```python
import numpy as np
import pandas as pd
import nltk
import matplotlib.pyplot as plt
import seaborn as sbn
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import re
from sklearn.feature_extraction.text import TfidfVectorizer
nltk.download('stopwords')
%matplotlib inline
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```python
## load dataset
df = pd.read_csv("bbc-text.csv")
df.head(10)
```

	category	text
0	tech	tv future in the hands of viewers with home th...
1	business	worldcom boss left books alone former worldc...
2	sport	tigers wary of farrell gamble leicester say ...
3	sport	yeading face newcastle in fa cup premiership s...
4	entertainment	ocean s twelve raids box office ocean s twelve...
5	politics	howard hits back at mongrel jibe michael howar...
6	politics	blair prepares to name poll date tony blair is...
7	sport	henman hopes ended in dubai third seed tim hen...
8	sport	wilkinson fit to face edinburgh england captai...
9	entertainment	last star wars not for children the sixth an...

```python
df.shape
```

```
(2225, 2)
```

```python
df['text'][0]
```

```
'tv future in the hands of viewers with home theatre systems  plasma high-definition tvs  and digital video recorders moving into th
```

```python
df['category'].unique()
```

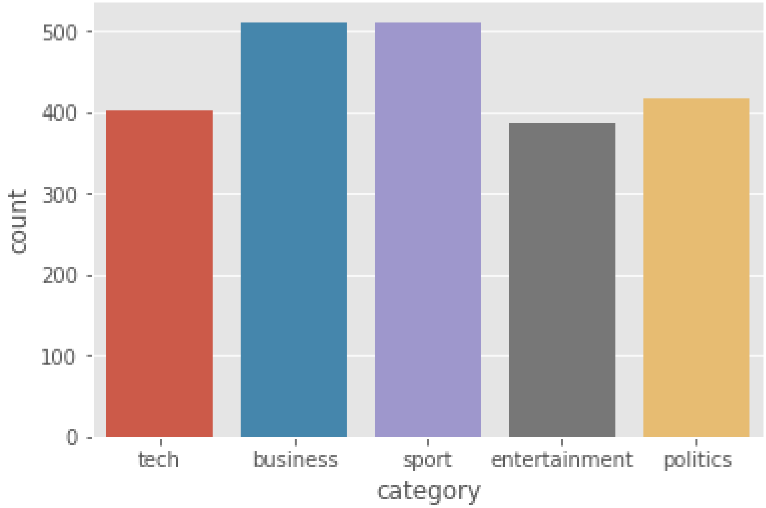
```
array(['tech', 'business', 'sport', 'entertainment', 'politics'],
      dtype=object)
```

```python
df['category'].value_counts()
```

```
sport            511
business         510
politics         417
tech             401
entertainment    386
Name: category, dtype: int64
```

```python
sbn.countplot(df['category'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f449c30dc50>
```




```python
# Use sklearn utility to convert label strings to numbered index
from sklearn.preprocessing import LabelEncoder
df["category"] = LabelEncoder().fit_transform(df["category"])
df.head()
```

	category	text
0	4	tv future in the hands of viewers with home th...
1	0	worldcom boss left books alone former worldc...
2	3	tigers wary of farrell gamble leicester say ...
3	3	yeading face newcastle in fa cup premiership s...
4	1	ocean s twelve raids box office ocean s twelve...

```
#tokenize the words (text)
stemmer = PorterStemmer()
words = stopwords.words("english")
df['text'] = df['text'].apply(lambda x: " ".join([stemmer.stem(i) for i in re.sub("[^a-zA-Z]", " ", x).split() if i not in words]).lower())
vectorizer = TfidfVectorizer(min_df= 3, stop_words="english", sublinear_tf=True, norm='l2', ngram_range=(1, 2))
final_features = vectorizer.fit_transform(df['text']).toarray()
df.head()
```

	category	text
0	4	tv futur hand viewer home theatr system plasma...
1	0	worldcom boss left book alon former worldcom b...
2	3	tiger wari farrel gambl leicest say rush make ...
3	3	yead face newcastl fa cup premiership side new...
4	1	ocean twelv raid box offic ocean twelv crime c...

```
#split the data into training and test sets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df['text'], df['category'], test_size=0.30, random_state=0)
# Inspect the dimensions of our training and test data
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1557,)
(668,)
(1557,)
(668,)
```

```
#convert to a vector with 1000 words
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_features = 1000)
vectorizer.fit(X_train)

X_train = vectorizer.transform(X_train)
X_test  = vectorizer.transform(X_test)
X_train = X_train.todense()
```

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression()
classifier.fit(X_train, y_train)
score = classifier.score(X_test, y_test)
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/logistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs
  FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/logistic.py:469: FutureWarning: Default multi_class will be changed to '
  "this warning.", FutureWarning)
```

```
print("Accuracy of Logistic Regression:", score)
```

```
Accuracy of Logistic Regression: 0.9745508982035929
```

```
# Converts the labels to a one-hot representation
import keras
num_classes = np.max(y_train) + 1
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
# Build the model
from tensorflow import keras
layers = keras.layers
models = keras.models
from keras.models import Sequential
from keras.layers import Dense
model = Sequential()

model.add(Dense(20, input_dim= 1000,activation='relu'))
model.add(Dense(20, activation='relu'))
model.add(Dense(num_classes , activation='softmax'))
```

```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_46 (Dense)             (None, 20)                20020
_________________________________________________________________
dense_47 (Dense)             (None, 20)                420
_________________________________________________________________
dense_48 (Dense)             (None, 5)                 105
=================================================================
Total params: 20,545
Trainable params: 20,545
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(X_train, y_train,
                    epochs=50,
                    validation_split=0.10,
                     batch_size=4)
```

```
Train on 1401 samples, validate on 156 samples
Epoch 1/50
1401/1401 [==============================] - 2s 2ms/step - loss: 0.6100 - acc: 0.8144 - val_loss: 0.3113 - val_acc: 0.8974
Epoch 2/50
1401/1401 [==============================] - 1s 901us/step - loss: 0.0733 - acc: 0.9843 - val_loss: 0.1114 - val_acc: 0.9808
Epoch 3/50
1401/1401 [==============================] - 1s 899us/step - loss: 0.0187 - acc: 0.9979 - val_loss: 0.0808 - val_acc: 0.9808
Epoch 4/50
1401/1401 [==============================] - 1s 884us/step - loss: 0.0068 - acc: 1.0000 - val_loss: 0.0778 - val_acc: 0.9744
Epoch 5/50
1401/1401 [==============================] - 1s 929us/step - loss: 0.0039 - acc: 1.0000 - val_loss: 0.0814 - val_acc: 0.9744
Epoch 6/50
1401/1401 [==============================] - 1s 905us/step - loss: 0.0024 - acc: 1.0000 - val_loss: 0.0764 - val_acc: 0.9744
Epoch 7/50
1401/1401 [==============================] - 1s 910us/step - loss: 0.0016 - acc: 1.0000 - val_loss: 0.0794 - val_acc: 0.9744
Epoch 8/50
1401/1401 [==============================] - 1s 868us/step - loss: 0.0011 - acc: 1.0000 - val_loss: 0.0784 - val_acc: 0.9744
Epoch 9/50
1401/1401 [==============================] - 1s 873us/step - loss: 7.8040e-04 - acc: 1.0000 - val_loss: 0.0783 - val_acc: 0.9744
Epoch 10/50
1401/1401 [==============================] - 1s 881us/step - loss: 5.6683e-04 - acc: 1.0000 - val_loss: 0.0801 - val_acc: 0.9744
Epoch 11/50
1401/1401 [==============================] - 1s 880us/step - loss: 4.1964e-04 - acc: 1.0000 - val_loss: 0.0816 - val_acc: 0.9744
Epoch 12/50
1401/1401 [==============================] - 1s 875us/step - loss: 3.1483e-04 - acc: 1.0000 - val_loss: 0.0842 - val_acc: 0.9744
Epoch 13/50
1401/1401 [==============================] - 1s 878us/step - loss: 2.4039e-04 - acc: 1.0000 - val_loss: 0.0823 - val_acc: 0.9744
Epoch 14/50
1401/1401 [==============================] - 1s 904us/step - loss: 1.8510e-04 - acc: 1.0000 - val_loss: 0.0858 - val_acc: 0.9744
Epoch 15/50
1401/1401 [==============================] - 1s 930us/step - loss: 1.4411e-04 - acc: 1.0000 - val_loss: 0.0892 - val_acc: 0.9744
Epoch 16/50
1401/1401 [==============================] - 1s 885us/step - loss: 1.1325e-04 - acc: 1.0000 - val_loss: 0.0900 - val_acc: 0.9744
Epoch 17/50
1401/1401 [==============================] - 1s 901us/step - loss: 8.9144e-05 - acc: 1.0000 - val_loss: 0.0916 - val_acc: 0.9744
Epoch 18/50
1401/1401 [==============================] - 1s 901us/step - loss: 7.0803e-05 - acc: 1.0000 - val_loss: 0.0950 - val_acc: 0.9744
Epoch 19/50
1401/1401 [==============================] - 1s 923us/step - loss: 5.6671e-05 - acc: 1.0000 - val_loss: 0.0964 - val_acc: 0.9744
Epoch 20/50
1401/1401 [==============================] - 1s 880us/step - loss: 4.5506e-05 - acc: 1.0000 - val_loss: 0.0963 - val_acc: 0.9744
Epoch 21/50
1401/1401 [==============================] - 1s 879us/step - loss: 3.6594e-05 - acc: 1.0000 - val_loss: 0.0998 - val_acc: 0.9744
Epoch 22/50
1401/1401 [==============================] - 1s 910us/step - loss: 2.9441e-05 - acc: 1.0000 - val_loss: 0.1000 - val_acc: 0.9744
Epoch 23/50
```

```
1401/1401 [==============================] - 1s 874us/step - loss: 2.3763e-05 - acc: 1.0000 - val_loss: 0.1005 - val_acc: 0.9744
Epoch 24/50
1401/1401 [==============================] - 1s 906us/step - loss: 1.9280e-05 - acc: 1.0000 - val_loss: 0.1030 - val_acc: 0.9744
Epoch 25/50
1401/1401 [==============================] - 1s 881us/step - loss: 1.5571e-05 - acc: 1.0000 - val_loss: 0.1041 - val_acc: 0.9744
Epoch 26/50
1401/1401 [==============================] - 1s 868us/step - loss: 1.2621e-05 - acc: 1.0000 - val_loss: 0.1061 - val_acc: 0.9744
Epoch 27/50
1401/1401 [==============================] - 1s 872us/step - loss: 1.0231e-05 - acc: 1.0000 - val_loss: 0.1099 - val_acc: 0.9744
Epoch 28/50
1401/1401 [==============================] - 1s 883us/step - loss: 8.3612e-06 - acc: 1.0000 - val_loss: 0.1122 - val_acc: 0.9744
Epoch 29/50
1401/1401 [==============================] - 1s 873us/step - loss: 6.7757e-06 - acc: 1.0000 - val_loss: 0.1125 - val_acc: 0.9744
Epoch 30/50
1401/1401 [==============================] - 1s 887us/step - loss: 5.5421e-06 - acc: 1.0000 - val_loss: 0.1159 - val_acc: 0.9744
Epoch 31/50
1401/1401 [==============================] - 1s 890us/step - loss: 4.5160e-06 - acc: 1.0000 - val_loss: 0.1159 - val_acc: 0.9744
Epoch 32/50
1401/1401 [==============================] - 1s 885us/step - loss: 3.6901e-06 - acc: 1.0000 - val_loss: 0.1183 - val_acc: 0.9744
Epoch 33/50
1401/1401 [==============================] - 1s 895us/step - loss: 3.0240e-06 - acc: 1.0000 - val_loss: 0.1203 - val_acc: 0.9744
Epoch 34/50
1401/1401 [==============================] - 1s 896us/step - loss: 2.4854e-06 - acc: 1.0000 - val_loss: 0.1217 - val_acc: 0.9744
Epoch 35/50
1401/1401 [==============================] - 1s 882us/step - loss: 2.0442e-06 - acc: 1.0000 - val_loss: 0.1213 - val_acc: 0.9744
Epoch 36/50
1401/1401 [==============================] - 1s 864us/step - loss: 1.6950e-06 - acc: 1.0000 - val_loss: 0.1263 - val_acc: 0.9744
Epoch 37/50
1401/1401 [==============================] - 1s 869us/step - loss: 1.4025e-06 - acc: 1.0000 - val_loss: 0.1270 - val_acc: 0.9744
Epoch 38/50
1401/1401 [==============================] - 1s 877us/step - loss: 1.1615e-06 - acc: 1.0000 - val_loss: 0.1271 - val_acc: 0.9744
Epoch 39/50
1401/1401 [==============================] - 1s 883us/step - loss: 9.6589e-07 - acc: 1.0000 - val_loss: 0.1272 - val_acc: 0.9744
Epoch 40/50
1401/1401 [==============================] - 1s 865us/step - loss: 8.0669e-07 - acc: 1.0000 - val_loss: 0.1300 - val_acc: 0.9744
Epoch 41/50
1401/1401 [==============================] - 1s 845us/step - loss: 6.7676e-07 - acc: 1.0000 - val_loss: 0.1298 - val_acc: 0.9744
Epoch 42/50
1401/1401 [==============================] - 1s 848us/step - loss: 5.7363e-07 - acc: 1.0000 - val_loss: 0.1328 - val_acc: 0.9744
Epoch 43/50
1401/1401 [==============================] - 1s 846us/step - loss: 4.8343e-07 - acc: 1.0000 - val_loss: 0.1351 - val_acc: 0.9744
Epoch 44/50
1401/1401 [==============================] - 1s 871us/step - loss: 4.1209e-07 - acc: 1.0000 - val_loss: 0.1389 - val_acc: 0.9744
Epoch 45/50
1401/1401 [==============================] - 1s 854us/step - loss: 3.5461e-07 - acc: 1.0000 - val_loss: 0.1380 - val_acc: 0.9744
Epoch 46/50
1401/1401 [==============================] - 1s 840us/step - loss: 3.0675e-07 - acc: 1.0000 - val_loss: 0.1410 - val_acc: 0.9744
```

```
Epoch 47/50
1401/1401 [==============================] - 1s 846us/step - loss: 2.6828e-07 - acc: 1.0000 - val_loss: 0.1420 - val_acc: 0.9744
Epoch 48/50
1401/1401 [==============================] - 1s 877us/step - loss: 2.3718e-07 - acc: 1.0000 - val_loss: 0.1445 - val_acc: 0.9744
Epoch 49/50
1401/1401 [==============================] - 1s 853us/step - loss: 2.1174e-07 - acc: 1.0000 - val_loss: 0.1442 - val_acc: 0.9744
Epoch 50/50
1401/1401 [==============================] - 1s 862us/step - loss: 1.9043e-07 - acc: 1.0000 - val_loss: 0.1465 - val_acc: 0.9744
```

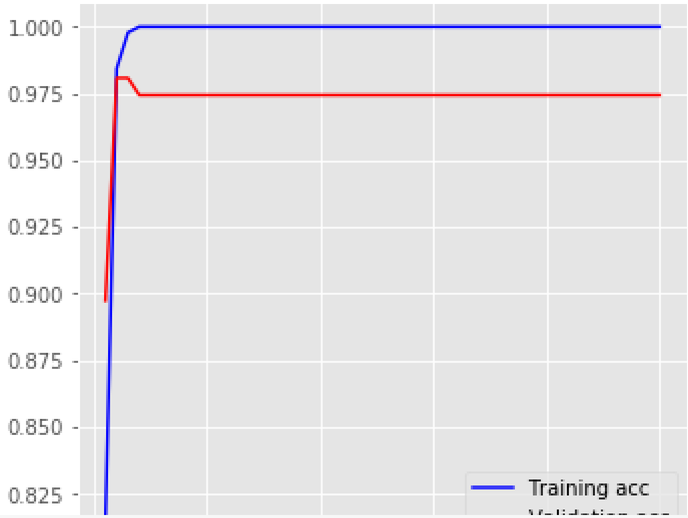
```python
loss, accuracy = model.evaluate(X_train, y_train, verbose=False)
print("Training Accuracy: {:.4f}".format(accuracy))
loss, accuracy = model.evaluate(X_test, y_test, verbose=False)
print("Testing Accuracy:  {:.4f}".format(accuracy))
```

```
Training Accuracy: 0.9974
Testing Accuracy:  0.9775
```

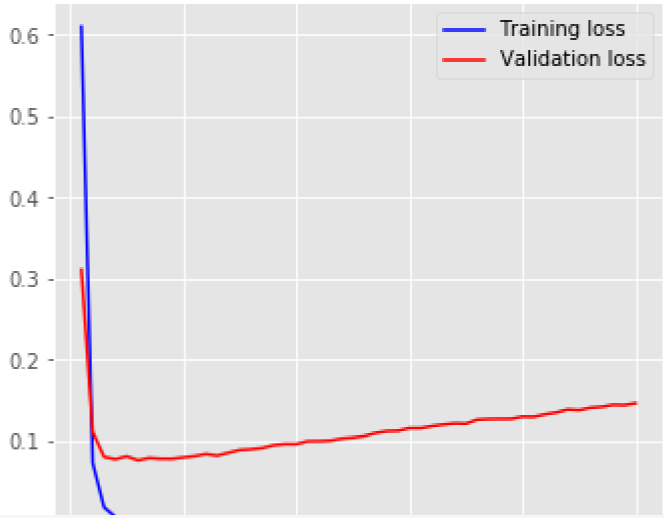
```python
import matplotlib.pyplot as plt
plt.style.use('ggplot')

def plot_history(history):
    acc = history.history['acc']
    val_acc = history.history['val_acc']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    x = range(1, len(acc) + 1)

    plt.figure(figsize=(12, 5))
    plt.subplot(1, 2, 1)
    plt.plot(x, acc, 'b', label='Training acc')
    plt.plot(x, val_acc, 'r', label='Validation acc')
    plt.title('Training and validation accuracy')
    plt.legend()
    plt.subplot(1, 2, 2)
    plt.plot(x, loss, 'b', label='Training loss')
    plt.plot(x, val_loss, 'r', label='Validation loss')
    plt.title('Training and validation loss')
    plt.legend()
```

```python
plot_history(history)
```

Training and validation accuracy



Training and validation loss




```
y_softmax = model.predict(X_test)

y_test_1d = []
y_pred_1d = []

for i in range(len(y_test)):
    probs = y_test[i]
    index_arr = np.nonzero(probs)
    one_hot_index = index_arr[0].item(0)
    y_test_1d.append(one_hot_index)

for i in range(0, len(y_softmax)):
    probs = y_softmax[i]
    predicted_index = np.argmax(probs)
    y_pred_1d.append(predicted_index)
```

```
from sklearn import metrics

print(metrics.confusion_matrix(y_test_1d, y_pred_1d))

print(metrics.classification_report(y_test_1d, y_pred_1d))

from sklearn.metrics import accuracy_score

print("Accuracy of Deep Model is:",accuracy_score(y_test_1d, y_pred_1d))
```