Problem statement

Dataset Link: https://www.kaggle.com/datasets/shantanudhak spam-detection-dataset-classification

TO CLASSIFY THE MAIL AS SPAM OR HAM BY USING MACHINE MODEL,AND GIVE THE BEST ACCURACY SCORE TO THE PROBLEMS.

In [1]:

```
#Import the standard librarys
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import re
from textblob import TextBlob
from nltk.tokenize.toktok import ToktokTokenizer
import string
import nltk
from sklearn.metrics import accuracy_score,classification_report,confusion_mat
import spacy
nlp = spacy.load('en_core_web_sm', disable=['ner'])
import warnings
warnings.filterwarnings('ignore')
```

```
2022-09-06 11:22:47.195127: I tensorflow/core/util/util.cc:169] oneDNN custom
operations are on. You may see slightly different numerical results due to flo
ating-point round-off errors from different computation orders. To turn them o
ff, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2022-09-06 11:22:47.199655: W tensorflow/stream_executor/platform/default/dso_
loader.cc:64] Could not load dynamic library 'libcudart.so.11.0'; dlerror: lib
cudart.so.11.0: cannot open shared object file: No such file or directory
2022-09-06 11:22:47.199670: I tensorflow/stream_executor/cuda/cudart_stub.cc:2
9] Ignore above cudart dlerror if you do not have a GPU set up on your machin
e.
2022-09-06 11:22:48.785024: W tensorflow/stream_executor/platform/default/dso_
loader.cc:64] Could not load dynamic library 'libcuda.so.1'; dlerror: libcuda.
so.1: cannot open shared object file: No such file or directory
2022-09-06 11:22:48.785059: W tensorflow/stream_executor/cuda/cuda_driver.cc:2
69] failed call to cuInit: UNKNOWN ERROR (303)
2022-09-06 11:22:48.785080: I tensorflow/stream_executor/cuda/cuda_diagnostic
s.cc:156] kernel driver does not appear to be running on this host (vinod-Vost
ro-3400): /proc/driver/nvidia/version does not exist
```

In [2]:

```
#Read the data uisng the pandas
data=pd.read_csv('/home/vinod/Downloads/spam.csv',encoding="ISO-8859-1")
data.head().style.background_gradient(cmap='jet')
```

Out[2]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...	nan	nan	nan
1	ham	Ok lar... Joking wif u oni...	nan	nan	nan
2	spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's	nan	nan	nan
3	ham	U dun say so early hor... U c already then say...	nan	nan	nan
4	ham	Nah I don't think he goes to usf, he lives around here though	nan	nan	nan

In [3]:
```
#data shape
data.shape
```

Out[3]:
```
(5572, 5)
```

In [4]:
```
#data information
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB
```

In [5]:
```
#data columns
data.columns
```

Out[5]:
```
Index(['v1', 'v2', 'Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], dtype='object')
```

DataPrepocessing

In [6]:
```
#Check the null the values in the dataset
data.isna().sum()
```

Out[6]:
```
v1               0
v2               0
Unnamed: 2    5522
Unnamed: 3    5560
Unnamed: 4    5566
dtype: int64
```

In [7]:
```
#Remove the unwanted columns
data.drop(['Unnamed: 2','Unnamed: 3','Unnamed: 4'],axis=1,inplace=True)
data.head()
```

Out[7]:

	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...

In [8]:

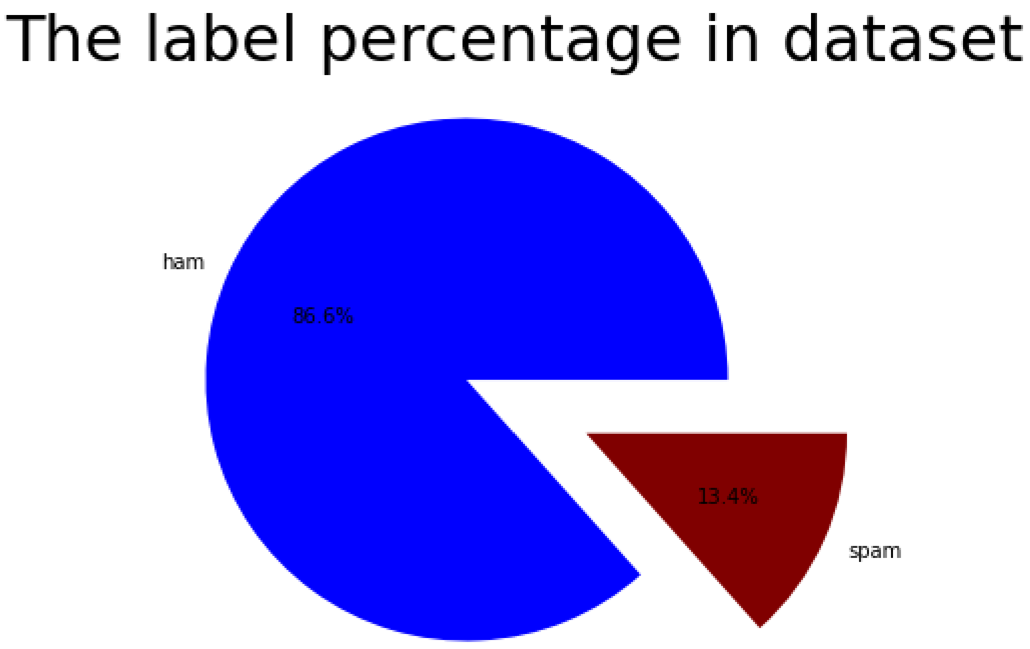
```
#Rename the columns
data.rename(columns={'v1':'label','v2':'text'},inplace=True)
data.head()
```

Out[8]:

	label	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...

In [9]:

```
#to visuaize the label percentage in the train dataset
label=data['label'].value_counts()
#to visualize the above information in the pie chart
plt.figure(figsize=(18,6))
plt.pie(label,labels=['ham','spam'],colors=['blue','maroon'],autopct='%1.1f%%'
#set the title name with fontsize
plt.title("The label percentage in dataset",fontsize=32)
plt.show()
```



In [10]:

```
#Creat the function and remove the punctuation
```

```
def remove_punctuation(text):
    for punctuation in string.punctuation:
        text=text.replace(punctuation,'')
    return text
#And apply function to the train dataset
data['text']=data['text'].apply(remove_punctuation)
```

In [11]:
```
#create the remove the stopwords
stop_words_list=nltk.corpus.stopwords.words('english')
stop_words_list.remove('no')
stop_words_list.remove('not')
```

In [12]:
```
#install the tokeninzer
tokenizer = ToktokTokenizer()
#Create function to remove the stopwords
def remove_stopwords(text,is_lower_case=True):
    tokens=tokenizer.tokenize(text)
    tokens=[token.strip() for token in tokens]
    if is_lower_case:
        filterd_tokens=[token for token in tokens if token not in stop_words_l
    else:
        filterd_tokens=[token for token in tokens if token.lower() not in stop
    filterd_text=' '.join(filterd_tokens)
    return filterd_text
data['text']=data['text'].apply(remove_stopwords)
```

In [13]:
```
#remove the special characters
def remove_special_characters(text):
    text=re.sub('[^a-zA-z0-9\s]','',text)
    return text
data['text']=data['text'].apply(remove_stopwords)
data.head().style.background_gradient(cmap='jet')
```

Out[13]:

	label	text
0	ham	Go jurong point crazy Available bugis n great world la e buffet Cine got amore wat
1	ham	Ok lar Joking wif u oni
2	spam	Free entry 2 wkly comp win FA Cup final tkts 21st May 2005 Text FA 87121 receive entry questionstd txt rateTCs apply 08452810075over18s
3	ham	U dun say early hor U c already say
4	ham	Nah I dont think goes usf lives around though

In [14]:
```
#Remove html patternst
def remove_html(text):
    html_pattran=re.compile('<.*?>')
    return html_pattran.sub(r' ',text)
#And apply the above function to the train data
data['text']=data['text'].apply(remove_html)
```

In [15]:
```
#Remove the html url from the train dataset
def remove_url(text):
    url=re.compile(r'https?://\S+|www\.\S+')
    return url.sub(r' ',text)
#And apply to the train dataset
data['text']=data['text'].apply(remove_url)
```

In [16]:
```
#Remove the numbers in the dataset
def remove_numbers(text):
    text=''.join([i for i in text if  not i.isdigit()])
    return text
#And apply the above function to the train data
data['text']=data['text'].apply(remove_numbers)
```

In [17]:
```
def cleanse(word):
    rx=re.compile(r'\D*\d')
    if rx.match(word):
        return ' '
    return word
#Remove the alpha numaric
def alpha_numeric(strings):
    nstrings=[" ".join(filter(None,(cleanse(word) for word in string.split())))
    str1=" ".join(nstrings)
    return str1
#Apply to the above function to the train dataset
data['text']=data['text'].apply(alpha_numeric)
```

In [18]:
```
#lemmatizer
def lemmatize(text):
    text=nlp(text)
    text=" ".join([word.lemma_ if word.lemma_!='-PRON-' else word for word in
    return text
#Apply the above function to the rain dataset
data['text']=data['text'].apply(lemmatize)
```

In [19]:
```
data.head(11).style.background_gradient(cmap='Reds')
```

Out[19]:

	label	text
0	ham	go jurong point crazy Available bugis n great world la e buffet Cine get amore wat
1	ham	ok lar Joking wif u oni
2	spam	free entry wkly comp win FA Cup final tkts st May Text FA receive entry questionstd txt ratetcs apply over
3	ham	U dun say early hor U c already say
4	ham	Nah I do not think go usf live around though
5	spam	FreeMsg hey darle week no word back I d like fun still tb ok XxX std chgs send å£ rcv
6	ham	even brother not like speak they treat like aids patent
7	ham	as per request Melle Melle Oru Minnaminunginte Nurungu Vettam set callertune Callers Press copy friend Callertune
8	spam	WINNER as value network customer select receivea å£ prize reward to claim call Claim code KL Valid hour
9	spam	have mobile month U r entitle Update late colour mobile camera Free Call the Mobile Update Co FREE
10	ham	I m go to home soon do not want talk stuff anymore tonight k I ve cry enough today

In [20]:
```
data['text'][11]
```

Out[20]:
```
'six chance win cash from pound txt CSH send Cost pday day TsandCs apply Reply
hl info'
```

Modeling

In [21]:
```
#Divided the data into X and y
X=data['text'].values
y=data['label'].values
```

In [22]:
```
#Install the TfidfVectorizer for covert the categorical data to array
vector=TfidfVectorizer()
vector.fit(X)
X=vector.transform(X)
```

In [23]:
```
#Divided the data into train_test and use 30% used for testing
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=
```

LogisticRegression

In [24]:
```
#Install the logisticregression model
logistic=LogisticRegression()
#Let's fit the train data
logistic.fit(X_train,y_train)
```

Out[24]:
```
LogisticRegression()
```

In [25]:
```
#Prediction to the test data
logistic_pred=logistic.predict(X_test)
logistic_pred
```

Out[25]:
```
array(['ham', 'ham', 'ham', ..., 'ham', 'ham', 'ham'], dtype=object)
```

In [26]:
```
#Check the test score and train score to the model
print(f'The Logisticregression model test score is {logistic.score(X_test,y_te
#Train score for the data
print(f'The Logisticregression model train scores is {logistic.score(X_train,y
#Check the accuracy_score to the model
print(f'The logisticRegression accuracy_score {accuracy_score(y_test,logistic_
```

```
The Logisticregression model test score is 94.56
The Logisticregression model train scores is 96.82
The logisticRegression accuracy_score 94.56
```

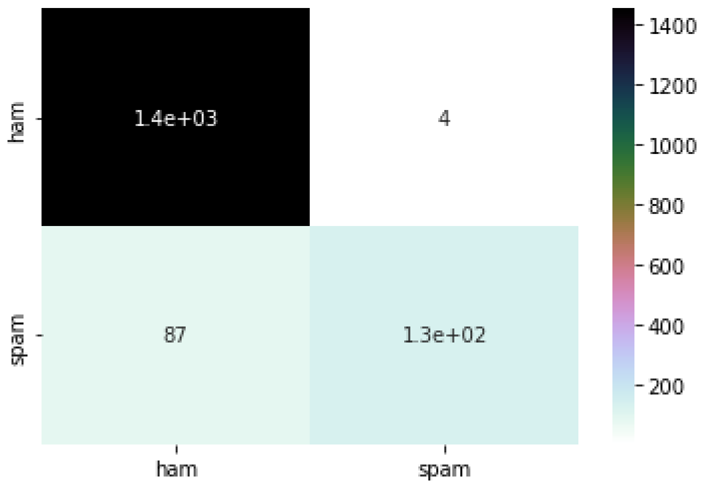
Classification_report and Confusion_matrix

In [27]:
```
#Classification report
print(classification_report(y_test,logistic_pred))
#confusion_matrix
cn=confusion_matrix(y_test,logistic_pred)
sns.heatmap(cn,annot=True,cmap='cubehelix_r',xticklabels=['ham','spam'],ytickl
              precision    recall  f1-score   support

         ham       0.94      1.00      0.97      1453
        spam       0.97      0.60      0.74       219

    accuracy                           0.95      1672
   macro avg       0.96      0.80      0.86      1672
weighted avg       0.95      0.95      0.94      1672
```

Out[27]:
```
<AxesSubplot:>
```



DecisionTreeClassifier

In [28]:
```
#Import the DecisionTreeClassifier from sklean library
from sklearn.tree import DecisionTreeClassifier
#install the library
tree=DecisionTreeClassifier()
#Fit the train data to the model
tree.fit(X_train,y_train)
```

Out[28]:
```
DecisionTreeClassifier()
```

In [29]:
```
#DecisionTreeClassifier Testdata prediction
tree_pred=tree.predict(X_test)
tree_pred
```

Out[29]:
```
array(['ham', 'ham', 'ham', ..., 'ham', 'ham', 'ham'], dtype=object)
```

In [30]:
```
#Check the test score and train score to the model
print(f'The DecisionTreeClassifier model test score is {tree.score(X_test,y_te
#Train score for the data
print(f'The DecisionTreeClassifier model train scores is {tree.score(X_train,y
#Check the accuracy_score to the model
print(f'The DecisionTreeClassifier accuracy_score {accuracy_score(y_test,tree_
```

```
The DecisionTreeClassifier model test score is 95.39
The DecisionTreeClassifier model train scores is 100.00
The DecisionTreeClassifier accuracy_score 95.39
```

Classification_report and Confusion_matrix

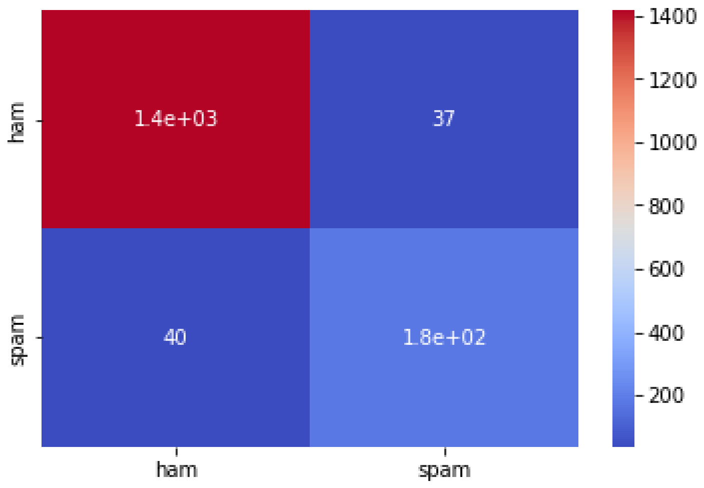
In [31]:
```
#Classification report
print(classification_report(y_test,tree_pred))
#confusion_matrix
cn=confusion_matrix(y_test,tree_pred)
sns.heatmap(cn,annot=True,cmap='coolwarm',xticklabels=['ham','spam'],yticklabe
```

```
              precision    recall  f1-score   support

         ham       0.97      0.97      0.97      1453
        spam       0.83      0.82      0.82       219

    accuracy                           0.95      1672
   macro avg       0.90      0.90      0.90      1672
weighted avg       0.95      0.95      0.95      1672
```

Out[31]:
```
<AxesSubplot:>
```



RandomForestClassifier

In [32]:
```
#Import the RandomForestClassifier from the sklean library
from sklearn.ensemble import RandomForestClassifier
#install the RandomForestClassifier
random=RandomForestClassifier()
#Let's fit the traing data to the model
random.fit(X_train,y_train)
```

Out[32]:
```
RandomForestClassifier()
```

In [33]:
```
#RandomForestClassifier prediction to the test data
random_pred=random.predict(X_test)
random_pred
```

Out[33]:
```
array(['ham', 'ham', 'spam', ..., 'ham', 'ham', 'ham'], dtype=object)
```

In [34]:
```
#Check the test score and train score to the model
print(f'The RandomForestClassifier model test score is {random.score(X_test,y_
#Train score for the data
```

```
print(f'The RandomForestClassifier model train scores is {random.score(X_train
#Check the accuracy_score to the model
print(f'The RandomForestClassifier accuracy_score {accuracy_score(y_test,rando
```

```
The RandomForestClassifier model test score is 97.25
The RandomForestClassifier model train scores is 100.00
The RandomForestClassifier accuracy_score 97.25
```

Classification_report and Confusion_matrix

In [35]:

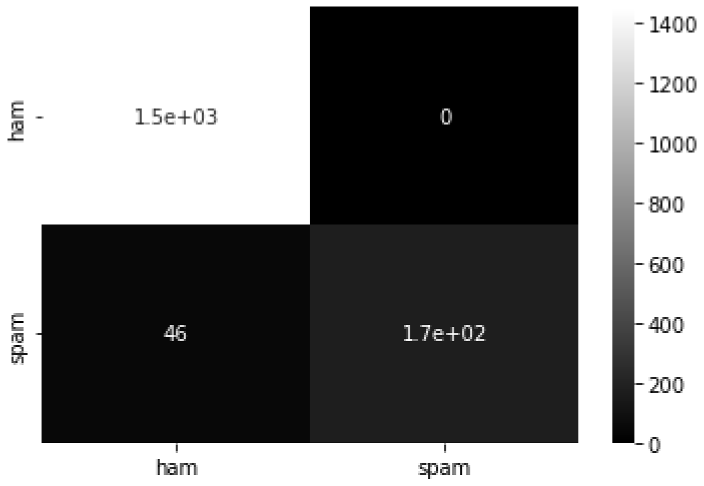
```
#Classification report
print(classification_report(y_test,random_pred))
#confusion_matrix
cn=confusion_matrix(y_test,random_pred)
sns.heatmap(cn,annot=True,cmap='gist_yarg_r',xticklabels=['ham','spam'],ytickl
```

```
              precision    recall  f1-score   support

         ham       0.97      1.00      0.98      1453
        spam       1.00      0.79      0.88       219

    accuracy                           0.97      1672
   macro avg       0.98      0.89      0.93      1672
weighted avg       0.97      0.97      0.97      1672
```

Out[35]: <AxesSubplot:>



XGBClassifier

In [36]:

```
#Install the XGBClassifier from the xgboost
from xgboost import XGBClassifier
#Install the XGBClassifier
xgb=XGBClassifier()
#Let's fit the traning data to the model
xgb.fit(X_train,y_train)
```

Out[36]:
```
XGBClassifier(base_score=0.5, booster=None, colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0, gpu_id=-1,
              importance_type='gain', interaction_constraints=None,
              learning_rate=0.300000012, max_delta_step=0, max_depth=6,
              min_child_weight=1, missing=nan, monotone_constraints=None,
              n_estimators=100, n_jobs=0, num_parallel_tree=1, random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method=None, validate_parameters=False, verbosity=None)
```

In [37]:
```
#XGBClassifier prediction to the test dataset
xgb_pred=xgb.predict(X_test)
xgb_pred
```

Out[37]:
```
array(['ham', 'ham', 'ham', ..., 'ham', 'ham', 'ham'], dtype=object)
```

In [38]:
```
#Check the test score and train score to the model
print(f'The XGBClassifier model test score is {xgb.score(X_test,y_test)*100:.2
#Train score for the data
print(f'The XGBClassifier model train scores is {xgb.score(X_train,y_train)*10
#Check the accuracy_score to the model
print(f'The XGBClassifier accuracy_score {accuracy_score(y_test,xgb_pred)*100:
```

```
The XGBClassifier model test score is 97.01
The XGBClassifier model train scores is 99.18
The XGBClassifier accuracy_score 97.01
```

Classification_report and Confusion_matrix

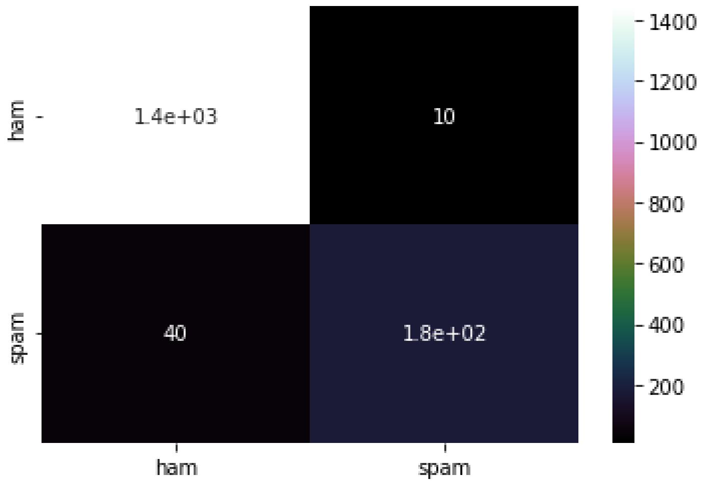
In [39]:
```
#Classification report
print(classification_report(y_test,xgb_pred))
#confusion_matrix
cn=confusion_matrix(y_test,xgb_pred)
sns.heatmap(cn,annot=True,cmap='cubehelix',xticklabels=['ham','spam'],yticklab
```

```
              precision    recall  f1-score   support

         ham       0.97      0.99      0.98      1453
        spam       0.95      0.82      0.88       219

    accuracy                           0.97      1672
   macro avg       0.96      0.91      0.93      1672
weighted avg       0.97      0.97      0.97      1672
```

Out[39]:
```
<AxesSubplot:>
```



MultinomialNB

```
In [40]: #Import the MultinomialNB algorithm to train the our model
         from sklearn.naive_bayes import MultinomialNB
         #install the model
         multinomial=MultinomialNB()
         #fit the train data to our model
         multinomial.fit(X_train,y_train)
```

```
Out[40]: MultinomialNB()
```

```
In [41]: #MultinomialNB prediction to the test data
         multinomial_pred=multinomial.predict(X_test)
         multinomial_pred
```

```
Out[41]: array(['ham', 'ham', 'ham', ..., 'ham', 'ham', 'ham'], dtype='<U4')
```

```
In [42]: #Check the test score and train score to the model
         print(f'The MultinomialNB model test score is {multinomial.score(X_test,y_test
         #Train score for the data
         print(f'The MultinomialNB model train scores is {multinomial.score(X_train,y_t
         #Check the accuracy_score to the model
         print(f'The MultinomialNB accuracy_score {accuracy_score(y_test,multinomial_pr
```

```
The MultinomialNB model test score is 95.22
The MultinomialNB model train scores is 96.67
The MultinomialNB accuracy_score 95.22
```

Classification_report and Confusion_matrix

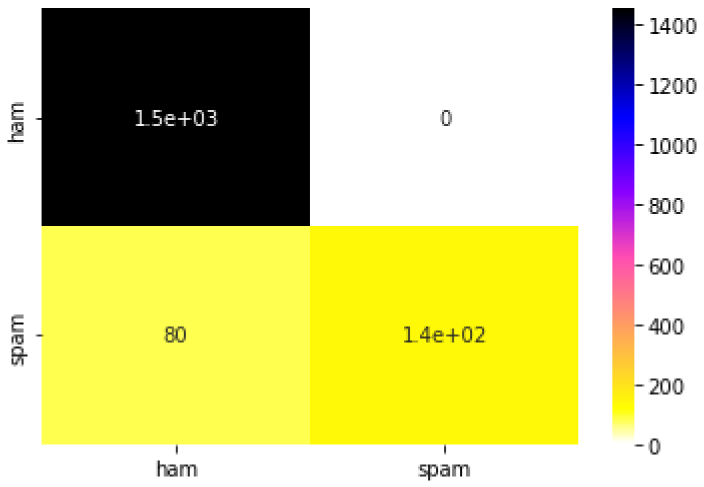
```
In [43]: #Classification report
         print(classification_report(y_test,multinomial_pred))
         #confusion_matrix
         cn=confusion_matrix(y_test,multinomial_pred)
         sns.heatmap(cn,annot=True,cmap='gnuplot2_r',xticklabels=['ham','spam'],yticklа
                       precision    recall  f1-score   support

                  ham       0.95      1.00      0.97      1453
                 spam       1.00      0.63      0.78       219

             accuracy                           0.95      1672
            macro avg       0.97      0.82      0.87      1672
         weighted avg       0.95      0.95      0.95      1672
```

Out[43]:

```
<AxesSubplot:>
```



In [44]:

```python
#Import the AdaBosstClassifier from sklearn library
from sklearn.ensemble import AdaBoostClassifier
#Install the model
adam=AdaBoostClassifier()
#Fit the traiing data to the model
adam.fit(X_train,y_train)
```

Out[44]:

```
AdaBoostClassifier()
```

In [45]:

```python
#AdaBoostClassifier prediction to the dataset
adam_pred=adam.predict(X_test)
adam_pred
```

Out[45]:

```
array(['ham', 'ham', 'ham', ..., 'ham', 'ham', 'ham'], dtype=object)
```

In [46]:

```python
#Check the test score and train score to the model
print(f'The AdaBoostClassifier model test score is {adam.score(X_test,y_test)*
#Train score for the data
print(f'The AdaBoostClassifier model train scores is {adam.score(X_train,y_tra
#Check the accuracy_score to the model
print(f'The AdaBoostClassifier accuracy_score {accuracy_score(y_test,adam_pred
```

```
The AdaBoostClassifier model test score is 95.10
The AdaBoostClassifier model train scores is 97.41
The AdaBoostClassifier accuracy_score 95.10
```

Classification_report and Confusion_matrix

In [47]:

```python
#Classification report
print(classification_report(y_test,adam_pred))
```

```
#confusion_matrix
cn=confusion_matrix(y_test,adam_pred)
sns.heatmap(cn,annot=True,cmap='tab10_r',xticklabels=['ham','spam'],yticklabel
```

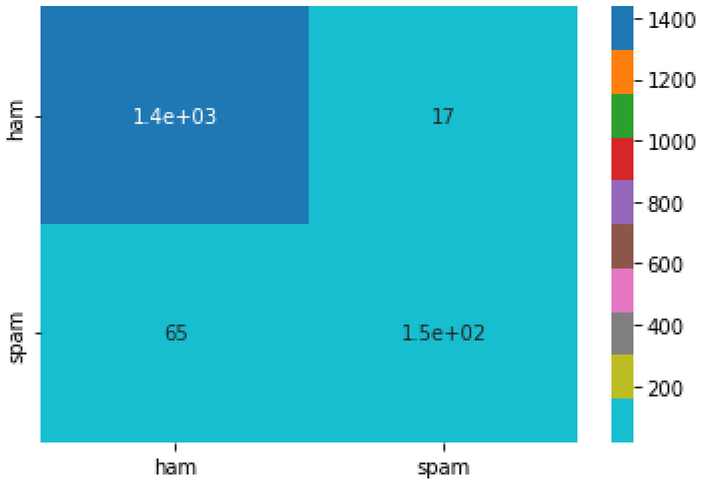
```
              precision    recall  f1-score   support

         ham       0.96      0.99      0.97      1453
        spam       0.90      0.70      0.79       219

    accuracy                           0.95      1672
   macro avg       0.93      0.85      0.88      1672
weighted avg       0.95      0.95      0.95      1672
```

Out[47]:

```
<AxesSubplot:>
```



Print the all model classification report

In [48]:

```
#Create a function to print the all model classification report
def model_classification(models,names,X_train,y_train,X_test,y_test):
    #create for loop for the model and name in zip of models,names
    for (model,name) in zip(models,names):
        #Prediction to the test data
        y_pred=model.predict(X_test)
        #Using the prediction to visualize the classification_report
        result=classification_report(y_test,y_pred)
        #And finaly print the classification report and divide the each classi
        print("Classification Report \n",result)
        print("----------------------------------------------------------------
        print("****************************************************************
        print("################################################################
```

In [49]:

```
#Create a list for the models
models=[logistic,tree,random,xgb,multinomial,adam]
#Create a list for the names
names=['LogisticRegression','DecisionTreeClassifier','RandomForestClassifier',
#And finally define the function with models and names
model_classification(models,names,X_train,y_train,X_test,y_test)
```

```
Classification Report
              precision    recall  f1-score   support

         ham       0.94      1.00      0.97      1453
        spam       0.97      0.60      0.74       219

    accuracy                           0.95      1672
   macro avg       0.96      0.80      0.86      1672
weighted avg       0.95      0.95      0.94      1672

--------------------------------------------------------------------------------
---------
********************************************************************************
*********
################################################################################
##########
Classification Report
              precision    recall  f1-score   support

         ham       0.97      0.97      0.97      1453
        spam       0.83      0.82      0.82       219

    accuracy                           0.95      1672
   macro avg       0.90      0.90      0.90      1672
weighted avg       0.95      0.95      0.95      1672

--------------------------------------------------------------------------------
---------
********************************************************************************
*********
################################################################################
##########
Classification Report
              precision    recall  f1-score   support

         ham       0.97      1.00      0.98      1453
        spam       1.00      0.79      0.88       219

    accuracy                           0.97      1672
   macro avg       0.98      0.89      0.93      1672
weighted avg       0.97      0.97      0.97      1672

--------------------------------------------------------------------------------
---------
********************************************************************************
*********
################################################################################
##########
Classification Report
              precision    recall  f1-score   support

         ham       0.97      0.99      0.98      1453
        spam       0.95      0.82      0.88       219

    accuracy                           0.97      1672
   macro avg       0.96      0.91      0.93      1672
weighted avg       0.97      0.97      0.97      1672

--------------------------------------------------------------------------------
---------
```

```
*****************************************************************************
*********
################################################################################
##########
Classification Report
               precision    recall  f1-score   support

         ham       0.95      1.00      0.97      1453
        spam       1.00      0.63      0.78       219

    accuracy                           0.95      1672
   macro avg       0.97      0.82      0.87      1672
weighted avg       0.95      0.95      0.95      1672

-----------------------------------------------------------------------------
---------
*****************************************************************************
*********
################################################################################
##########
Classification Report
               precision    recall  f1-score   support

         ham       0.96      0.99      0.97      1453
        spam       0.90      0.70      0.79       219

    accuracy                           0.95      1672
   macro avg       0.93      0.85      0.88      1672
weighted avg       0.95      0.95      0.95      1672

-----------------------------------------------------------------------------
---------
*****************************************************************************
*********
################################################################################
##########
```

About the data Classification reports:

Observe the above classification report, DecisionTreeClassifier given 96%,RandomForestClassifier given 97% and XGBClassifier given 97% accuracy_scores, the reaming model also give 95% accuracy_scores.And we also visualize the confusin_matrix using the seaborn

How to perfom the our model to the data

In [50]:
```
text="free entry wkly comp win FA Cup final tkts st May Text FA receive entry
text=[text]
text_int=vector.transform(text)
prediction=tree.predict(text_int)
f"The email is {prediction[0]}"
```

```
Out[50]: 'The email is spam'
```

```
In [51]: text="go jurong point crazy Available bugis n great world la e buffet Cine get
         text=[text]
         text_int=vector.transform(text)
         prediction=random.predict(text_int)
         f"The email is {prediction[0]}"
```

```
Out[51]: 'The email is ham'
```

```
In [52]: text='I m go to home soon do not want talk stuff anymore tonight k I ve cry en
         text=[text]
         text_int=vector.transform(text)
         prediction=xgb.predict(text_int)
         f"The email is {prediction[0]}"
```

```
Out[52]: 'The email is ham'
```

```
In [53]: text='WINNER as value network customer select receivea å£ prize reward to clai
         text=[text]
         text_int=vector.transform(text)
         prediction=xgb.predict(text_int)
         f"The email is {prediction[0]}"
```

```
Out[53]: 'The email is spam'
```

```
In [54]: text='have mobile month U r entitle Update late colour mobile camera Free Call
         text=[text]
         text_int=vector.transform(text)
         prediction=random.predict(text_int)
         f"The email is {prediction[0]}"
```

```
Out[54]: 'The email is spam'
```

CONCLUSION

The dataset is taken from the Kaggle website, We predict the text is spam or harm, The data prepocessing contain several steps such as basicEDA, text preprocessing, build different models, such as LogisticRegression,DecisiontreeClassification,RandomForestClassic the above model DecisiontreeClassification,RandomForestClassication,XGBboostClas have good accuracy score compare to the other model. After that We do Perfome the models for the test data and use the demo it.

```
In [ ]:
```