

TASK 4- SPAM SMS DETECTION DATASET



Objective

- The objective is to build a model that can classify SMS messages as spam or legitimate.

Machine learning models applied

- Gaussian Naive Bayes
- Multinomial Naive Bayes
- K Nearest Neighbors
- Random Forest
- Support vector machines

Dataset source & brief

- The dataset has been sourced from kaggle. The SMS Spam Collection is a set of SMS tagged messages that have been collected for SMS Spam research. It contains one set of SMS messages in English of 5,574 messages, tagged according to being ham (legitimate) or spam.

Import the basic libraries

In [50]:

```

import os
import numpy as np
import pandas as pd

#visualization
import matplotlib.pyplot as plt
import seaborn as sns

#evaluation
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.model_selection import cross_val_score

import warnings
warnings.filterwarnings('ignore')

```

Load and read the dataset

In [51]:

```

df=pd.read_csv(r"C:\Users\manme\Documents\Priya\Codsoft\spam.csv",encoding='latin-1')
df.head()

```

Out[51]:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	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 fina...	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 aro...	NaN	NaN	NaN

Basic info about the dataset

In [52]:

```
df.shape #check shape
```

Out[52]:

(5572, 5)

In [53]:

```
df.columns #check column names
```

Out[53]:

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

Drop the non significant variables

In [54]:

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

Renaming the columns for easy understanding

In [55]:

```
df.rename(columns = {'v1': 'Label', 'v2': 'Messages'}, inplace = True)
```

In [56]:

```
df.isnull().sum() #check null values
```

Out[56]:

```
Label      0
Messages   0
dtype: int64
```

In [57]:

```
df.info() #check info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  -
 0   Label      5572 non-null   object
 1   Messages   5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```


Exploratory Data Analysis

In [73]:

```
from dataprep.eda import create_report
report = create_report(df, title='Data Report')
report
```

0%|
| 0/258 [00:00<...

Out[73]:

Data Report
[Data Report Overview](#)
[Variables](#) 
[Label Messages](#)
[Interactions](#) [Missing Values](#)

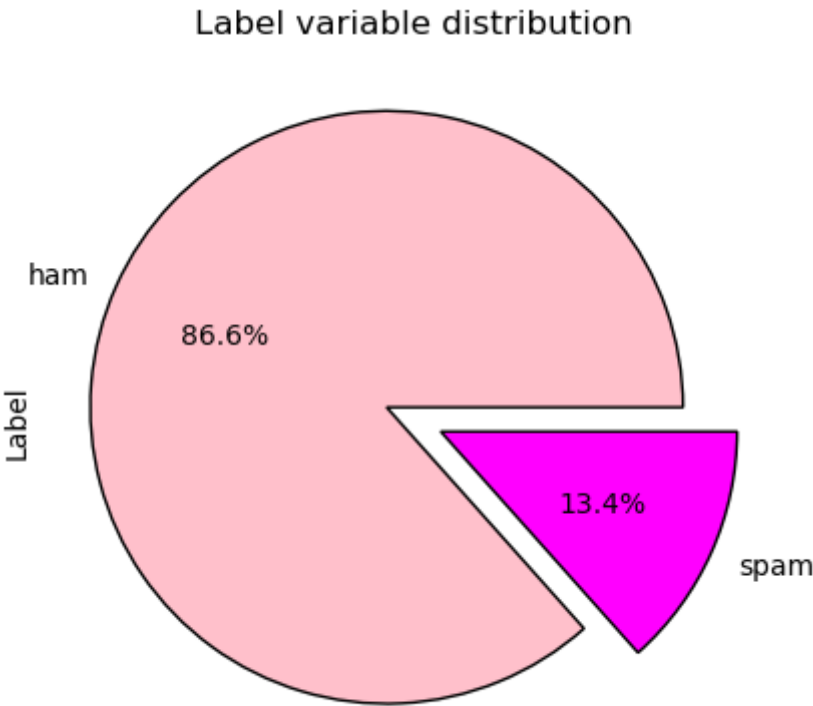
Overview

Dataset Statistics

Number of Variables	2
Number of Rows	5572
Missina Cells	0

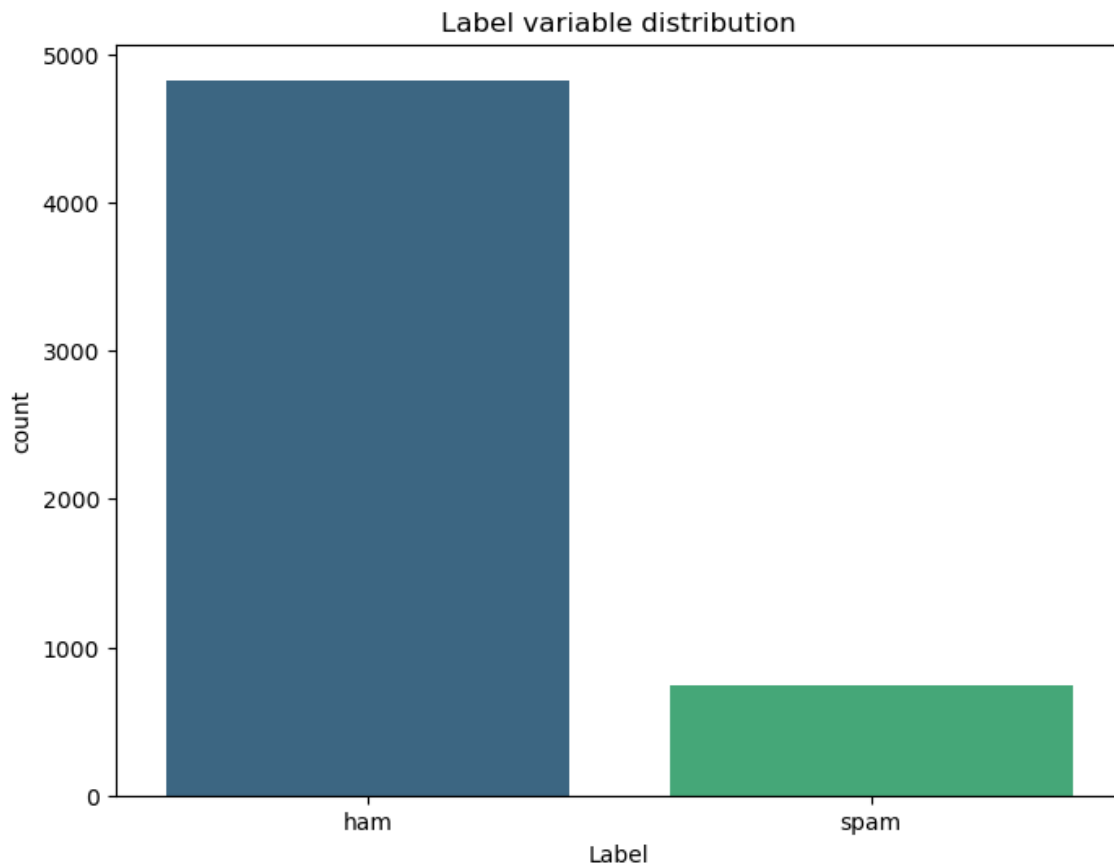
In [59]:

```
df['Label'].value_counts().plot(kind='pie',explode=[0.1,0.1],autopct='%0.1f%%',
                                colors=('pink','fuchsia'),wedgeprops={'edgecolor': 'black'})
plt.title('Label variable distribution')
plt.show()
```



In [64]:

```
plt.figure(figsize=(8,6))
sns.countplot(x='Label',data=df, palette="viridis")
plt.xlabel("Label")
plt.title('Label variable distribution')
plt.show()
```



Check balance of data

In [9]:

```
df['Label'].value_counts()/len(df)*100 # Data is imbalanced
```

Out[9]:

```
ham      86.593683
spam     13.406317
Name: Label, dtype: float64
```

Balancing the data

In [10]:

```
ham = df[df['Label']=='ham']
spam = df[df['Label']=='spam']
```

In [11]:

```
print(ham.shape, spam.shape)
```

```
(4825, 2) (747, 2)
```

In [12]:

```
spam = spam.sample(ham.shape[0], replace=True)
```

In [13]:

```
print(ham.shape, spam.shape)
```

```
(4825, 2) (4825, 2)
```

In [14]:

```
data = ham.append(spam, ignore_index=True)  
data.shape
```

Out[14]:

```
(9650, 2)
```

In [15]:

```
data.head()
```

Out[15]:

	Label	Messages
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	ham	U dun say so early hor... U c already then say...
3	ham	Nah I don't think he goes to usf, he lives aro...
4	ham	Even my brother is not like to speak with me. ...

In [16]:

```
data['Label'].value_counts() # Data is balanced
```

Out[16]:

```
ham      4825  
spam     4825  
Name: Label, dtype: int64
```

Data cleaning, dropping the stop words and passing through stemming

In [17]:

```
import re
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data] C:\Users\manme\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
```

Out[17]:

True

In [18]:

```
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
corpus = []

for i in range(0, len(data)):
    review = re.sub('[^a-zA-Z]', ' ', data['Messages'][i])
    review = review.lower()
    review = review.split()
    review = [ps.stem(word) for word in review if not word in stopwords.words('english')]
    review = ' '.join(review)
    corpus.append(review)
```

In [19]:

corpus

Out[19]:

```
['go jurong point crazi avail bugi n great world la e buffet cine got a
mor wat',
 'ok lar joke wif u oni',
 'u dun say earli hor u c already say',
 'nah think goe usf live around though',
 'even brother like speak treat like aid patent',
 'per request mell mell oru minnaminungint nurungu vettam set callertun
caller press copi friend callertun',
 'gonna home soon want talk stuff anymor tonight k cri enough today',
 'search right word thank breather promis wont take help grant fulfil p
romis wonder bless time',
 'date sunday',
 'oh k watch',
 'eh u rememb spell name ye v naughti make v wet',
 'fine way u feel way gota b',
 'serious spell name',
 'go tri month ha ha joke',
 'dav first lar da stock comin'.
```

Bag of words and labels for train and test data

In [20]:

```
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer()
x = cv.fit_transform(corpus).toarray()
```

In [21]:

```
x.shape
```

Out[21]:

```
(9650, 6219)
```

In [22]:

```
x
```

Out[22]:

```
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]], dtype=int64)
```

In [23]:

```
data.head()
```

Out[23]:

	Label	Messages
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	ham	U dun say so early hor... U c already then say...
3	ham	Nah I don't think he goes to usf, he lives aro...
4	ham	Even my brother is not like to speak with me. ...

Label encoder for label variable

In [24]:

```
data['Label'] = data['Label'].astype('category')
data['Label'] = data['Label'].cat.codes
```

In [25]:

```
data.shape
```

Out[25]:

```
(9650, 2)
```


Split the data into train and test data

In [26]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, data['Label'], test_size=0.25, ra
```

Model Building

Model No. 1 Gaussian Naive Bayes

In [27]:

```
# Model building
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(x_train, y_train)
# Predict the model
y_pred_train_nb = nb.predict(x_train)
y_pred_test_nb = nb.predict(x_test)
# Evaluate
accuracy_nb_test=accuracy_score(y_test,y_pred_test_nb)
accuracy_nb_train=accuracy_score(y_train,y_pred_train_nb)
print('Gaussian Naive Bayes -Train accuracy:', accuracy_score(y_train, y_pred_train_nb))
print('-----'*10)
print('Gaussian Naive Bayes -Test accuracy:', accuracy_score(y_test, y_pred_test_nb))
```

Gaussian Naive Bayes -Train accuracy: 0.9491502003592649

Gaussian Naive Bayes -Test accuracy: 0.9183588893493576

In [28]:

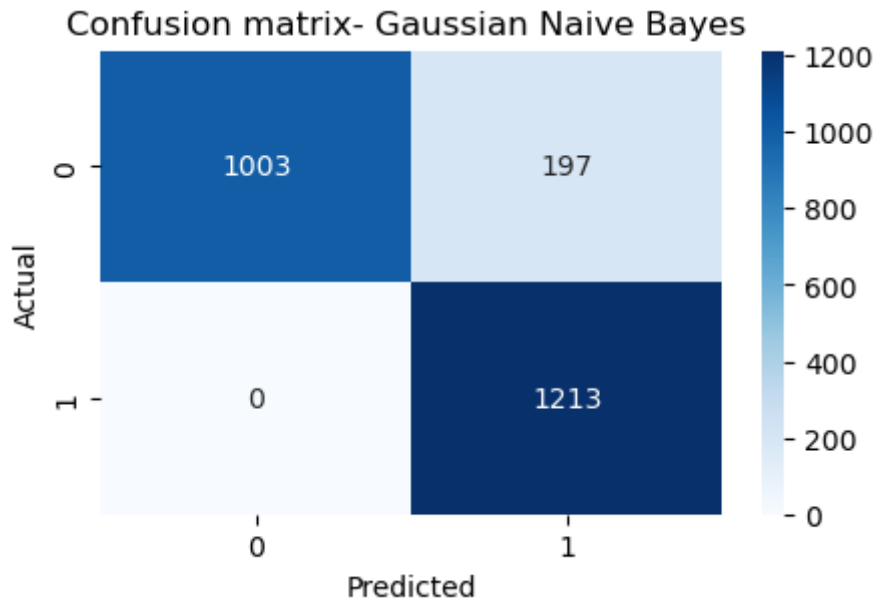
```
train_accuracy_nb = cross_val_score(nb,x_train, y_train, cv=10)
crossval_train_nb=train_accuracy_nb.mean()
test_accuracy_nb = cross_val_score(nb,x_test, y_test, cv=10)
crossval_test_nb=test_accuracy_nb.mean()
print('Gaussian Naives Bayes after Cross validation Train accuracy:', crossval_train_nb)
print('-----'*5)
print('Gaussian Naives Bayes after Cross validation Test accuracy:', crossval_test_nb)
```

Gaussian Naives Bayes after Cross validation Train accuracy: 0.92040913015
90213

-
Gaussian Naives Bayes after Cross validation Test accuracy: 0.886018997976
7498

In [29]:

```
plt.figure(figsize=(5,3))
sns.heatmap(confusion_matrix(y_test,y_pred_test_nb),cmap='Blues',annot=True, fmt='g')
plt.title("Confusion matrix- Gaussian Naive Bayes ")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



Model No. 2 - Multinomial Naive Bayes

In [30]:

```
# Model building
from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB()
mnb.fit(x_train, y_train)
# Predict the model
y_pred_train_mnb = mnb.predict(x_train)
y_pred_test_mnb = mnb.predict(x_test)
# Evaluate
accuracy_mnb_test=accuracy_score(y_test,y_pred_test_mnb)
accuracy_mnb_train=accuracy_score(y_train,y_pred_train_mnb)
print('Multinomial Naive Bayes -Train accuracy:', accuracy_score(y_train, y_pred_train_mnb))
print('-----'*10)
print('Multinomial Naive Bayes -Test accuracy:', accuracy_score(y_test, y_pred_test_mnb))
```

Multinomial Naive Bayes -Train accuracy: 0.9818985767583253

Multinomial Naive Bayes -Test accuracy: 0.9689183588893494

In [31]:

```

train_accuracy_mnb = cross_val_score(mnb,x_train, y_train, cv=10)
crossval_train_mnb=train_accuracy_mnb.mean()
test_accuracy_mnb = cross_val_score(mnb,x_test, y_test, cv=10)
crossval_test_mnb=test_accuracy_mnb.mean()
print('Multinomial Naives Bayes after Cross validation Train accuracy:', crossval_train_
print('-----'*5)
print('Multinomial Naives Bayes after Cross validation Test accuracy:', crossval_test_mn

```

Multinomial Naives Bayes after Cross validation Train accuracy: 0.97706341
74671221

-

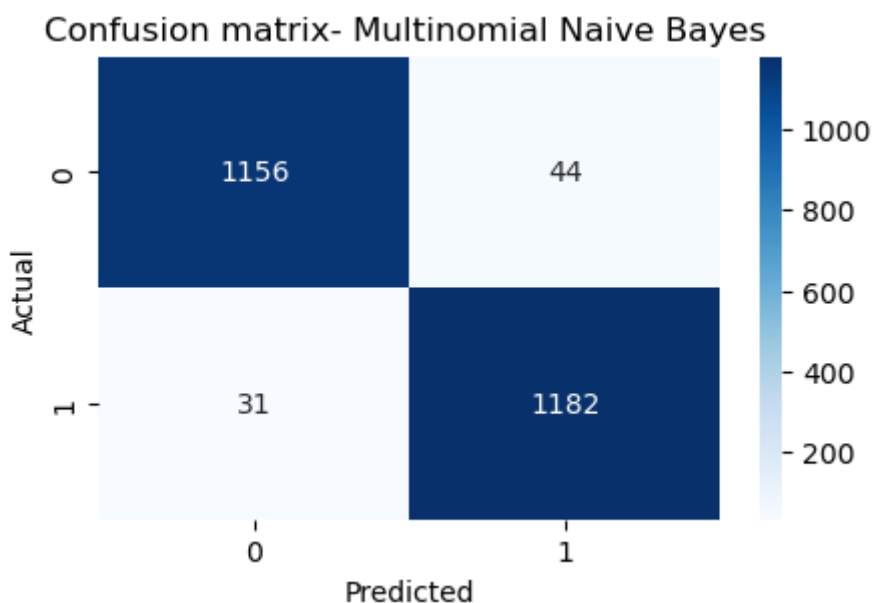
Multinomial Naives Bayes after Cross validation Test accuracy: 0.965589314
4953878

In [32]:

```

plt.figure(figsize=(5,3))
sns.heatmap(confusion_matrix(y_test,y_pred_test_mnb),cmap='Blues',annot=True, fmt='g')
plt.title("Confusion matrix- Multinomial Naive Bayes ")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()

```



Model No. 3 - K Nearest Neighbor

In [33]:

```
# Model building with K point as 3
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train, y_train)
# Predict
y_pred_train_knn = knn.predict(x_train)
y_pred_test_knn = knn.predict(x_test)
#Evaluate
accuracy_knn_test=accuracy_score(y_test,y_pred_test_knn)
accuracy_knn_train=accuracy_score(y_train,y_pred_train_knn)
print('K nearest neighbor - Train accuracy:', accuracy_score(y_train, y_pred_train_knn))
print('-----'*10)
print('K nearest neighbor - Test accuracy:', accuracy_score(y_test, y_pred_test_knn))
```

K nearest neighbor - Train accuracy: 0.9988945695730275

K nearest neighbor - Test accuracy: 0.9933692498963945

In [34]:

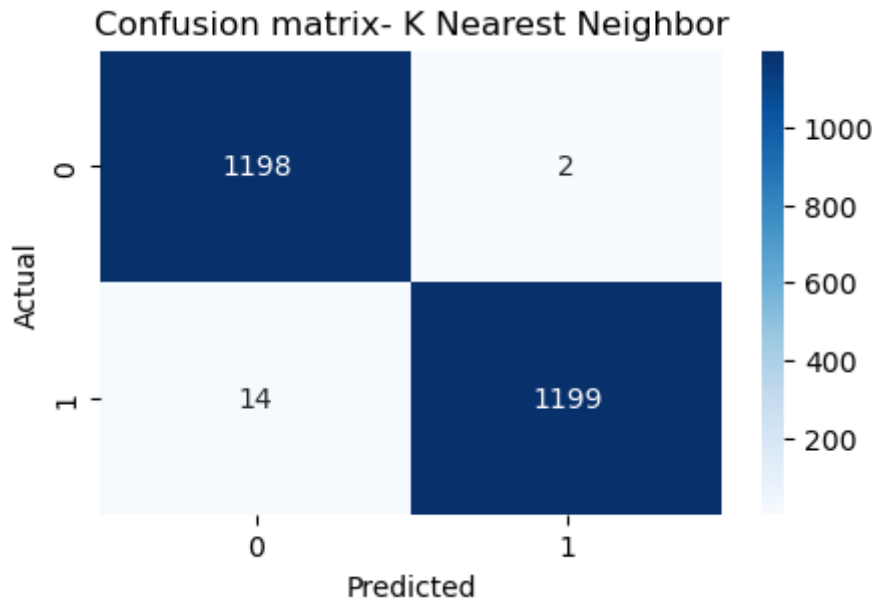
```
train_accuracy_knn = cross_val_score(knn,x_train, y_train, cv=10)
crossval_train_knn=train_accuracy_knn.mean()
test_accuracy_knn = cross_val_score(knn,x_test, y_test, cv=10)
crossval_test_knn=test_accuracy_knn.mean()
print('K Nearest Neighbor after Cross validation Train accuracy:', crossval_train_knn)
print('-----'*5)
print('K Nearest Neighbor after Cross validation Test accuracy:', crossval_test_knn)
```

K Nearest Neighbor after Cross validation Train accuracy: 0.98949875060177
44

-
K Nearest Neighbor after Cross validation Test accuracy: 0.926233668255546
8

In [35]:

```
plt.figure(figsize=(5,3))
sns.heatmap(confusion_matrix(y_test,y_pred_test_knn),cmap='Blues',annot=True, fmt='g')
plt.title("Confusion matrix- K Nearest Neighbor")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



Model No. 4 - Random Forest Classifier

In [36]:

```
# Model building
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=200,oob_score=False)
rf.fit(x_train,y_train)
# Predict
y_pred_train_rf=rf.predict(x_train)
y_pred_test_rf=rf.predict(x_test)
# Evaluate
accuracy_rf_test=accuracy_score(y_test,y_pred_test_rf)
accuracy_rf_train=accuracy_score(y_train,y_pred_train_rf)
print('Random Forest - Train accuracy:', accuracy_score(y_train, y_pred_train_rf))
print('-----'*10)
print('Random Forest - Test accuracy:', accuracy_score(y_test, y_pred_test_rf))
```

Random Forest - Train accuracy: 1.0

Random Forest - Test accuracy: 0.998756734355574

In [37]:

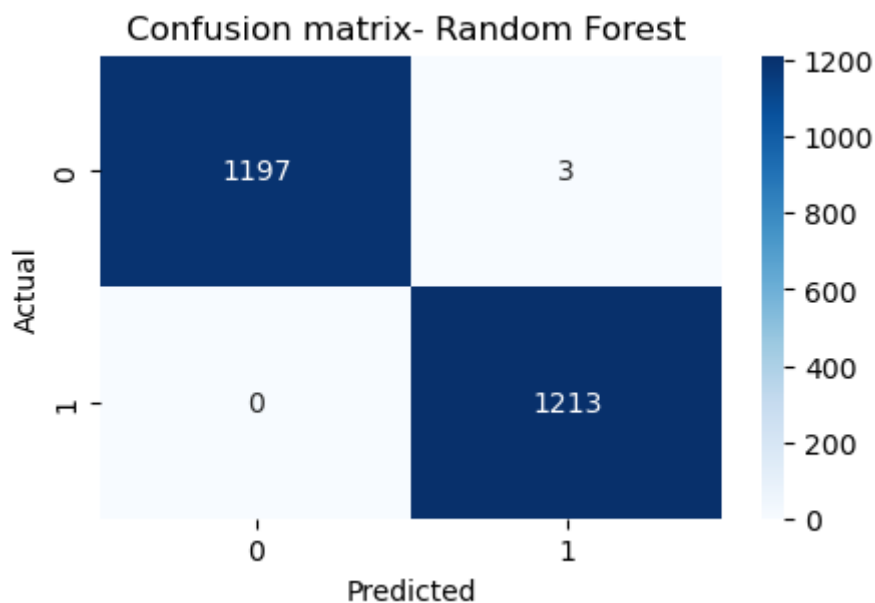
```
train_accuracy_rf = cross_val_score(rf,x_train, y_train, cv=10)
crossval_train_rf=train_accuracy_rf.mean()
test_accuracy_rf = cross_val_score(rf,x_test, y_test, cv=10)
crossval_test_rf=test_accuracy_rf.mean()
print('Random forest after Cross validation Train accuracy:', crossval_train_rf)
print('-----'*10)
print('Random forest after Cross validation Test accuracy:', crossval_test_rf)
```

Random forest after Cross validation Train accuracy: 0.9982040378105346

Random forest after Cross validation Test accuracy: 0.9875604403141182

In [38]:

```
plt.figure(figsize=(5,3))
sns.heatmap(confusion_matrix(y_test,y_pred_test_rf),cmap='Blues',annot=True, fmt='g')
plt.title("Confusion matrix- Random Forest ")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



Model No. 5 - Support Vector Machine

In [39]:

```
# Radial Basis Function Kernel (RBF) - (Default SVM) Model building
from sklearn.svm import SVC
svm_rbf = SVC(kernel='rbf')
svm_rbf.fit(x_train, y_train)
#Predict
y_pred_train_rbf = svm_rbf.predict(x_train)
y_pred_test_rbf = svm_rbf.predict(x_test)
#Evaluate
accuracy_rbf_test=accuracy_score(y_test,y_pred_test_rbf)
accuracy_rbf_train=accuracy_score(y_train,y_pred_train_rbf)
print('Rbf - SVM - Train accuracy:', accuracy_score(y_train, y_pred_train_rbf))
print('-----'*10)
print('Rbf - SVM - Test accuracy:', accuracy_score(y_test, y_pred_test_rbf))
```

Rbf - SVM - Train accuracy: 0.9982036755561697

Rbf - SVM - Test accuracy: 0.9958557811852465

In [41]:

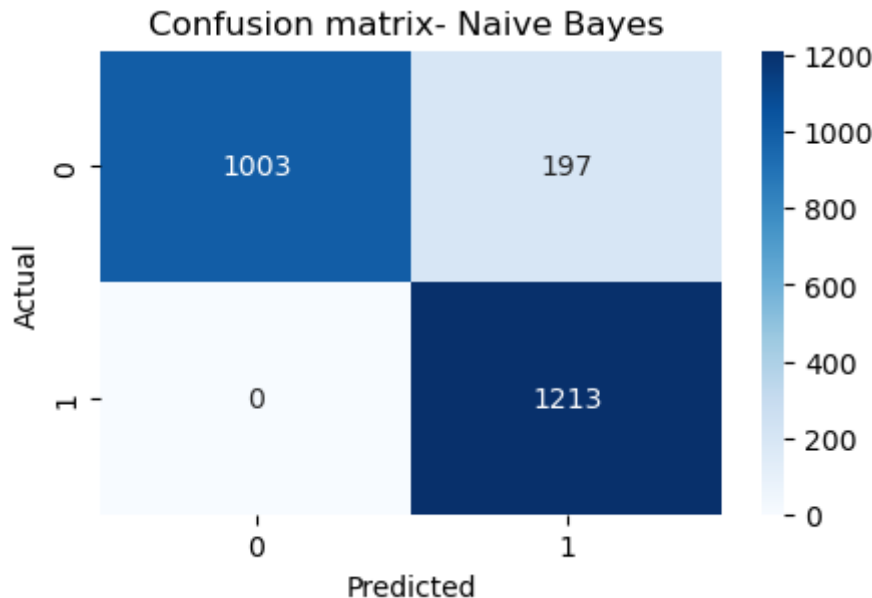
```
train_accuracy_rbf = cross_val_score(svm_rbf,x_train, y_train, cv=10)
crossval_train_rbf=train_accuracy_rbf.mean()
test_accuracy_rbf = cross_val_score(svm_rbf,x_test, y_test, cv=10)
crossval_test_rbf=test_accuracy_rbf.mean()
print('Rbf- SVM after Cross validation Train accuracy:', crossval_train_rbf)
print('-----'*5)
print('Rbf- SVM after Cross validation Test accuracy:', crossval_test_rbf)
```

Rbf- SVM after Cross validation Train accuracy: 0.9951636444220291

-
Rbf- SVM after Cross validation Test accuracy: 0.9788518912245807

In [42]:

```
plt.figure(figsize=(5,3))
sns.heatmap(confusion_matrix(y_test,y_pred_test_nb),cmap='Blues',annot=True, fmt='g')
plt.title("Confusion matrix- Naive Bayes ")
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.show()
```



Combining all models in Tabular form

In [43]:

```
Models=['KNN','Random_forest','Gaussian_Naive_bayes','Multinomial_Naive_bayes','SVM']
Trainacc=[accuracy_knn_train,accuracy_rf_train,accuracy_nb_train,accuracy_mnb_train,accuracy_svm_train]
Testacc=[accuracy_knn_test,accuracy_rf_test,accuracy_nb_test,accuracy_mnb_test,accuracy_svm_test]
Cross_val_train=[crossval_train_knn, crossval_train_rf,crossval_train_nb,crossval_train_mnb,crossval_train_svm]
Cross_val_test=[crossval_test_knn,crossval_test_rf,crossval_test_nb,crossval_test_mnb,crossval_test_svm]
```

In [44]:

```
Combined_accuracy=pd.DataFrame({'Model name':Models,'Train Accuracy':Trainacc,'Test Accuracy':Testacc,
                                'CV Train acc':Cross_val_train,'CV Test acc':Cross_val_test})
print(Combined_accuracy)
```

	Model name	Train Accuracy	Test Accuracy	CV Train acc	CV Test acc
0	KNN	0.998895	0.993369	0.989499	0.926234
1	Random_forest	1.000000	0.998757	0.998204	0.987560
2	Gaussian_Naive_bayes	0.949150	0.918359	0.920409	0.886019
3	Multinomial_Naive_bayes	0.981899	0.968918	0.977063	0.965589
4	SVM	0.998204	0.995856	0.995164	0.978852

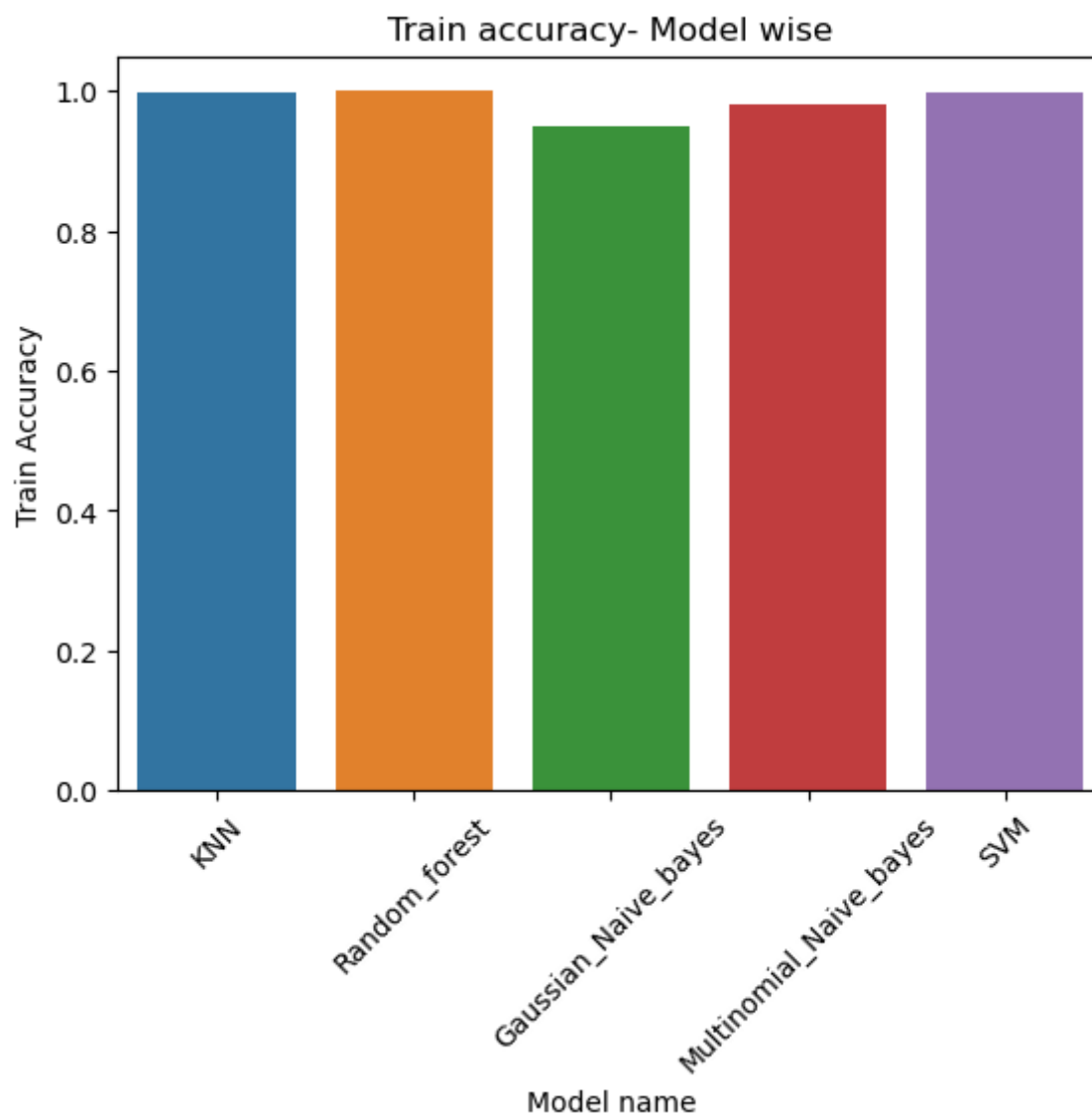
CV Test acc

0	0.926234
1	0.987560
2	0.886019
3	0.965589
4	0.978852

Accuracy Visualization

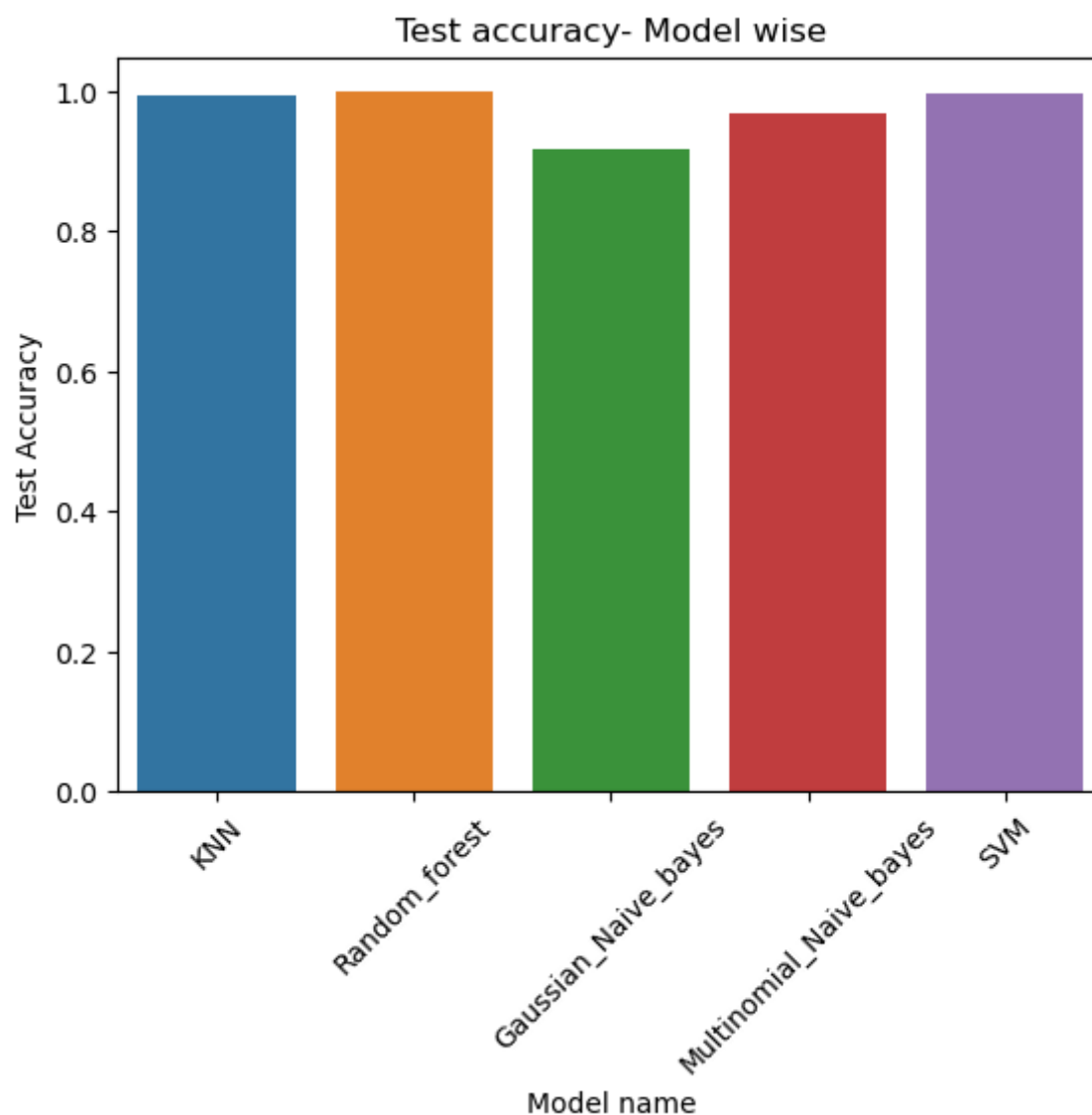
In [45]:

```
sns.barplot(x='Model name',y='Train Accuracy',data=Combined_accuracy)
plt.xticks(rotation=45)
plt.title('Train accuracy- Model wise')
plt.show()
```



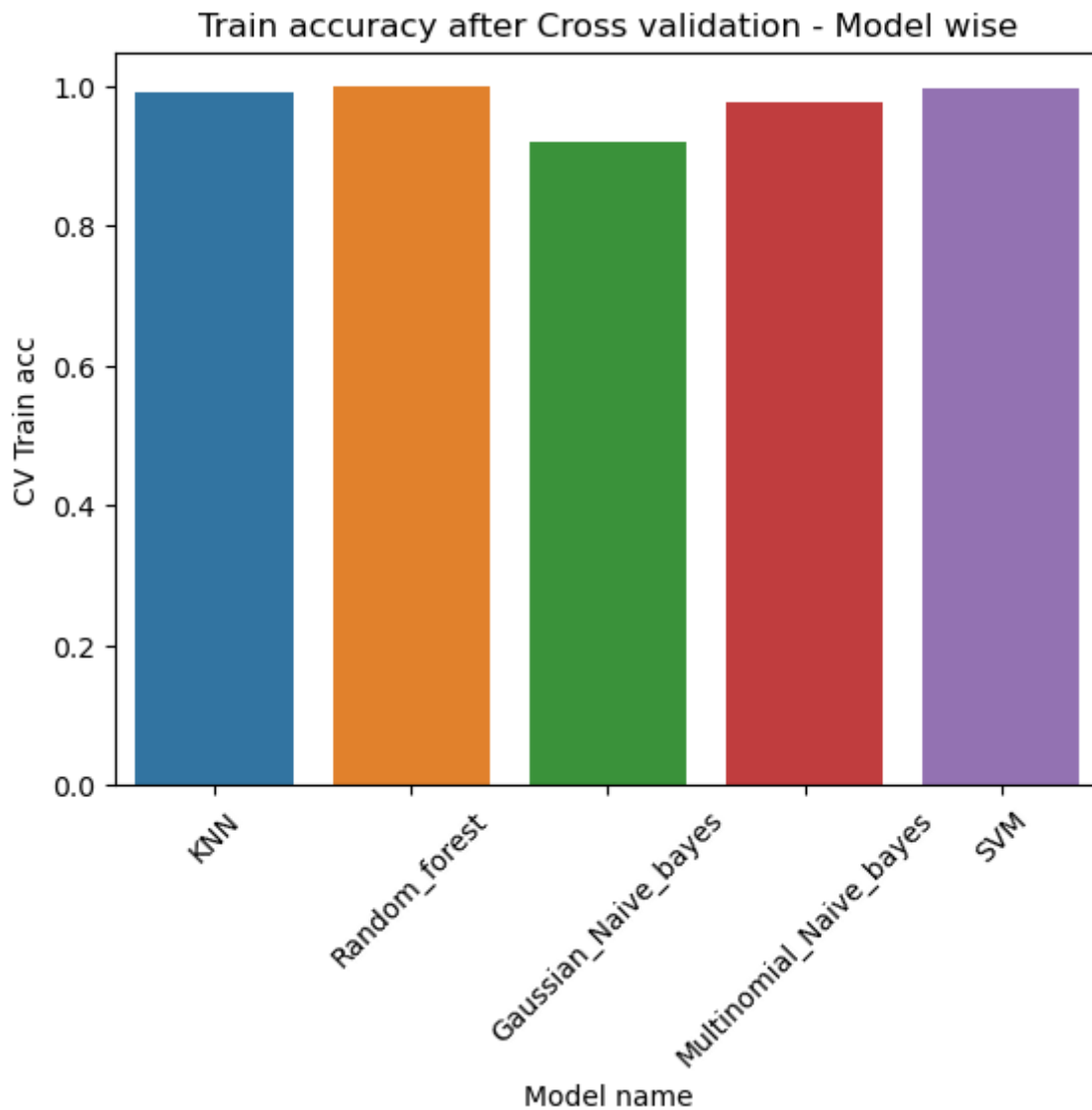
In [46]:

```
sns.barplot(x='Model name',y='Test Accuracy',data=Combined_accuracy)
plt.xticks(rotation=45)
plt.title('Test accuracy- Model wise')
plt.show()
```



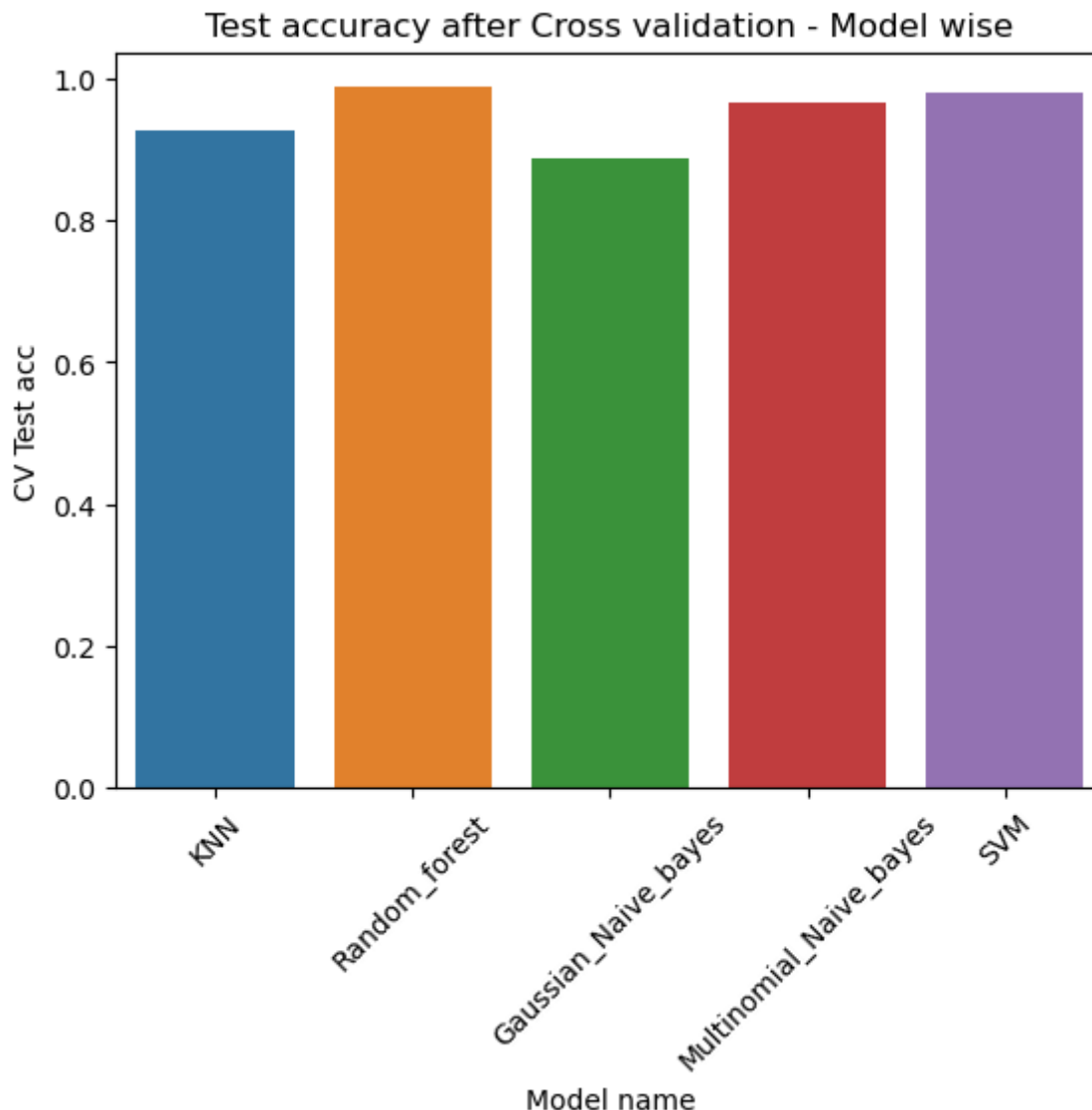
In [48]:

```
sns.barplot(x='Model name',y='CV Train acc',data=Combined_accuracy)
plt.xticks(rotation=45)
plt.title('Train accuracy after Cross validation - Model wise')
plt.show()
```



In [49]:

```
sns.barplot(x='Model name',y='CV Test acc',data=Combined_accuracy)
plt.xticks(rotation=45)
plt.title('Test accuracy after Cross validation - Model wise')
plt.show()
```



Conclusion

- I used 5 Supervised machine learning models to solve the classification problem performed on 5572 observations of the dataset.
- The objective is to build a model that can classify SMS messages as spam or legitimate.
- Random Forest classifier with Train accuracy at 100% and Test accuracy at 99% gives the highest accuracy amongst all the models but it has overfitting issue.
- So after cross validation to deal with overfitting issue of Random Forest model train accuracy is 99% and Test accuracy is 98% making it the best model to solve the classification problem for this dataset.
- Support Vector Machine model follows very closely.
- Gaussian Naive Bayes had the lowest accuracy as compared to rest of the models.
- We can safely say that all the models have the accuracy of more than 90% making all of them ideal for classification of this dataset.

In []: