Import ML Algorithms

In [3]:
```
import pandas as pd
import string
from nltk.corpus import stopwords
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import cross_val_score,StratifiedKFold,cross_val_predict
from sklearn.metrics import confusion_matrix,classification_report
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
import seaborn as sns
import plotly.express as px
from nltk.stem import WordNetLemmatizer
from sklearn.feature_extraction.text import TfidfVectorizer
from imblearn.over_sampling import RandomOverSampler
import warnings as w
w.filterwarnings('ignore')
```

In [5]:
```
df = pd.read_csv("Spam Email Detection - spam.csv", encoding='latin1')
```

In [6]:
```
df.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'] ,inplace=True,axis=1)
```

In [8]:
```
df.head(10)
```
Out[8]:

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
5	spam	FreeMsg Hey there darling it's been 3 week's n...
6	ham	Even my brother is not like to speak with me. ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...

In [9]:
```
df.shape
```
Out[9]:
```
(5572, 2)
```
In [10]:
```
df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   v1      5572 non-null   object
 1   v2      5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

In [11]:
```
df.rename(columns={'v1':'target' , 'v2': 'text'},inplace=True)
```

In [12]:
```
df.head()
```

Out[12]:

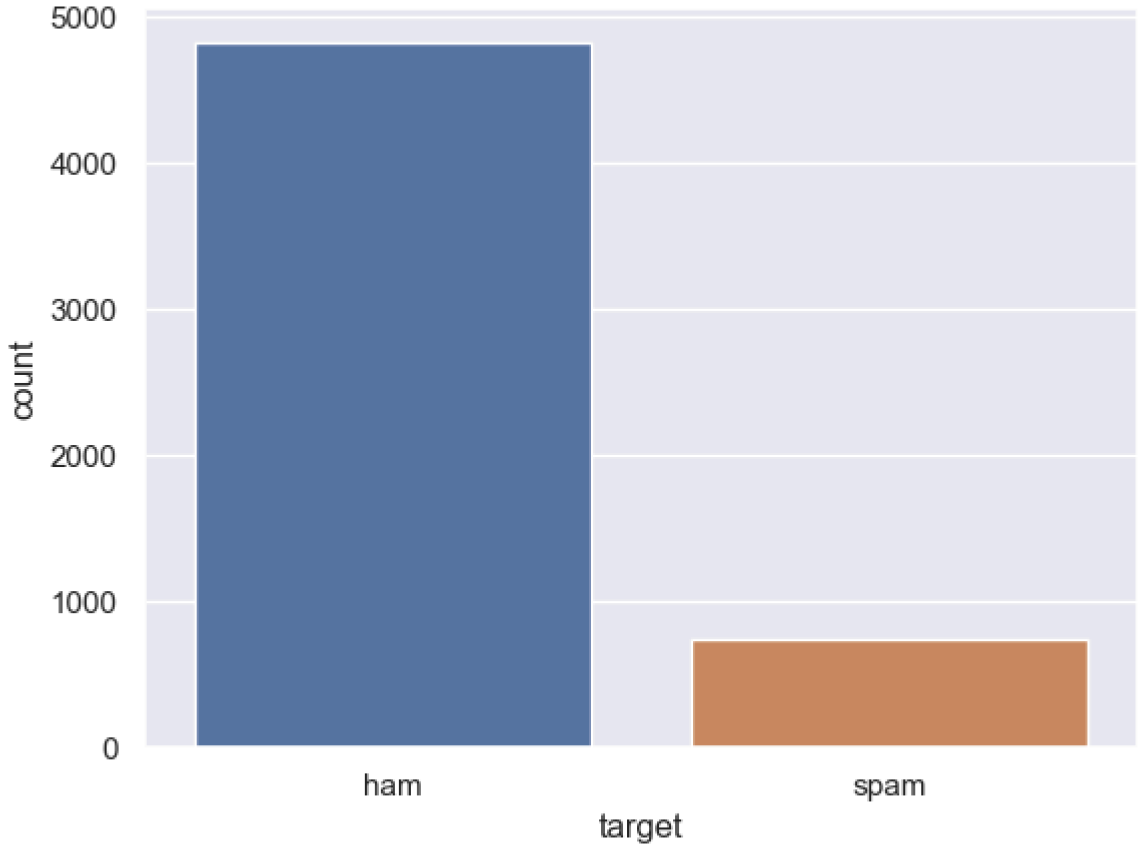
	target	text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Visualisation

In [13]:
```
df.columns
```

Out[13]:
```
Index(['target', 'text'], dtype='object')
```

In [14]:
```
sns.set()
```

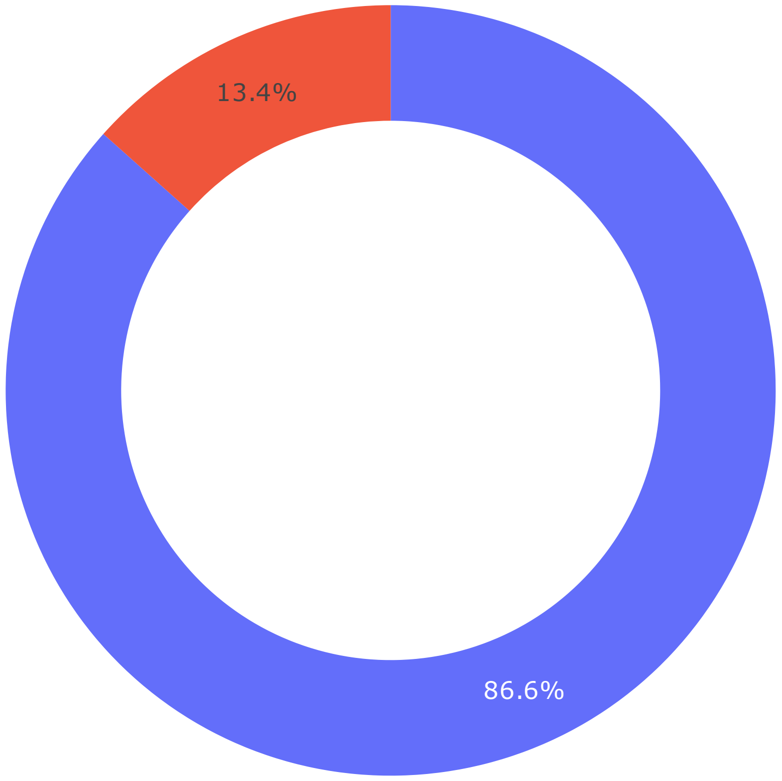
In [15]:
```
sns.countplot(x=df['target'])
```
Out[15]:
```
<Axes: xlabel='target', ylabel='count'>
```


In [16]:
```
ham_spam = df.target.value_counts()
```
In [17]:
```
value = ham_spam.values
index = ham_spam.index
```

In [18]:
```
ham_spam
```

Out[18]:
```
ham     4825
spam     747
Name: target, dtype: int64
```

```
In [19]: px.pie(df,
               values = value,
               names = index,
               hole = .7)
```



```
In [20]: le = LabelEncoder()
In [21]: df['target'] = le.fit_transform(df['target'])
```

```
In [22]: df.head()
```

Out[22]:

	target	text
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

```
In [23]: df.isnull().sum()
```

```
Out[23]: target    0
         text      0
         dtype: int64
```

```
In [24]: df.duplicated().sum()
```

```
Out[24]: 409
```

```
In [25]: df.drop_duplicates(inplace=True)
```

```
In [26]: df.shape
```

```
Out[26]: (5163, 2)
```

```
In [27]: df.duplicated().sum()
```

```
Out[27]: 0
```

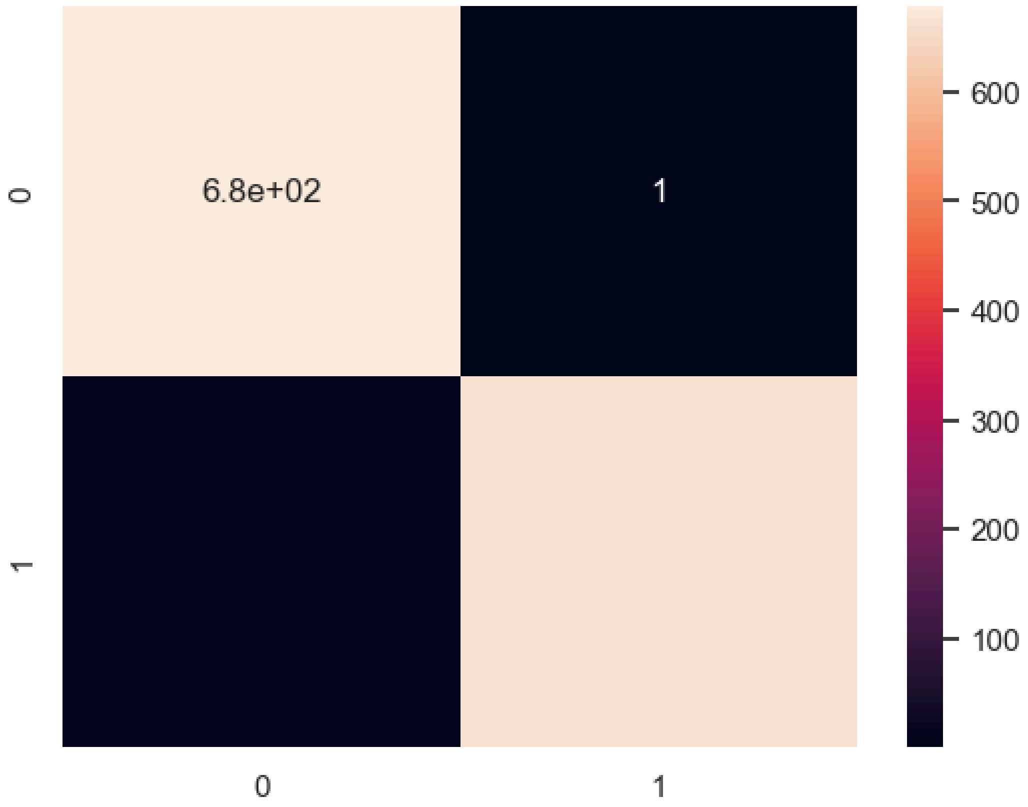
```
In [28]: tar = df.target.value_counts()
         tar

Out[28]: 0    4516
         1     647
         Name: target, dtype: int64

In [29]: string.punctuation

Out[29]: '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'

In [30]: def text_preprocessing(text):

             remove_punctuation = [word for word in text if word not in string.punctuation]
             join_word = ''.join(remove_punctuation)
             split_word = join_word.split()

             stop_word= [ word for word in split_word if word.lower() not in stopwords.words('english')]
             join_ = ' '.join(stop_word)

             lemmatize_text =WordNetLemmatizer().lemmatize(join_)
             return lemmatize_text

In [31]: df['text'] = df['text'].apply(text_preprocessing)

In [32]: df.head()

Out[32]:
```

	target	text
0	0	Go jurong point crazy Available bugis n great ...
1	0	Ok lar Joking wif u oni
2	1	Free entry 2 wkly comp win FA Cup final tkts 2...
3	0	U dun say early hor U c already say
4	0	Nah dont think goes usf lives around though

```
In [33]: df['text'][50]

Out[33]: 'thinked First time saw class'

In [34]: import numpy as np

In [35]: x =TfidfVectorizer().fit_transform(df['text']).toarray()
         y=df['target']

In [36]: new_x , new_y =RandomOverSampler(random_state=100).fit_resample(x,y)

In [37]: new_x

Out[37]: array([[0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.],
                ...,
                [0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.],
                [0., 0., 0., ..., 0., 0., 0.]])

In [38]: def result(model,new_x,new_y):
             mull = model(alpha=1.0 , fit_prior=True )
             mod = mull.fit(new_x,new_y)
             st = StratifiedKFold(n_splits=6)
             cro = cross_val_score(mod , new_x,new_y , cv = st)
             return cro

In [39]: result(MultinomialNB,new_x,new_y)

Out[39]: array([0.97675963, 0.98339973, 0.98671096, 0.98471761, 0.98272425,
                0.98538206])
```

```
In [40]: result(BernoulliNB,new_x,new_y)

Out[40]: array([0.98871182, 0.98339973, 0.98803987, 0.98471761, 0.98671096,
                0.99202658])

In [41]: from sklearn.model_selection import train_test_split

In [42]: xtrain,xtest,ytrain,ytest = train_test_split(new_x,new_y,test_size=.15)

In [43]: mull = BernoulliNB(alpha=1.0 , fit_prior=True )
         mod = mull.fit(xtrain,ytrain)

In [44]: y_pre = mod.predict(xtest)
         y_pre

Out[44]: array([1, 0, 1, ..., 0, 0, 0])

In [45]: mod.score(xtest,ytest)

Out[45]: 0.9874538745387453

In [46]: mod.score(xtrain,ytrain)

Out[46]: 0.9887977074378013

In [47]: print(classification_report(ytest,y_pre))

              precision    recall  f1-score   support

           0       0.98      1.00      0.99       679
           1       1.00      0.98      0.99       676

    accuracy                           0.99      1355
   macro avg       0.99      0.99      0.99      1355
weighted avg       0.99      0.99      0.99      1355
```

```
In [48]: cm = confusion_matrix(ytest,y_pre)
         cm

Out[48]: array([[678,   1],
                [ 16, 660]], dtype=int64)

In [49]: sns.heatmap(cm,annot = True)

Out[49]: <Axes: >
```



```
In [ ]:
```