```
import pandas as pd
import numpy as np
data=pd.read_csv('/fakenews.csv')
data
```

	text	label
0	Get the latest from TODAY Sign up for our news...	1
1	2d Conan On The Funeral Trump Will Be Invited...	1
2	It's safe to say that Instagram Stories has fa...	0
3	Much like a certain Amazon goddess with a lass...	0
4	At a time when the perfect outfit is just one ...	0
...	...	...
4981	The storybook romance of WWE stars John Cena a...	0
4982	The actor told friends he's responsible for en...	0
4983	Sarah Hyland is getting real. The Modern Fami...	0
4984	Production has been suspended on the sixth and...	0
4985	A jury ruled against Bill Cosby in his sexual ...	0

4986 rows × 2 columns

Next steps: Generate code with data View recommended plots

```
# pre processing function
import re
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer

nltk.download('punkt')
nltk.download('wordnet')


def preprocess_text(text):
    # Convert text to lowercase
    text = text.lower()

    # Remove numbers
    text = re.sub(r'\d+', '', text)

    # Remove punctuation
    text = re.sub(r'[^\w\s]', '', text)

    # Tokenize text
    tokens = word_tokenize(text)

    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    filtered_tokens = [word for word in tokens if word not in stop_words]

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    lemmatized_tokens = [lemmatizer.lemmatize(word) for word in filtered_tokens]

    # Join tokens back into a string
    preprocessed_text = ' '.join(lemmatized_tokens)

    return preprocessed_text
```

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

```
data['text']=data.text.apply(preprocess_text)
```

```
data.head()
```

	text	label
0	get latest today sign newsletter one ever trul...	1
1	conan funeral trump invited conan tb	1
2	safe say instagram story far surpassed competi...	0
3	much like certain amazon goddess lasso height ...	0
4	time perfect outfit one click away high demand...	0

Next steps: Generate code with data View recommended plots

Train test split

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(data.text,data.label,test_size=0.2,random_state=0)
```

```
print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
```

```
(3988,) (3988,)
(998,) (998,)
```

Import google word2vec model

```
import gensim
```

```
from gensim.models import Word2Vec, KeyedVectors
```

```
# importing google word2vec model
import gensim.downloader as api
wv=api.load("word2vec-google-news-300")
```

```
[==================================================] 100.0% 1662.8/1662.8MB downloaded
```

- word2vec model will give 300 vectors for each word.
- if word not in corpus (words list) add 300 zeros.
- finally (horizontally) average of all words in a sentance is known as vector of that sentances.

```
def vect(text):
  words=text.split()
  word_vec=[wv[word] if word in wv else np.zeros(300) for word in words]
  word_vec=np.array(word_vec).mean(axis=0)
  return word_vec
```

```
X_train_vec=np.array([vect(text) for text in X_train])
X_test_vec=np.array([vect(text) for text in X_test])
```

```
X_train_vec.shape
```

```
(3988, 300)
```

```
X_test_vec.shape
```

```
(998, 300)
```

Model building

```
from sklearn.ensemble import RandomForestClassifier
RF=RandomForestClassifier()
RF.fit(X_train_vec,y_train)
```

```
RandomForestClassifier()
```

```
predictios=RF.predict(X_test_vec)
```

Evaluaction metrics

```
from sklearn.metrics import classification_report,confusion_matrix,f1_score,precision_score,recall_score
```

```
print(classification_report(predictios,y_test))
```

```
              precision    recall  f1-score   support

           0       0.89      0.73      0.81       726
           1       0.52      0.77      0.62       272

    accuracy                           0.74       998
   macro avg       0.71      0.75      0.71       998
weighted avg       0.79      0.74      0.76       998
```

```
confusion_matrix(predictios,y_test)
```

```
array([[533, 193],
       [ 63, 209]])
```

```
precision_score(predictios,y_test)
```

```
0.5199004975124378
```

```
recall_score(predictios,y_test)
```

```
0.7683823529411765
```

```
f1_score(predictios,y_test)
```

```
0.6201780415430267
```