UNIVERSITY OF SUNDERLAND

NATURAL LANGUAGE PROCESSING PIPELINE REPORT

STUDENT NAME: ABDULAZEEZ OLADIPUPO ADIGUN
STUDENT NUMBER: 219143011

Methodology:

Overview:

The task assigned was to design a prediction system to classify the news articles according to the nature of the article in their certain pre-defined categories. The plan is to split the data into training and testing datasets and then produce a pipeline to vectorize and transform the textual data. The data is then to be passed into number of predictive models, classification and regression both, to determine accuracy of each model.

Dataset Used:

The dataset used was 20 Newsgroups dataset. The dataset is divided into two groups from the start i.e., Train and Test and has 20 categories to classify the news articles into. Each split has similar values separated with respect to a certain ratio that is pre-defined. Each split includes information about the writer, subject, date of publishing and the content of the article. The length of train split is 11314 records and test split contain 7532 records.

Pre-processing:

Firstly, the frequency of an article lying in a certain group is calculated separately. Figure 1 shows the class distribution of the news groups/categories and the frequency of articles in each category.

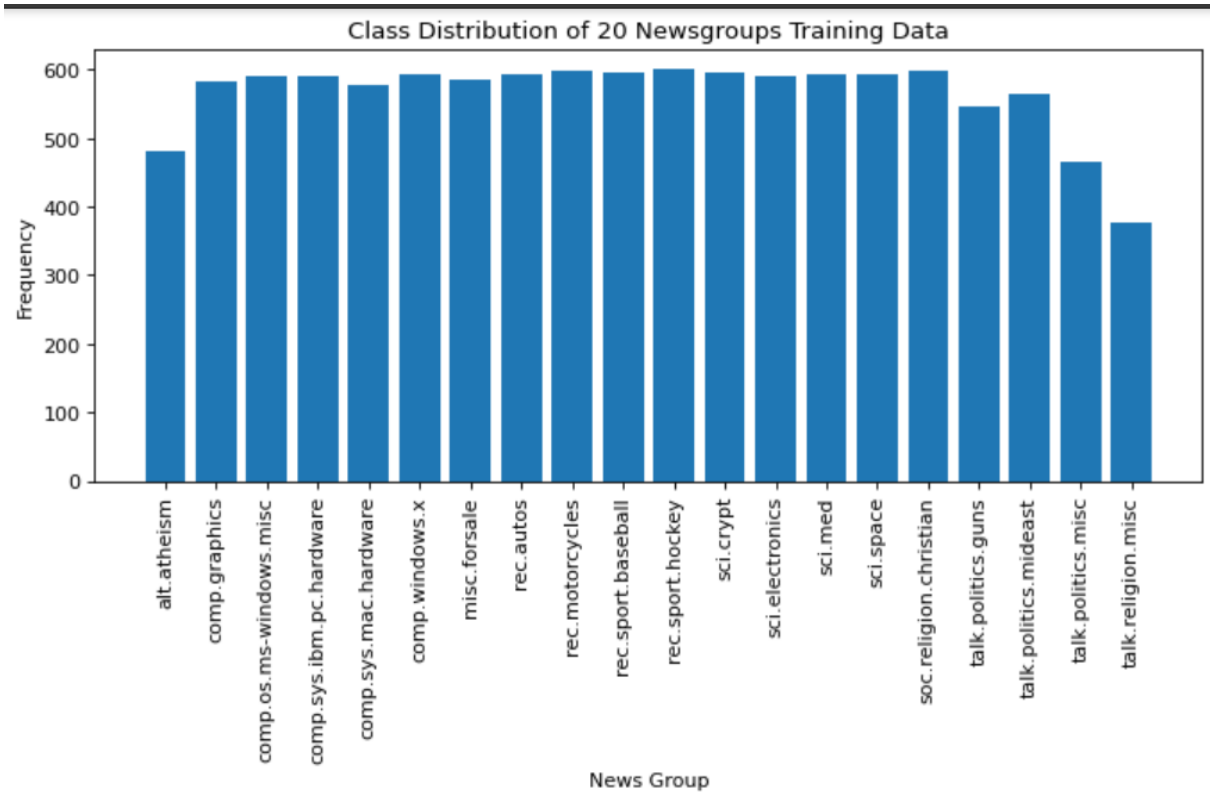

Figure 1

A user-defined function is then written to flatten each data split to convert the data into single dimensional vectors.

```
train_flatten = flatten(train.data)
test_flatten = flatten(test.data)

print(f"{len(train_flatten)} , {len(test_flatten)}")
```

```
22054494 , 13800509
```

Figure 2

Figure 2 shows the flattening of the textual data into the single dimensional vectors.

Results:

Vectorization:

The flattened textual data is then passed to the CountVectorizer in the sklearn library in Python. This converts the textual data into numerical values so it can be used in the classification and regression models.

```
print(X_train_count.data)
print(X_test_count.data)
```

```
[3 2 2 ... 1 1 1]
[1 1 2 ... 1 1 1]
```

Figure 3

Figure 3 shows the vectorized data obtained from both the training and testing data in the form of a python list.

Tf-Idf Transform:

The vectorized data is then passed into the tf-idf transformer to convert/transform a count matrix to a normalized tf-idf representation. Tf-idf means term-frequency times inverse document-frequency. This is a common term weighting scheme in information retrieval, that has also found good use in document classification.

The goal of using tf-idf instead of the raw frequencies of occurrence of a token in a given document is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus.

```
from sklearn.feature_extraction.text import TfidfTransformer
transformer = TfidfTransformer()
X_train_tfidf = transformer.fit_transform(X_train_count)
X_test_tfidf = transformer.transform(X_test_count)
X_test_tfidf.shape
```

```
(7532, 130107)
```

Figure 4

Figure 4 shows the tf-idf implementation on the vectorized data. The transforming model is passed the training vector for training the model. Then both the training and testing vectors are transformed into tf-idf.

Evaluation:

Multinomial Naïve Bayes:

```
from sklearn.metrics import classification_report, accuracy_score

y_test = test.target
print(f"Accuracy of Naive Bayes is {accuracy_score(y_pred_nb,y_test)}")
print(f"\n\n{classification_report(y_pred_nb,y_test)}")
```

```
Accuracy of Naive Bayes is 0.7728359001593202


              precision    recall  f1-score   support

           0       0.77      0.79      0.78       310
           1       0.74      0.67      0.70       430
           2       0.00      0.20      0.01         5
           3       0.77      0.56      0.65       538
           4       0.75      0.84      0.79       345
           5       0.84      0.65      0.73       510
           6       0.65      0.93      0.77       271
           7       0.91      0.87      0.89       415
           8       0.92      0.96      0.94       379
           9       0.87      0.96      0.91       358
          10       0.96      0.93      0.95       415
          11       0.95      0.67      0.78       565
          12       0.66      0.79      0.72       329
          13       0.82      0.87      0.85       372
          14       0.89      0.83      0.86       422
          15       0.96      0.70      0.81       547
          16       0.91      0.69      0.79       475
          17       0.94      0.85      0.89       417
          18       0.63      0.58      0.60       335
          19       0.33      0.89      0.49        94

    accuracy                           0.77      7532
   macro avg       0.76      0.76      0.75      7532
weighted avg       0.84      0.77      0.79      7532
```

Figure 5

Figure 5 shows the accuracy of the **Multinomial Naïve Bayes** Model which was computed to 0.772 or 77.2%. A classification report is also shown for each category stating their precision, recall, F-1 Score and Support.

A Confusion Matrix, shown in Figure 6, is also printed for the model which showcases the predicted and actual values in a beautifully visualized confusion matrix.

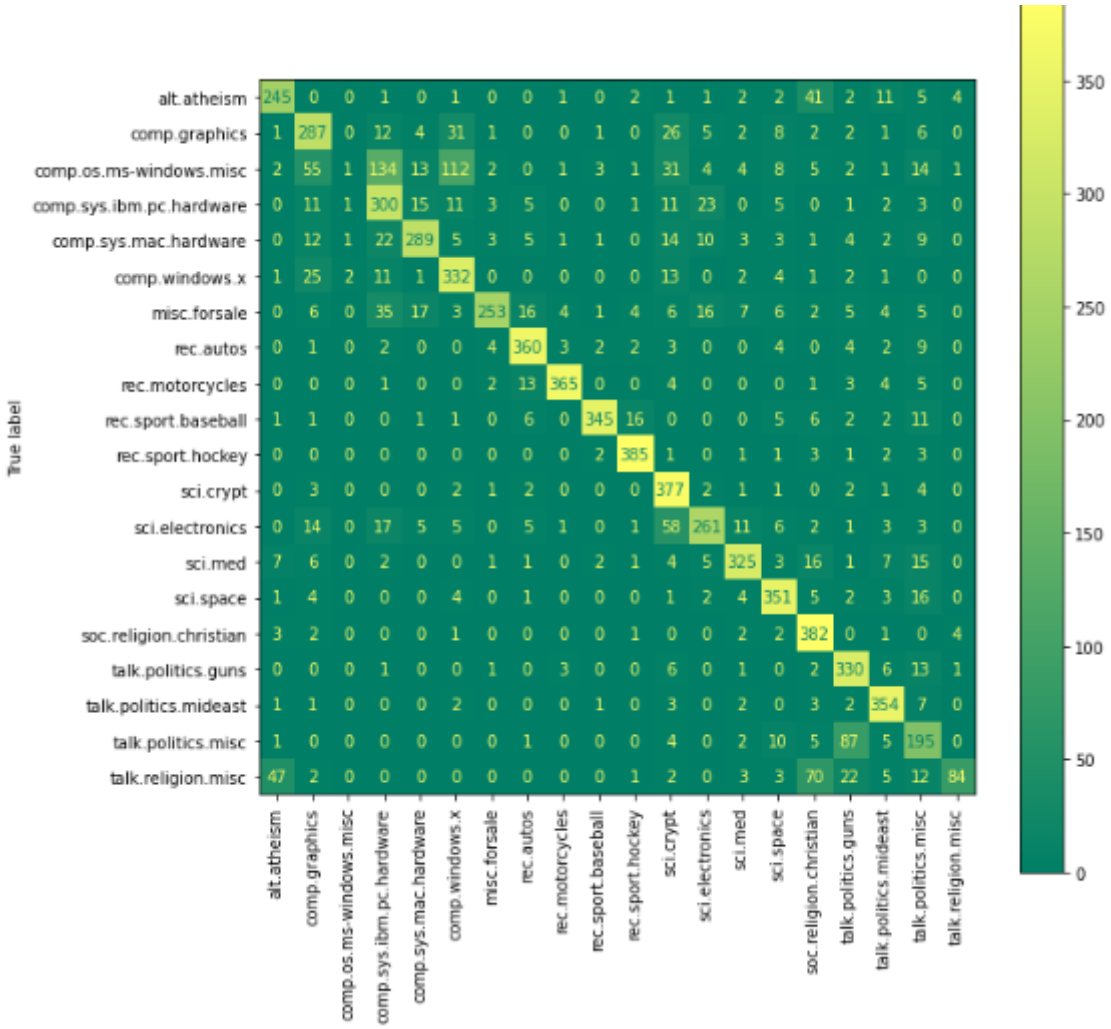

Figure 6

Support Vector Classifier (SVC):

```
print(f"Accuracy of SVC is {accuracy_score(y_pred_svc,y_test)}")
print("\n\n",classification_report(y_test,y_pred_svc))
```

```
Accuracy of SVC is 0.8186404673393521


              precision    recall  f1-score   support

           0       0.83      0.71      0.76       319
           1       0.62      0.82      0.71       389
           2       0.80      0.70      0.75       394
           3       0.73      0.78      0.76       392
           4       0.82      0.83      0.82       385
           5       0.83      0.73      0.77       395
           6       0.73      0.91      0.81       390
           7       0.90      0.87      0.89       396
           8       0.96      0.93      0.95       398
           9       0.88      0.91      0.90       397
          10       0.97      0.91      0.94       399
          11       0.96      0.85      0.90       396
          12       0.65      0.85      0.74       393
          13       0.88      0.78      0.82       396
          14       0.93      0.88      0.90       394
          15       0.79      0.92      0.85       398
          16       0.75      0.88      0.81       364
          17       0.97      0.81      0.88       376
          18       0.75      0.59      0.66       310
          19       0.79      0.55      0.65       251

    accuracy                           0.82      7532
   macro avg       0.83      0.81      0.81      7532
weighted avg       0.83      0.82      0.82      7532
```

Figure 7

Figure 7 shows the accuracy of the **Support Vector Classifier** Model which was computed to 0.818 or 81.8%. A classification report is also shown for each category stating their precision, recall, F-1 Score and Support.

A Confusion Matrix, shown in Figure 8, is also printed for the model which showcases the predicted and actual values in a beautifully visualized confusion matrix.

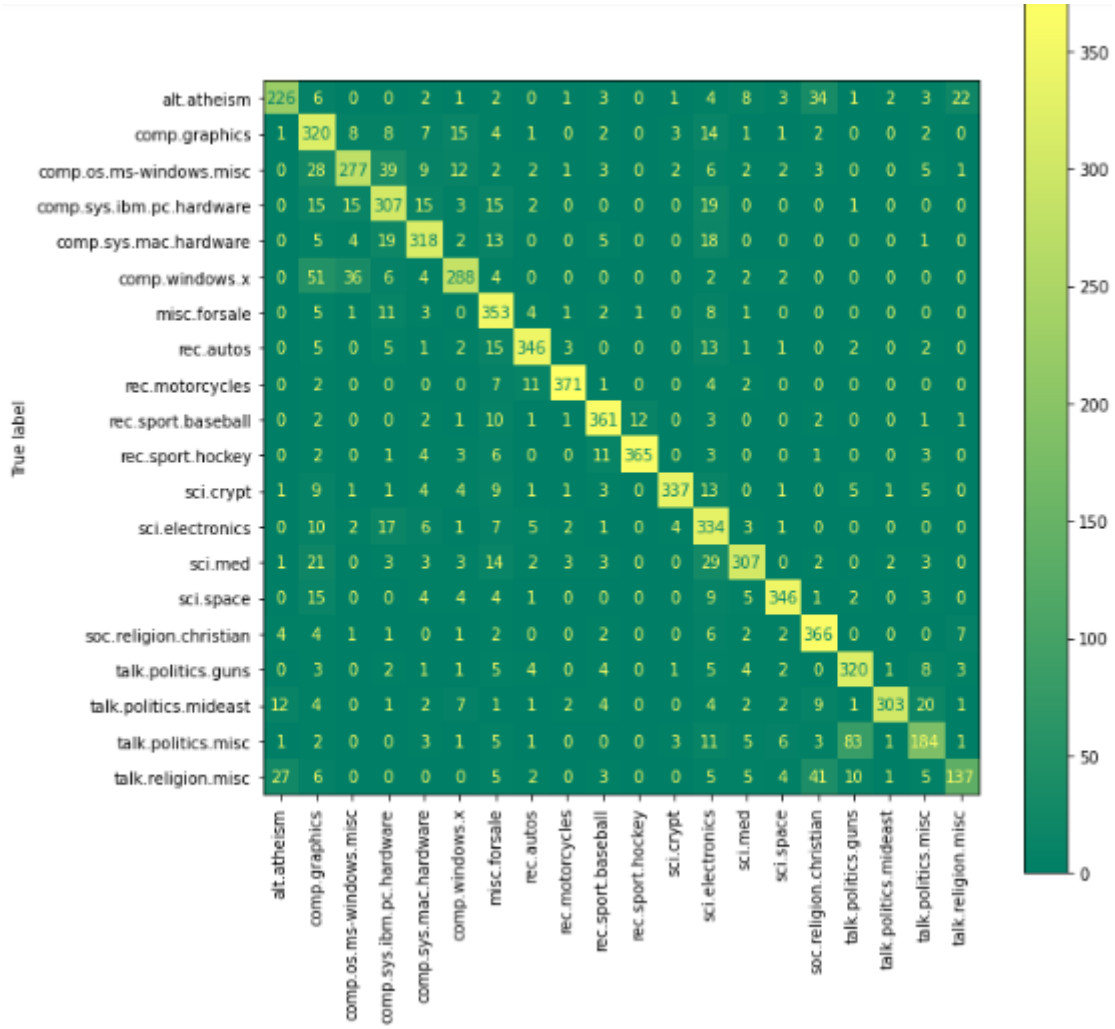
True label
alt.atheism 226 6 0 0 2 1 2 0 1 3 0 1 4 8 3 34 1 2 3 22
comp.graphics 1 320 8 8 7 15 4 1 0 2 0 3 14 1 1 2 0 0 2 0
comp.os.ms-windows.misc 0 28 277 39 9 12 2 2 1 3 0 2 6 2 2 3 0 0 5 1
comp.sys.ibm.pc.hardware 0 15 15 307 15 3 15 2 0 0 0 0 19 0 0 0 1 0 0 0
comp.sys.mac.hardware 0 5 4 19 318 2 13 0 0 5 0 0 18 0 0 0 0 0 1 0
comp.windows.x 0 51 36 6 4 288 4 0 0 0 0 0 2 2 2 0 0 0 0 0
misc.forsale 0 5 1 11 3 0 353 4 1 2 1 0 8 1 0 0 0 0 0 0
rec.autos 0 5 0 5 1 2 15 346 3 0 0 0 13 1 1 0 2 0 2 0
rec.motorcycles 0 2 0 0 0 0 7 11 371 1 0 0 4 2 0 0 0 0 0 0
rec.sport.baseball 0 2 0 0 2 1 10 1 1 361 12 0 3 0 0 2 0 0 1 1
rec.sport.hockey 0 2 0 1 4 3 6 0 0 11 365 0 3 0 0 1 0 0 3 0
sci.crypt 1 9 1 1 4 4 9 1 1 3 0 337 13 0 1 0 5 1 5 0
sci.electronics 0 10 2 17 6 1 7 5 2 1 0 4 334 3 1 0 0 0 0 0
sci.med 1 21 0 3 3 3 14 2 3 3 0 0 29 307 0 2 0 2 3 0
sci.space 0 15 0 0 4 4 4 1 0 0 0 0 9 5 346 1 2 0 3 0
soc.religion.christian 4 4 1 1 0 1 2 0 0 2 0 0 6 2 2 366 0 0 0 7
talk.politics.guns 0 3 0 2 1 1 5 4 0 4 0 1 5 4 2 0 320 1 8 3
talk.politics.mideast 12 4 0 1 2 7 1 1 2 4 0 0 4 2 2 9 1 303 20 1
talk.politics.misc 1 2 0 0 3 1 5 1 0 0 0 3 11 5 6 3 83 1 184 1
talk.religion.misc 27 6 0 0 0 0 5 2 0 3 0 0 5 5 4 41 10 1 5 137
alt.atheism
comp.graphics
comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware
comp.sys.mac.hardware
comp.windows.x
misc.forsale
rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey
sci.crypt
sci.electronics
sci.med
sci.space
soc.religion.christian
talk.politics.guns
talk.politics.mideast
talk.politics.misc
talk.religion.misc
350
300
250
200
150
100
50
0

Figure 8

K-Nearest Neighbors:

```
y_pred_knn = knn.predict(X_test)
print(f"Accuracy of k-NN is {accuracy_score(y_pred_knn,y_test)}")
print("\n\n",classification_report(y_test,y_pred_knn))
```

```
Accuracy of k-NN is 0.6578597981943707


              precision    recall  f1-score   support

           0       0.43      0.75      0.54       319
           1       0.45      0.65      0.53       389
           2       0.52      0.56      0.54       394
           3       0.52      0.62      0.57       392
           4       0.55      0.58      0.56       385
           5       0.69      0.59      0.64       395
           6       0.59      0.48      0.53       390
           7       0.75      0.68      0.71       396
           8       0.83      0.80      0.82       398
           9       0.77      0.75      0.76       397
          10       0.88      0.82      0.85       399
          11       0.74      0.83      0.78       396
          12       0.72      0.47      0.57       393
          13       0.78      0.51      0.62       396
          14       0.81      0.77      0.79       394
          15       0.80      0.72      0.76       398
          16       0.72      0.72      0.72       364
          17       0.70      0.72      0.71       376
          18       0.60      0.56      0.58       310
          19       0.62      0.51      0.56       251

    accuracy                           0.66      7532
   macro avg       0.67      0.65      0.66      7532
weighted avg       0.68      0.66      0.66      7532
```

Figure 9

Figure 9 shows the accuracy of the **k-Nearest Neighbors** Model which was computed to 0.657 or 65.7%. A classification report is also shown for each category stating their precision, recall, F-1 Score and Support.

A Confusion Matrix, shown in Figure 10, is also printed for the model which showcases the predicted and actual values in a beautifully visualized confusion matrix.

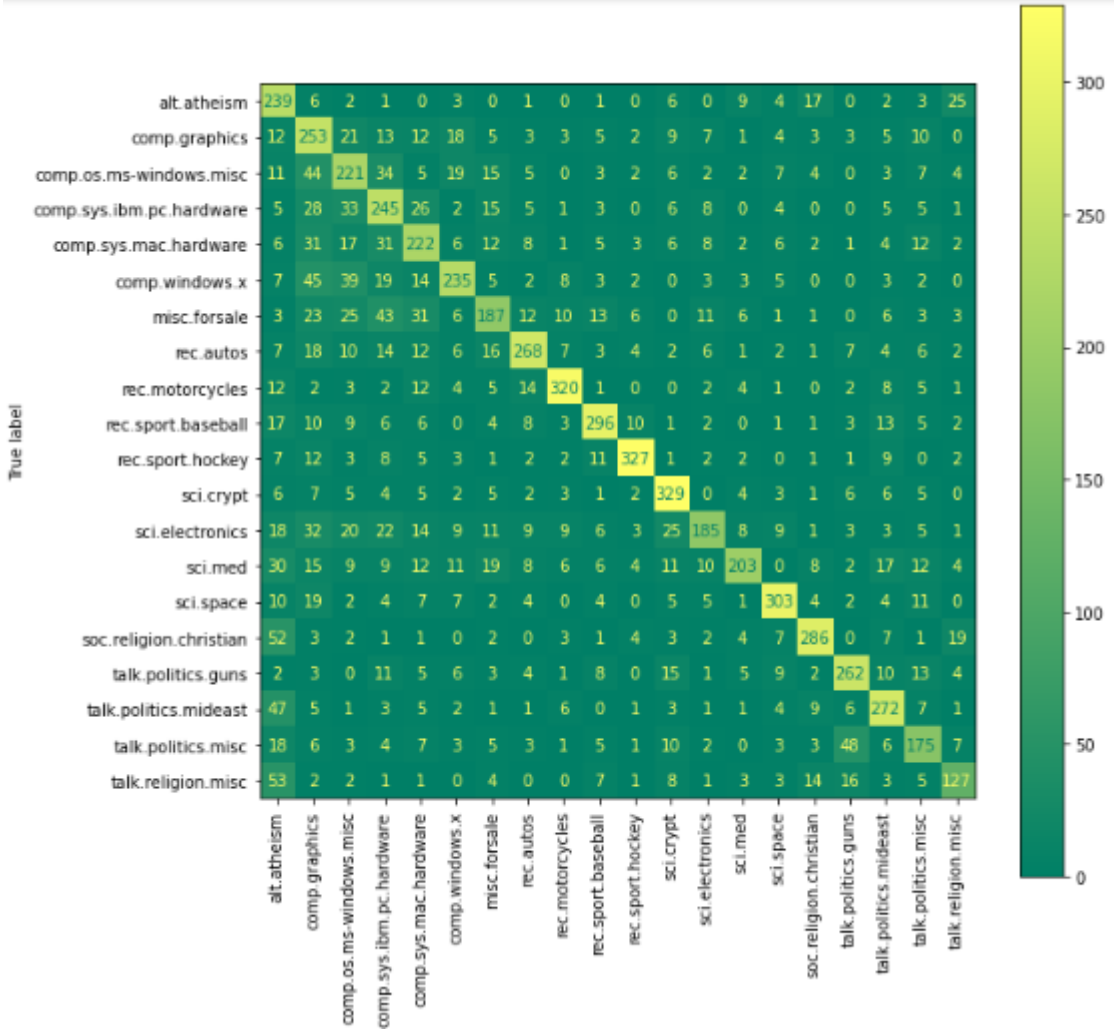

Figure 10

Logistic Regression:

```
print(f"Accuracy of Logistic Regression is {accuracy_score(y_pred_lr,y_test)}")
print("\n\n",classification_report(y_test,y_pred_lr))
```

```
Accuracy of Logistic Regression is 0.8274030801911842


              precision    recall  f1-score   support

           0       0.80      0.74      0.77       319
           1       0.69      0.79      0.74       389
           2       0.75      0.73      0.74       394
           3       0.72      0.72      0.72       392
           4       0.81      0.83      0.82       385
           5       0.83      0.74      0.78       395
           6       0.76      0.90      0.82       390
           7       0.90      0.89      0.90       396
           8       0.95      0.95      0.95       398
           9       0.88      0.92      0.90       397
          10       0.94      0.95      0.95       399
          11       0.94      0.88      0.91       396
          12       0.76      0.80      0.78       393
          13       0.89      0.83      0.85       396
          14       0.91      0.92      0.91       394
          15       0.81      0.94      0.87       398
          16       0.72      0.88      0.79       364
          17       0.96      0.87      0.92       376
          18       0.76      0.59      0.66       310
          19       0.81      0.49      0.61       251

    accuracy                           0.83      7532
   macro avg       0.83      0.82      0.82      7532
weighted avg       0.83      0.83      0.83      7532
```

Figure 11

Figure 9 shows the accuracy of the **Logistic Regression** Model which was computed to 0.827 or 82.7%. A classification report is also shown for each category stating their precision, recall, F-1 Score and Support.

A Confusion Matrix, shown in Figure 12, is also printed for the model which showcases the predicted and actual values in a beautifully visualized confusion matrix.

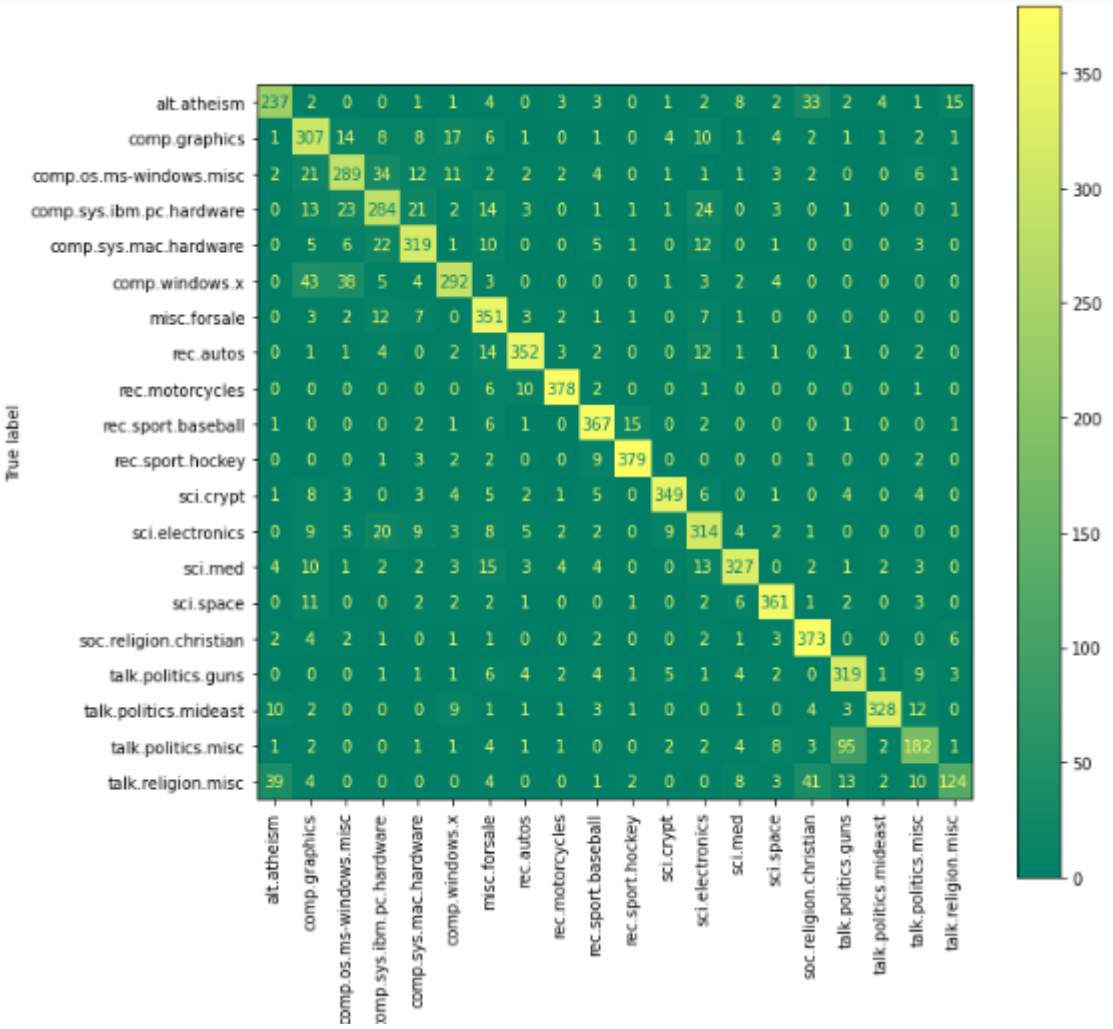

Figure 12

Conclusion:

The accuracies of all the models are:

Models	Accuracy
Multinomial Naïve Bayes	77.2%
Support Vector Classifier	81.8%
k-Nearest Neighbors	**65.7%**
Logistic Regression	**82.7%**

So, we conclude that the "good" model for this data is **Logistic Regression** and the "bad" model for this data is **k-Nearest Neighbors**.