Import the libraries

```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        import warnings
        warnings.filterwarnings("ignore")
        import re
        import nltk
        from nltk.corpus import stopwords
        from nltk.stem.porter import PorterStemmer
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LogisticRegression
        from sklearn.metrics import accuracy_score
```

```
In [2]: data = pd.read_csv('/home/tamanna/Downloads/Twitter_Data.csv')
        data
```

Out[2]:

	clean_text	category
0	when modi promised “minimum government maximum...	-1.0
1	talk all the nonsense and continue all the dra...	0.0
2	what did just say vote for modi welcome bjp t...	1.0
3	asking his supporters prefix chowkidar their n...	1.0
4	answer who among these the most powerful world...	1.0
...	...	...
162975	why these 456 crores paid neerav modi not reco...	-1.0
162976	dear rss terrorist payal gawar what about modi...	-1.0
162977	did you cover her interaction forum where she ...	0.0
162978	there big project came into india modi dream p...	0.0
162979	have you ever listen about like gurukul where ...	1.0

162980 rows × 2 columns

```
In [3]: nltk.download('stopwords')

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

Out[3]: True
```

```
In [4]: print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're
", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', '
he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it'
, "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had
', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', '
if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with
', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'af
ter', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off'
, 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'whe
n', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'mos
t', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so'
, 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 's
hould', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'ar
en', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", '
hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma',
'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan'
t", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "wo
n't", 'wouldn', "wouldn't"]
```

data preprocessing

In [5]:
```
data.shape
```

Out[5]:
```
(162980, 2)
```

In [6]:
```
# printing first 5 rows
data.head(5)
```

Out[6]:

	clean_text	category
0	when modi promised “minimum government maximum...	-1.0
1	talk all the nonsense and continue all the dra...	0.0
2	what did just say vote for modi welcome bjp t...	1.0
3	asking his supporters prefix chowkidar their n...	1.0
4	answer who among these the most powerful world...	1.0

In [7]:
```
data.isnull().sum()
```

Out[7]:
```
clean_text    4
category      7
dtype: int64
```

In [8]:
```
data =  data.dropna()
```

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
clean_text    0
category      0
dtype: int64
```

In [10]:
```
data['category'].value_counts()
```

Out[10]:
```
category
 1.0    72249
 0.0    55211
-1.0    35509
Name: count, dtype: int64
```

In [11]:
```
data['category'] = data['category'].replace({ -1.0 : 'Negative',
                                               0.0 : 'Neutral',
                                               1.0 : 'positive'})
```

In [12]:
```
data
```

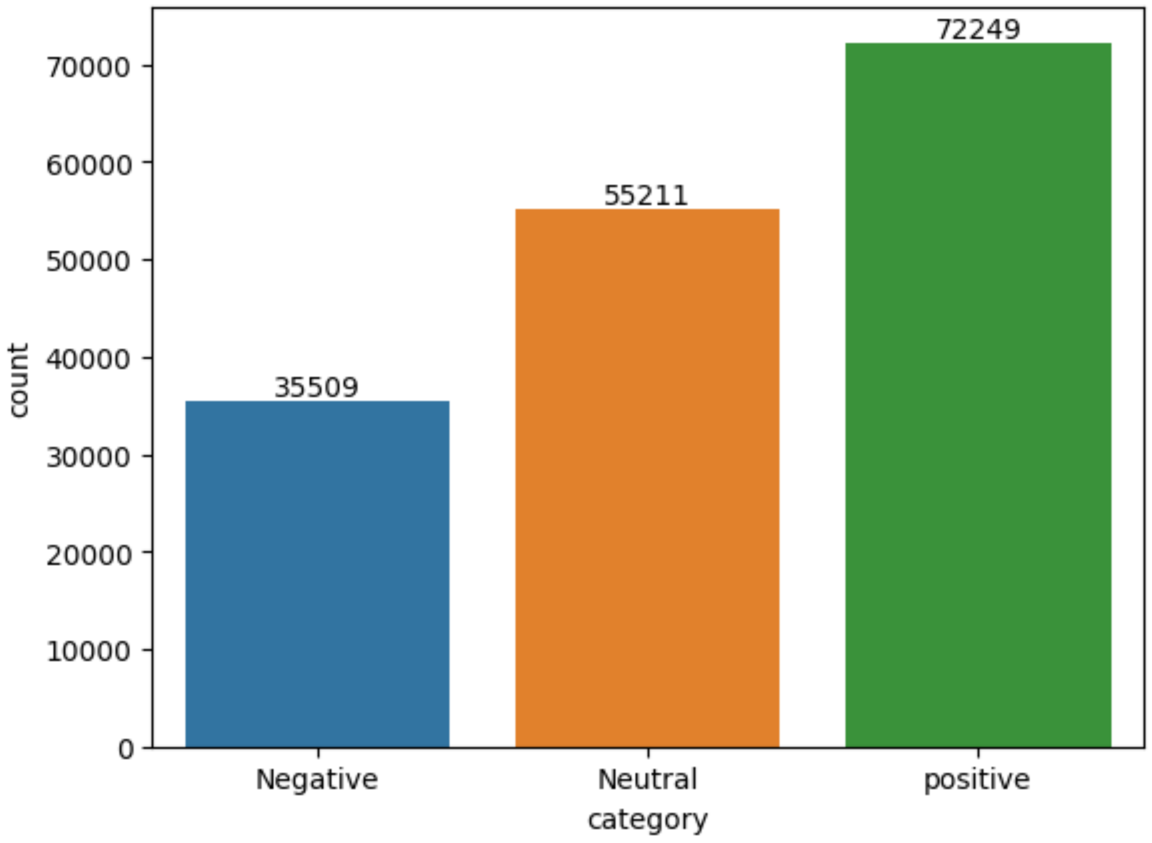
Out[12]:

	clean_text	category
0	when modi promised “minimum government maximum...	Negative
1	talk all the nonsense and continue all the dra...	Neutral
2	what did just say vote for modi welcome bjp t...	positive
3	asking his supporters prefix chowkidar their n...	positive
4	answer who among these the most powerful world...	positive
...	...	...
162975	why these 456 crores paid neerav modi not reco...	Negative
162976	dear rss terrorist payal gawar what about modi...	Negative
162977	did you cover her interaction forum where she ...	Neutral
162978	there big project came into india modi dream p...	Neutral
162979	have you ever listen about like gurukul where ...	positive

162969 rows × 2 columns

In [13]:
```
fig = sns.countplot(x = 'category', data = data)

for bars in fig.containers:
    fig.bar_label(bars)
```



Stemming

stemming is the process of reducing a word to its root word

```
In [14]: port_stem = PorterStemmer()
```

```
In [15]: def stemming(tweet):

             stemmed_tweet = re.sub('[^a-zA-Z]',' ', tweet)
             stemmed_tweet = stemmed_tweet.lower()
             stemmed_tweet = stemmed_tweet.split()
             stemmed_tweet = [port_stem.stem(word) for word in stemmed_tweet if not w
             stemmed_tweet = ' '.join(stemmed_tweet)

             return stemmed_tweet
```

```
In [16]: data['stemmed_tweet'] = data['clean_text'].apply(stemming)
```

```
In [17]: data.head(5)
```

Out[17]:

	clean_text	category	stemmed_tweet
0	when modi promised “minimum government maximum...	Negative	modi promis minimum govern maximum govern expe...
1	talk all the nonsense and continue all the dra...	Neutral	talk nonsens continu drama vote modi
2	what did just say vote for modi welcome bjp t...	positive	say vote modi welcom bjp told rahul main campa...
3	asking his supporters prefix chowkidar their n...	positive	ask support prefix chowkidar name modi great s...
4	answer who among these the most powerful world...	positive	answer among power world leader today trump pu...

now we have a new column stemmed_tweet which contains only lower case alphabets. There are no numbers, upperletters or any special characters in this data

In [18]:
```
data = data.rename(columns = {'category' : 'Target'})
data
```

Out[18]:

	clean_text	Target	stemmed_tweet
0	when modi promised “minimum government maximum...	Negative	modi promis minimum govern maximum govern expe...
1	talk all the nonsense and continue all the dra...	Neutral	talk nonsens continu drama vote modi
2	what did just say vote for modi welcome bjp t...	positive	say vote modi welcom bjp told rahul main campa...
3	asking his supporters prefix chowkidar their n...	positive	ask support prefix chowkidar name modi great s...
4	answer who among these the most powerful world...	positive	answer among power world leader today trump pu...
...	...	...	...
162975	why these 456 crores paid neerav modi not reco...	Negative	crore paid neerav modi recov congress leader h...
162976	dear rss terrorist payal gawar what about modi...	Negative	dear rss terrorist payal gawar modi kill plu m...
162977	did you cover her interaction forum where she ...	Neutral	cover interact forum left
162978	there big project came into india modi dream p...	Neutral	big project came india modi dream project happ...
162979	have you ever listen about like gurukul where ...	positive	ever listen like gurukul disciplin maintain ev...

162969 rows × 3 columns

In [19]:
```
# separating the labels(Target) and data(stemmed_tweet)
x = data['stemmed_tweet'].values
y = data['Target'].values
```

In [20]:
```
print(x)
```

```
['modi promis minimum govern maximum govern expect begin difficult job refor
m state take year get justic state busi exit psu templ'
 'talk nonsens continu drama vote modi'
 'say vote modi welcom bjp told rahul main campaign modi think modi relax'
 ... 'cover interact forum left'
 'big project came india modi dream project happen realiti'
 'ever listen like gurukul disciplin maintain even narendra modi rss maintai
n cultur indian attack polit someon attack hinduism rss take action proud']
```

In [21]:
```
print(y)
```

```
['Negative' 'Neutral' 'positive' ... 'Neutral' 'Neutral' 'positive']
```

splitting the data into train and test dataset

In [22]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size = 0.2, st
```

In [23]:
```
print(x.shape, x_train.shape, x_test.shape)
```

```
(162969,) (130375,) (32594,)
```

converting textual data to numerical

In [24]:
```
# befor training data we need to convert text data into numerical data so th

vectorizer = TfidfVectorizer()

x_train = vectorizer.fit_transform(x_train)
x_test = vectorizer.transform(x_test)
```

In [25]:
```
print(x_train)
```

```
  (0, 28743)	0.16144962488758455
  (0, 20978)	0.15479943877392816
  (0, 50384)	0.12988943705645903
  (0, 40067)	0.03080998402945807
  (0, 66302)	0.1435219122722211
  (0, 17850)	0.3817977967743573
  (0, 43120)	0.22291765786234816
  (0, 2034)	0.22573882918468724
  (0, 28781)	0.16797131234622645
  (0, 40393)	0.16396791065369923
  (0, 5012)	0.1613648461909348
  (0, 20422)	0.22664524725751634
  (0, 49423)	0.16421731815523039
  (0, 47497)	0.09688951516287975
  (0, 39426)	0.18227525083206278
  (0, 35892)	0.23025229242873607
  (0, 57982)	0.26612892189445914
  (0, 68125)	0.2248614991186051
  (0, 39514)	0.1809730472121113
  (0, 51719)	0.17437940942836028
  (0, 39243)	0.4748512441133469
  (1, 24267)	0.12664393809286514
  (1, 37545)	0.28573581089266126
  (1, 21221)	0.17809460359176146
  (1, 14076)	0.22357950358283196
  :	:
  (130372, 19548)	0.23311499327392124
  (130372, 37068)	0.3399756765290224
  (130372, 49124)	0.17228211716582778
  (130372, 35639)	0.17304961608306435
  (130372, 36113)	0.1686654280070792
  (130372, 40067)	0.039623339012206556
  (130372, 40393)	0.210871777951681
  (130373, 4596)	0.401225594070786
  (130373, 26177)	0.401225594070786
  (130373, 65466)	0.378212959960119
  (130373, 65464)	0.37080454650253203
  (130373, 1124)	0.25038723908127153
  (130373, 68663)	0.2240980966278706
  (130373, 30584)	0.3201391520097798
  (130373, 4420)	0.2076545693858091
  (130373, 70095)	0.17348491742206887
  (130373, 13840)	0.17673993719277972
  (130373, 64305)	0.19401476302762596
  (130373, 19016)	0.1211203615339453
  (130373, 25407)	0.15914105400679707
  (130374, 69154)	0.7622550029419848
  (130374, 69176)	0.3463632080152143
  (130374, 69672)	0.4458279895952802
  (130374, 17758)	0.3050575580335642
  (130374, 40067)	0.08471793554704667
```

In [26]:
```
print(x_test)
```

```
(0, 70684)    0.3654149628097209
(0, 66014)    0.45784497985286926
(0, 55677)    0.24200813532769908
(0, 53119)    0.5740234816604777
(0, 44277)    0.3329310283820778
(0, 40067)    0.07279159587619888
(0, 36113)    0.30985338388557265
(0, 13125)    0.23796675969722694
(1, 69743)    0.23602561185313362
(1, 69468)    0.19826967725609115
(1, 64863)    0.2524313718526405
(1, 62901)    0.17942214875438922
(1, 58144)    0.20903722861121768
(1, 53990)    0.2311755329435266
(1, 51265)    0.16670571485515115
(1, 40067)    0.038496991353423785
(1, 39524)    0.14993361029132493
(1, 35472)    0.21172428361723097
(1, 31506)    0.31226674264058546
(1, 30016)    0.33252736500192187
(1, 28131)    0.2095224388833978
(1, 17578)    0.18553313487676096
(1, 17089)    0.2247105266594949
(1, 11741)    0.28858700512651103
(1, 8128)     0.2828109113508008
:     :
(32591, 51602)        0.3019021318548467
(32591, 40067)        0.08155814336840318
(32591, 36622)        0.38797626729888585
(32591, 34162)        0.30099471305438064
(32591, 17758)        0.2936795838220085
(32591, 3295) 0.7581769403009194
(32592, 63851)        0.1780969041382061
(32592, 50204)        0.3903573880116547
(32592, 42309)        0.2792840862760357
(32592, 40067)        0.05065546332463321
(32592, 38360)        0.6672336989745263
(32592, 35125)        0.21807274324914974
(32592, 25656)        0.25776241593997956
(32592, 17758)        0.18240331097645682
(32592, 14693)        0.31831030211182737
(32592, 3997) 0.2038298697371942
(32593, 67618)        0.3892792396020725
(32593, 59437)        0.34428853537620835
(32593, 57055)        0.34845828635894904
(32593, 55677)        0.24131849787254966
(32593, 40067)        0.07258416561411966
(32593, 40041)        0.3970864486194937
(32593, 39795)        0.32716391657924615
(32593, 12936)        0.3721035349584999
(32593, 12418)        0.37659780331334297
```

training the logistic regression model

```
In [27]: model1 = LogisticRegression(max_iter=1000)
```

```
In [28]: model1.fit(x_train, y_train)
```

```
Out[28]: ▾        LogisticRegression

         LogisticRegression(max_iter=1000)
```

Model Evaluation

Accuracy Score

```
In [29]: # accuracy on training data
         x_train_prediction = model1.predict(x_train)
         training_data_accuracy = accuracy_score(y_train, x_train_prediction)
```

```
In [30]: print('Accuracy score on training data :', training_data_accuracy)

        Accuracy score on training data : 0.8784352828379675
```

```
In [31]: x_test_prediction = model1.predict(x_test)
         test_data_accuracy = accuracy_score(y_test, x_test_prediction)
```

```
In [32]: print('Accuracy score on test data :', test_data_accuracy)

        Accuracy score on test data : 0.8436215254341289
```

checking the model's accuracy on input data

```
In [33]: input_data = ['modi promis minimum govern maximum govern expect begin diffic

         # Vectorize the input data using the same TF-IDF vectorizer
         input_text_vectorized = vectorizer.transform(input_data).toarray()

         # Predict using the trained logistic regression model
         predicted_label = model1.predict(input_text_vectorized)

         print("Predicted Label:", predicted_label)

        Predicted Label: ['Negative']
```