TEXT CLASSIFICATION MODEL

BY:

MANSI JAIN

APPROACH

- The dataset had two columns containing Text and Target. The text was related to the target classes like blockchain, bigdata etc.
- From this unstructured text data we needed to derive some meaningful information that must be related to the given target classes.
- For this purpose I followed Natural Language Processing technique using language Python. The following steps were taken to sove the problem:
  - Loading the dataset: using python libraries and dataset location
  - Text preprocessing: to eliminate the non or less meaningful text
  - Conversion of text into numbers: making the data understandable for computer.
  - Splitting : to split the data into training and test datasets.
  - Training the classification model
  - Evaluation of the model: this step followed by running the test dataset onto the classification model.

Model interpretation

- Importing the data

```
In [14]: data=pd.read_csv("C:/Users/abhin/Desktop/root2ai.csv")

In [15]: print(data.head())

                                              Text      Target
0  reserve bank forming expert committee based in...  Blockchain
1          director could play role financial system  Blockchain
2  preliminary discuss secure transaction study r...  Blockchain
3  security indeed prove essential transforming f...  Blockchain
4  bank settlement normally take three days based...  Blockchain

In [16]: X, y = data.Text,data.Target
```

- I used Regex Expressions from Python re library to perform text preprocessing tasks for removing all non-word characters such as special characters, numbers, etc. and then spaces, single characters etc.
- Finally lemmatization and stemming was performed to reduce the word into dictionary root form.

```
In [13]: #data preprocessing
         documents = []

         from nltk.stem import WordNetLemmatizer

         stemmer = WordNetLemmatizer()

         for sen in range(0, len(X)):
             # Remove all the special characters
             document = re.sub(r'\W', ' ', str(X[sen]))

             # remove all single characters
             document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)

             # Remove single characters from the start
             document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)

             # Substituting multiple spaces with single space
             document = re.sub(r'\s+', ' ', document, flags=re.I)

             # Removing prefixed 'b'
             document = re.sub(r'^b\s+', '', document)

             # Converting to Lowercase
             document = document.lower()

             # Lemmatization
             document = document.split()

             document = [stemmer.lemmatize(word) for word in document]
             document = ' '.join(document)

             documents.append(document)
```

- In order to make the computer understand this raw text it should be converted into corresponding numerical form. In this classification model I have used Bag Of Words +TFIDF model. In the max_features parameter was set to 1500 so that we have 1500 most occurring words as features for training our classifier.

```
In [18]: #converting text into numbers
         from sklearn.feature_extraction.text import CountVectorizer
         vectorizer = CountVectorizer(max_features=1500, min_df=5, max_df=0.7, stop_words=stopwords.words('english'))
         X = vectorizer.fit_transform(documents).toarray()

         from sklearn.feature_extraction.text import TfidfTransformer
         tfidfconverter = TfidfTransformer()
         X = tfidfconverter.fit_transform(X).toarray()
```

- In order to train this dataset I have used RandomForestClassifier imported from(Scikit-Learn) sklearn ensemble.
- Random forest algorithm creates decision trees on data samples and then gets prediction from each of them and finally selects the solution by means of voting. It also reduces the over-fitting by averaging the result.
  - Random forest algorithm uses two key concepts:
    - Random sampling of training data points when building trees.
    - Random subsets of features considered when splitting nodes.

```
In [21]: #Training Text Classification Model and Predicting Sentiment
         from sklearn.ensemble import RandomForestClassifier
         classifier = RandomForestClassifier(n_estimators=1000, random_state=0)
         classifier.fit(X_train, y_train)

Out[21]: RandomForestClassifier(n_estimators=1000, random_state=0)

In [22]: y_pred = classifier.predict(X_test)
```

P.S.: I particularly used this model as it works well with the large datasets more efficiently.

- To evaluate the trained model I have used confusion matrix and F1 measure.

```
In [23]: #evaluating the model
         from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
         print(confusion_matrix(y_test,y_pred))
         print(classification_report(y_test,y_pred))
         print(accuracy_score(y_test, y_pred))
```

Accuracy Score

```
[[ 332    3    8    1   74    4    1   10    4    2   12]
 [  12   81   15    1  154    6    0    5    1    3    8]
 [  33   10  223    4  234    6    4   12    1    4   10]
 [  10    0   16    5   21    0    1    3    0    0    2]
 [  81   15   39    2 1456   12   17   20   13   18   38]
 [  18    6    6    0   93   71    0    0    1    0    1]
 [   8    1    7    0  123    4   57    4    0    1    6]
 [  63    2   13    0   71    2    0  286    2    1    2]
 [   5    0    1    0   82    1    2    0   45    0    2]
 [   7    1    3    0   84    2    0    0    0   77    1]
 [  21    2    6    2  105    1    1    3    1    2  188]]
                  precision    recall  f1-score   support

        Bigdata       0.56      0.74      0.64       451
     Blockchain       0.67      0.28      0.40       286
 Cyber Security       0.66      0.41      0.51       541
  Data Security       0.33      0.09      0.14        58
        FinTech       0.58      0.85      0.69      1711
  Microservices       0.65      0.36      0.47       196
       Neobanks       0.69      0.27      0.39       211
       Reg Tech       0.83      0.65      0.73       442
  Robo Advising       0.66      0.33      0.44       138
  Stock Trading       0.71      0.44      0.54       175
credit reporting       0.70      0.57      0.62       332

       accuracy                           0.62      4541
      macro avg       0.64      0.45      0.51      4541
   weighted avg       0.64      0.62      0.60      4541

0.6212288042281435
```

From the output it can be clearly seen that the Accuracy score is 62.12%. I randomally chose all the parameters for countvectorizer and RandomForest.

Limitations

- The main limitation of this model is that it took a long time to train might be due to big dataset.
- Moreover, random forest algorithm takes a large number of trees that can make the algorithm too slow and ineffective for real-time predictions. In general, these algorithms are fast to train, but quite slow to create predictions once they are trained. A more accurate prediction requires more trees, which results in a slower model.
- It is a predictive modelling tool and cannot provide the relationships in our data.

THANK YOU