$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Recall : Recall can be defined as the ratio of the total number of correctly classified positive examples divide to the total number of positive examples. High Recall indicates the class is correctly recognized (small number of FN). Recall is given by the relation –

$$\textit{Recall} = \frac{TP}{TP + FN}$$

Precision : To get the value of precision we divide the total number of correctly classified positive examples by the total number of predicted positive examples. High Precision indicates an example labeled as positive is indeed positive (small number of FP). Precision is given by the relation –

$$\textit{Precision} = \frac{TP}{TP + FP}$$

Implementation steps:

1. **Download Anaconda latest version from its official site. and configure it.**
2. **Open Spyder IDE**
3. **Create a project folder and copy the dataset into the folder (indiadata.csv)**
4. **Importing dataset in python using pandas**

```
import pandas as pd

df1 = pd.read_csv("indiadata.csv",engine='python',usecols = ['gname'])

df = pd.read_csv("indiadata.csv",skipinitialspace=True, usecols =
['city','longitude','latitude','attacktype1_txt','targtype1_txt'])
```

Index	city	latitude	longitude	attacktype1_txt	targtype1_txt
34	Amarpur	23.5195	91.6542	Armed Assault	Private Citizens & Property
37	Siala	33.7782	76.5762	Armed Assault	Private Citizens & Property
41	Vizianagaram district	18.1067	83.3956	Armed Assault	Police
46	Chawmanu	23.8515	91.9955	Armed Assault	Violent Political Party
47	Unknown	23.9408	91.9882	Armed Assault	Violent Political Party
48	Unknown	26.2753	92.9534	Armed Assault	Private Citizens & Property
49	Dantewada District	18.9	81.35	Bombing/Explosion	Police
51	Sonitpor district	26.6788	92.8577	Armed Assault	Private Citizens & Property
53	Athiabari	26.398	90.2694	Armed Assault	Private Citizens & Property
55	Patacharkuchi	26.5131	91.2574	Armed Assault	Police
59	Tinsukia district	27.4919	95.3478	Bombing/Explosion	Utilities
60	Moranhat	27.1782	94.9167	Armed Assault	Police
61	Nalli	27.4423	94.2584	Bombing/Explosion	Utilities
62	Darrang District	26.459	92.0248	Bombing/Explosion	Transportation
64	Guwahati	26.1792	91.7533	Bombing/Explosion	Police
66	Manoharpur	22.2323	83.6854	Armed Assault	Private Citizens & Property
67	Allahabad	25.4358	81.8463	Unarmed Assault	Religious Figures/Institutions
68	Narayanpur	19.7602	78.705	Armed Assault	Private Citizens & Property
70	Bhimpura	25.0776	84.9148	Armed Assault	Private Citizens & Property
72	Haflong	25.1635	93.0136	Armed Assault	Police
73	Sinari	25.0961	85.3131	Armed Assault	Private Citizens & Property

Format Resize Background color Column min/max OK

1)imported data (independant labels).

Index	gname
31	Bodo Liberation Tigers (BLT)
34	National Liberation Front of Tripura (NLFT)
37	Hizbul Mujahideen (HM)
41	People's War Group (PWG)
46	National Liberation Front of Tripura (NLFT)
47	National Liberation Front of Tripura (NLFT)
48	Bodo Liberation Tigers (BLT)
49	Naxalites
51	National Democratic Front of Bodoland (NDFB)
53	Bodo Liberation Tigers (BLT)
55	Bodo Liberation Tigers (BLT)
59	United Liberation Front of Assam (ULFA)
60	United Liberation Front of Assam (ULFA)
61	United Liberation Front of Assam (ULFA)
62	United Liberation Front of Assam (ULFA)
64	United Liberation Front of Assam (ULFA)
66	Vishwa Hindu Parishad (VHP)
67	Vishwa Hindu Parishad (VHP)
68	Ranbir Sena
70	Mazdoor Kisan Sangram Samiti (MKSS)
72	Other

Format Resize Background color Column min/max

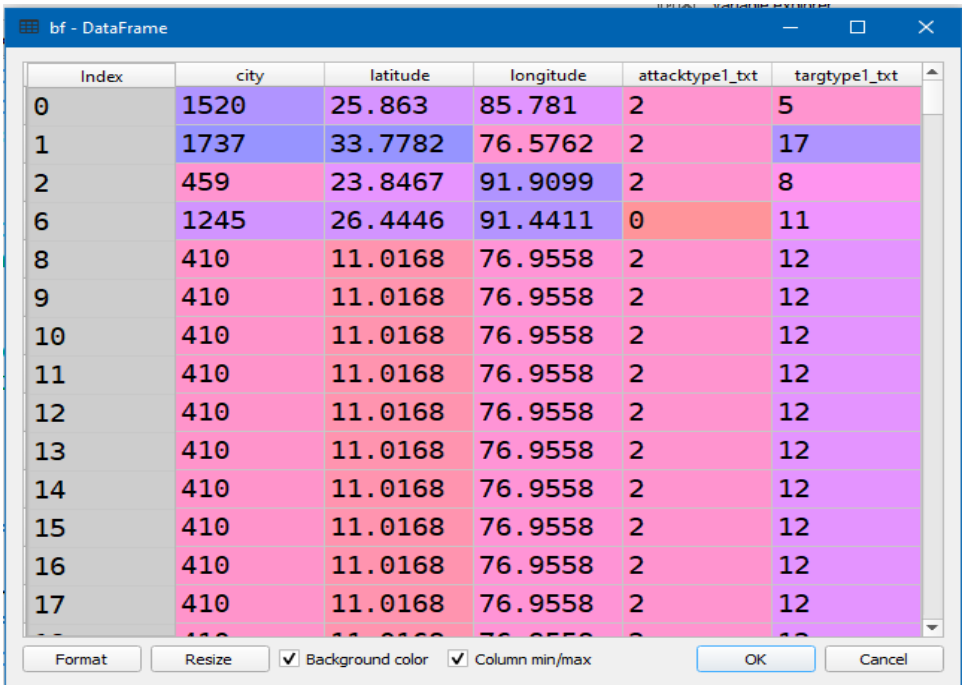
2)Imported data (dependant label)

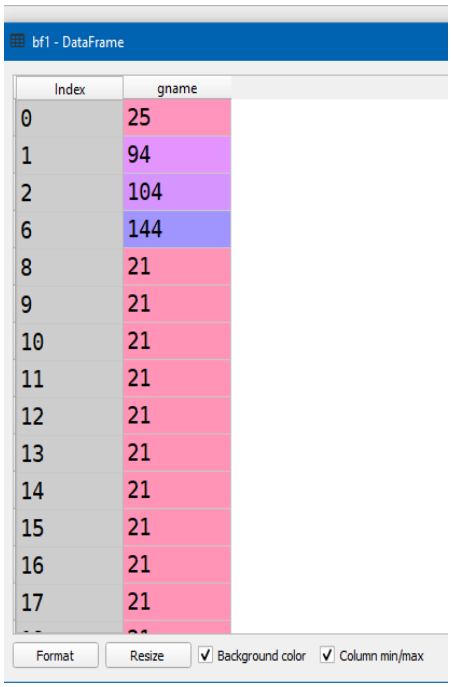
5.Data preprocessing removing Null record and handling categorical data

```
import pandas as pd
df = df[df1.gname != 'Unknown']                #removing unknown names record from both data frames
df1 = df1[df1.gname != 'Unknown']
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df1['gname']= le.fit_transform(df1['gname'])
df['city']= le.fit_transform(df['city'])
df['attacktype1_txt']= le.fit_transform(df['attacktype1_txt'])
df['targtype1_txt']= le.fit_transform(df['targtype1_txt'])
```

To convert categorical Data into numeric data using labelencoder.





6. Spliting the data into training and testing sets

```
from sklearn.model_selection import train_test_split

X_train , X_test ,y_train,y_test = train_test_split(bf,bf1,test_size = 0.3 , random_state = 0 )
```

The dataset is split in ratio of 0.3 means 70% training set and 30% testing set.

7. Creating classification models

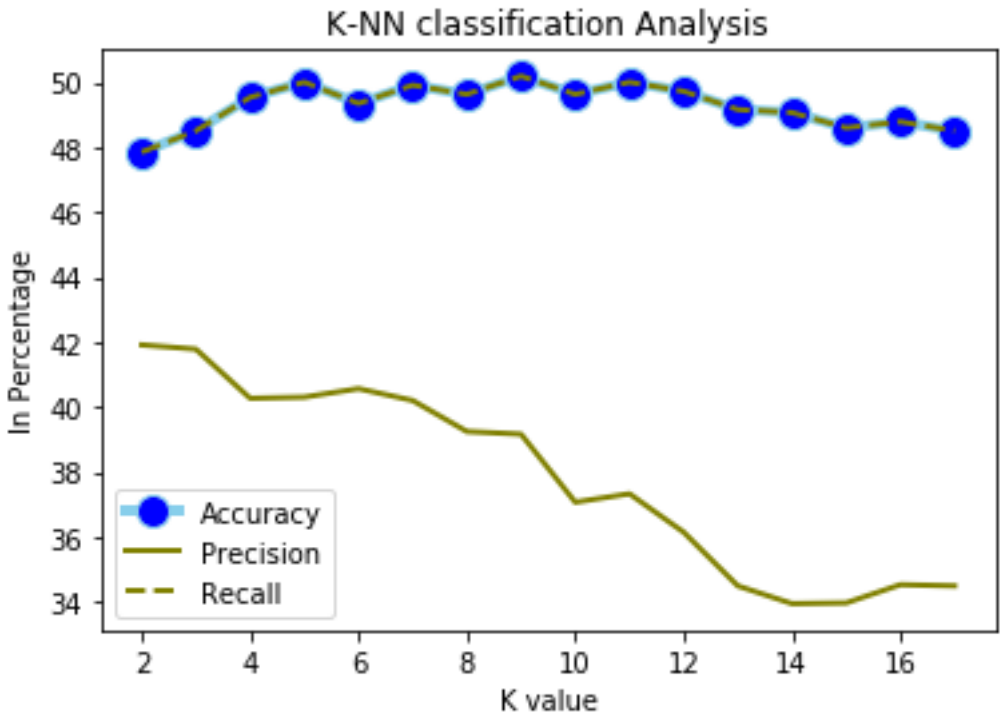
- **KNN classifier**

Creating classifier

```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=k)

model.fit(X_train,y_train)

predictedknn= model.predict(X_test)
```

Find the Best possible K calculating based on accuracy and precision.

For value k =2 to k =18.



- **Decision Tree :**
  Creating classifier :

```
from sklearn.tree import DecisionTreeClassifier

declf = DecisionTreeClassifier()
```

```
declf = declf.fit(X_train,y_train)

predictedDT= declf.predict(X_test)
```

- **Naïve Bayes :**
  Creating classifier :

```
from sklearn.naive_bayes import GaussianNB

modelNB = GaussianNB()

modelNB.fit(X_train,y_train)

predictedNB= modelNB.predict(X_test)
```

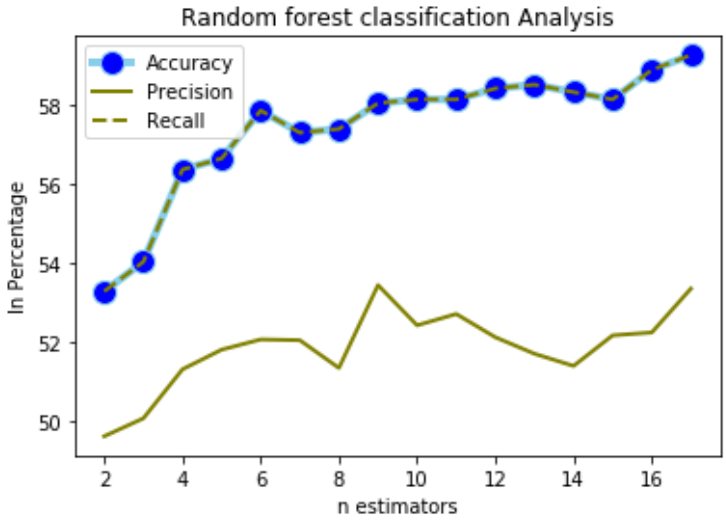
- **Support Vector Machine :**
  Creating Classifier model

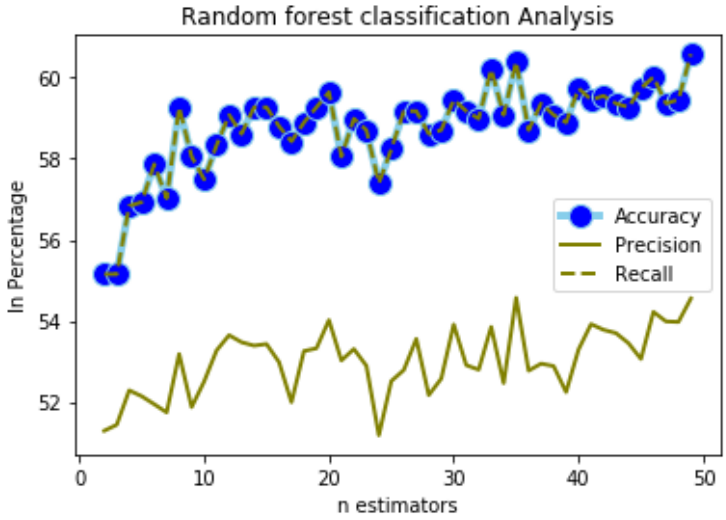
```
from sklearn.svm import SVC

clf = SVC(gamma='auto')

clf.fit(X_train, y_train)

predictedSVC = clf.predict(X_test)
```

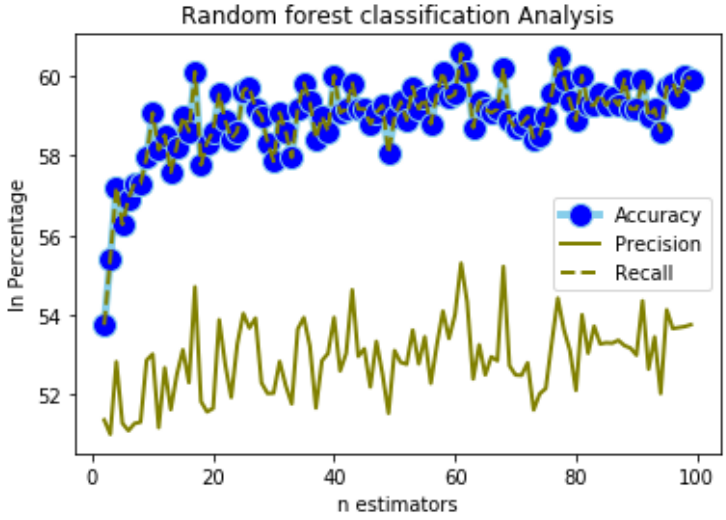
- **Random Forest :**
  Creating Classifier model

```
from sklearn.ensemble import RandomForestClassifier

forestclass=RandomForestClassifier(n_estimators=50)

forestclass.fit(X_train,y_train)

predictedRF=forestclass.predict(X_test)
```

Finding the best n-estimators for random forest :







8.Calculating Accuracy , Precision and Recall .

```
from sklearn import metrics

from sklearn.metrics import recall_score

from sklearn.metrics import precision_score

metrics.accuracy_score(y_test,predicted)*100,

precision_score(y_test,predicted,average='weighted')*100,

recall_score(y_test,predicted,average='weighted')*100,
```

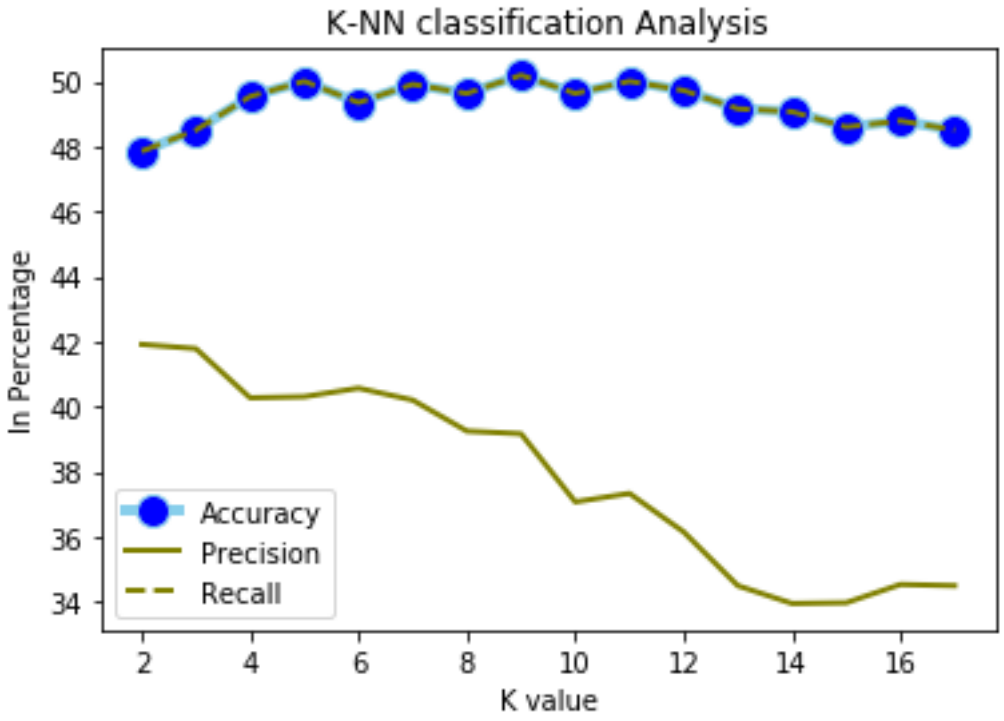
Method	Accuracy	Precision	Recall
KNN	50.00%	40.03%	50.00%
Decision Tree	53.18%	53.33%	53.18%
Naïve bayes	42.13%	47.76%	42.13%
SVM	49.81%	43.08%	49.81%
Random Forest	58.98%	52.33%	58.98%

9 Data Visualization

- **Data visualization of KNN analysis**

```
pldf=pd.DataFrame({'x': range(2,18)})

pldf['Accuracy'] =accuracy

pldf['Precision'] =pre

pldf['Recall'] =recall

# multiple line plot

plt.title('K-NN classification Analysis')

plt.xlabel('K value', fontsize=10)

plt.ylabel('In Percentage', fontsize=10)
```

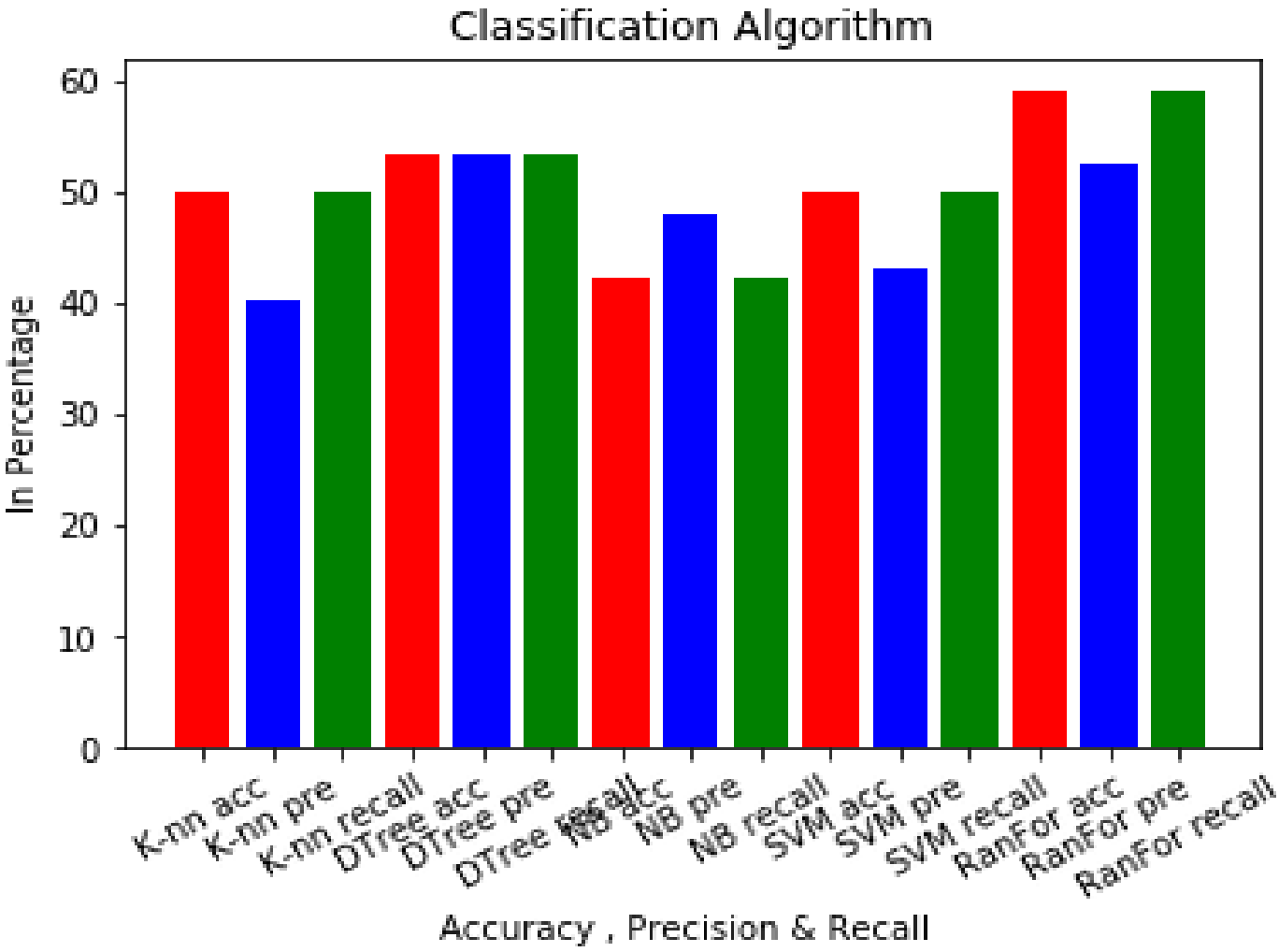
```
plt.plot( 'x', 'Accuracy', data=pldf, marker='o', markerfacecolor='blue', markersize=12, color='skyblue', linewidth=4)

plt.plot( 'x', 'Precision', data=pldf, marker='', color='olive', linewidth=2)

plt.plot( 'x', 'Recall', data=pldf, marker='', color='olive', linewidth=2, linestyle='dashed', label="Recall")

plt.savefig('report2.png')

plt.legend()
```



- **Data visualization for comparing classification algorithm**

```
import matplotlib.pyplot as plt

index = np.arange(len(label))

plt.bar(index,acc, color=['red','blue','green','red','blue','green','red','blue','green'])

plt.xlabel('Accuracy , Precision & Recall ', fontsize=10)

plt.ylabel('In Percentage', fontsize=10)
```

```
plt.xticks(index, label, fontsize=10, rotation=30)

plt.title('Classification Algorithm')

plt.show()
```



Conclusion: In this way using python we have performed preprocessing on a dataset and analyzed confusion matrix for different classifier models.