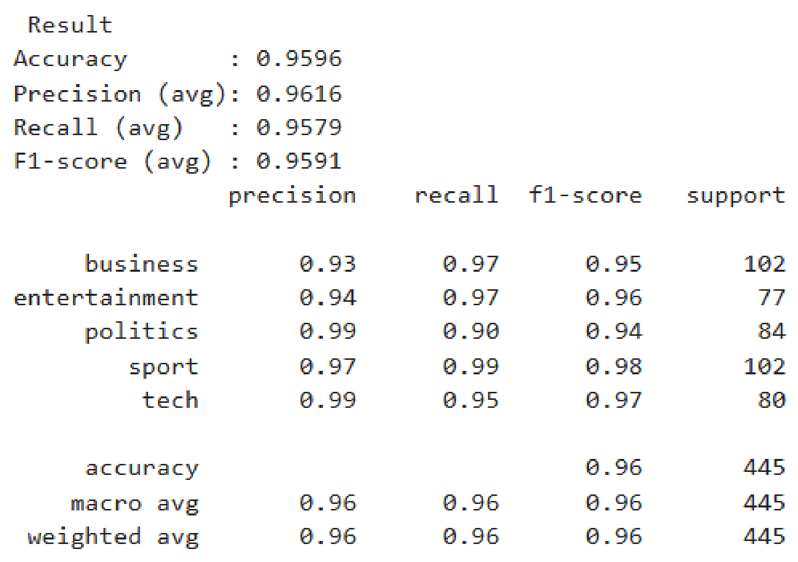
For CountVectorizer + LinearSVC :

```
 Result
Accuracy        : 0.9596
Precision (avg): 0.9616
Recall (avg)    : 0.9579
F1-score (avg) : 0.9591
               precision    recall  f1-score   support

     business       0.93      0.97      0.95       102
entertainment       0.94      0.97      0.96        77
     politics       0.99      0.90      0.94        84
        sport       0.97      0.99      0.98       102
         tech       0.99      0.95      0.97        80

     accuracy                           0.96       445
    macro avg       0.96      0.96      0.96       445
 weighted avg       0.96      0.96      0.96       445
```

For CountVectorizer + MLP (NN) :



```
 Result
Accuracy        : 0.9596
Precision (avg): 0.9616
Recall (avg)    : 0.9579
F1-score (avg) : 0.9591
               precision    recall  f1-score   support

     business       0.93      0.97      0.95       102
entertainment       0.94      0.97      0.96        77
     politics       0.99      0.90      0.94        84
        sport       0.97      0.99      0.98       102
         tech       0.99      0.95      0.97        80

     accuracy                           0.96       445
    macro avg       0.96      0.96      0.96       445
 weighted avg       0.96      0.96      0.96       445
```

The experimental results show that all tested models are capable of effectively classifying BBC news articles into categories such as *business*, *politics*, *sport*, *technology*, and *entertainment*.

Models based on TF-IDF vectorization consistently outperform those using simple CountVectorizer representations. This improvement can be attributed to the fact that TF-IDF not only accounts for word frequency but also considers the relative importance of each word across documents, thus providing a more meaningful and discriminative representation of text data.

Among the classical machine learning models, Logistic Regression and LinearSVC achieved the most reliable and stable results, with high accuracy and F1-scores across all categories. These models were particularly effective in identifying topic-specific keywords — for example, “government”, “minister”, and “election” for *politics*, or “goal”, “match”, and “team” for *sport*.

On the other hand, neural network architectures such as the MLP (Multi-Layer Perceptron) and BiLSTM models also produced strong results, demonstrating their ability to learn complex non-linear relationships. However, they required significantly longer training times and computational resources. The Word2Vec + BiLSTM model, in particular, leveraged semantic word embeddings to better capture contextual relationships within the text, performing well especially for longer and more descriptive articles.