```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.cm as cm
%matplotlib inline
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

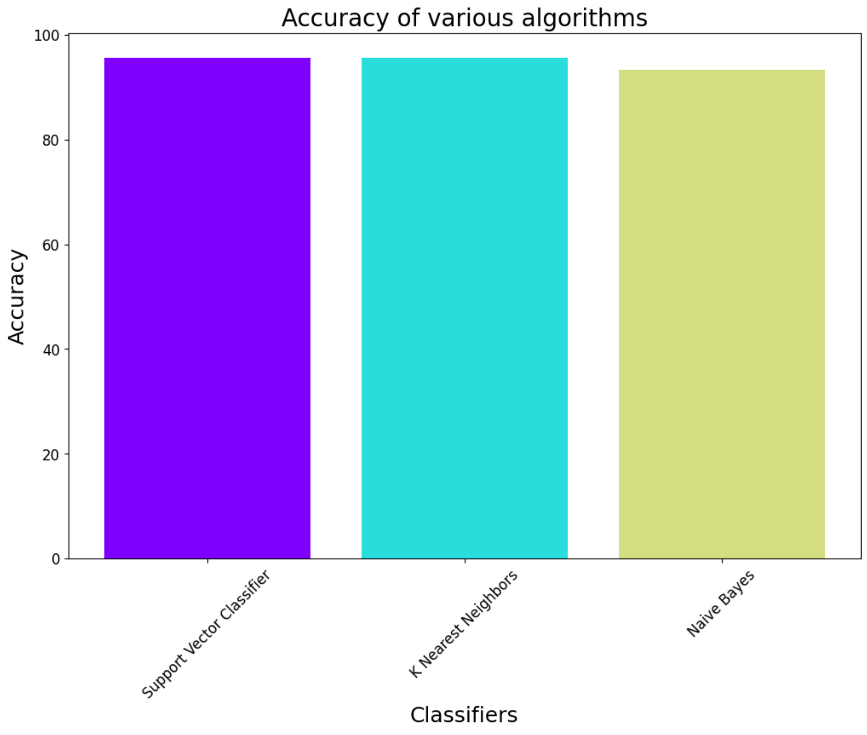
```
data=pd.read_csv("/content/Iris.csv")
#Training a Linear Regression Model
X = data[['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm']]
y = data['Species']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

```
((105, 4), (45, 4), (105,), (45,))
```

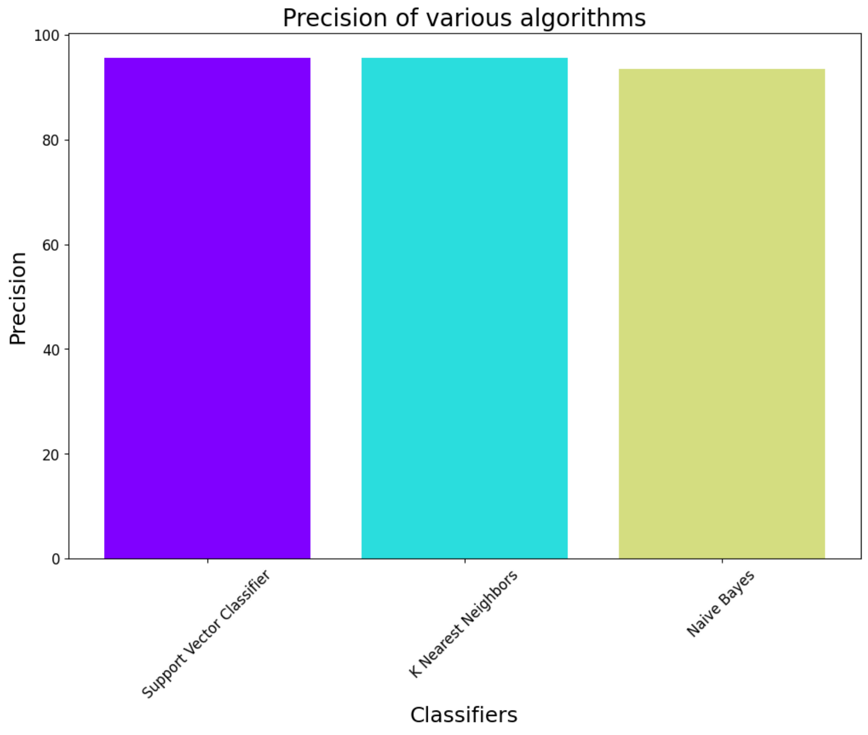
```
plt.figure(figsize=(12,8))
colors = cm.rainbow(np.linspace(0, 1, 4))
labels = ['Support Vector Classifier', 'K Nearest Neighbors','Naive Bayes' ]
plt.bar(labels,
accuracy_scores,
color = colors)
plt.xlabel('Classifiers',fontsize=18)
plt.ylabel('Accuracy',fontsize=18)
plt.title('Accuracy of various algorithms',fontsize=20)
plt.xticks(rotation=45,fontsize=12)
plt.yticks(fontsize=12)
```

```
(array([  0.,  20.,  40.,  60.,  80., 100., 120.]),
 [Text(0, 0.0, '0'),
  Text(0, 20.0, '20'),
  Text(0, 40.0, '40'),
  Text(0, 60.0, '60'),
  Text(0, 80.0, '80'),
  Text(0, 100.0, '100'),
  Text(0, 120.0, '120')])
```



```
plt.figure(figsize=(12,8))
colors = cm.rainbow(np.linspace(0, 1, 4))
labels = ['Support Vector Classifier', 'K Nearest Neighbors','Naive Bayes' ]
plt.bar(labels,
precision_scores,
color = colors)
plt.xlabel('Classifiers',fontsize=18)
plt.ylabel('Precision',fontsize=18)
plt.title('Precision of various algorithms',fontsize=20)
plt.xticks(rotation=45,fontsize=12)
plt.yticks(fontsize=12)
```

```
(array([  0.,  20.,  40.,  60.,  80., 100., 120.]),
 [Text(0, 0.0, '0'),
  Text(0, 20.0, '20'),
  Text(0, 40.0, '40'),
  Text(0, 60.0, '60'),
  Text(0, 80.0, '80'),
  Text(0, 100.0, '100'),
  Text(0, 120.0, '120')])
```



```
X_train[0:5]
```

	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm
12	4.8	3.0	1.4	0.1
82	5.8	2.7	3.9	1.2
28	5.2	3.4	1.4	0.2
14	5.8	4.0	1.2	0.2
17	5.1	3.5	1.4	0.3

```
y_train[0:5]
```

```
12        Iris-setosa
82    Iris-versicolor
28        Iris-setosa
14        Iris-setosa
17        Iris-setosa
Name: Species, dtype: object
```

```
accuracy_scores = np.zeros(3)
precision_scores = np.zeros(3)
recall_scores = np.zeros(3)
f1_scores = np.zeros(3)
```

```
# Support Vector Classifier
clf = SVC().fit(X_train, y_train)
prediction = clf.predict(X_test)
accuracy_scores[0] = accuracy_score(y_test, prediction)*100
print('Support Vector Classifier accuracy: {}%'.format(accuracy_scores[0]))
precision_scores[0]= precision_score(y_test, prediction, average='macro')*100
print('Support Vector Classifier precision: {}%'.format(precision_scores[0]))
recall_scores[0]=recall_score(y_test, prediction, average='macro')*100
print('Support Vector Classifier recall: {}%'.format(recall_scores[0]))
f1_scores[0]=f1_score(y_test, prediction, average='macro')*100
print('Support Vector Classifier f1: {}%'.format(f1_scores[0]))
```

```
    Support Vector Classifier accuracy: 95.55555555555556%
    Support Vector Classifier precision: 95.53571428571429%
    Support Vector Classifier recall: 95.53571428571429%
    Support Vector Classifier f1: 95.53571428571429%
```

```
# K Nearest Neighbors
clf = KNeighborsClassifier().fit(X_train, y_train)
prediction = clf.predict(X_test)
accuracy_scores[1] = accuracy_score(y_test, prediction)*100
print('K Nearest Neighbors Classifier accuracy: {}%'.format(accuracy_scores[1]))
precision_scores[1]= precision_score(y_test, prediction, average='macro')*100
print('Support Vector Classifier precision: {}%'.format(precision_scores[1]))
recall_scores[1]=recall_score(y_test, prediction, average='macro')*100
print('Support Vector Classifier precision: {}%'.format(recall_scores[1]))
f1_scores[1]=f1_score(y_test, prediction, average='macro')*100
print('Support Vector Classifier f1: {}%'.format(f1_scores[1]))
```

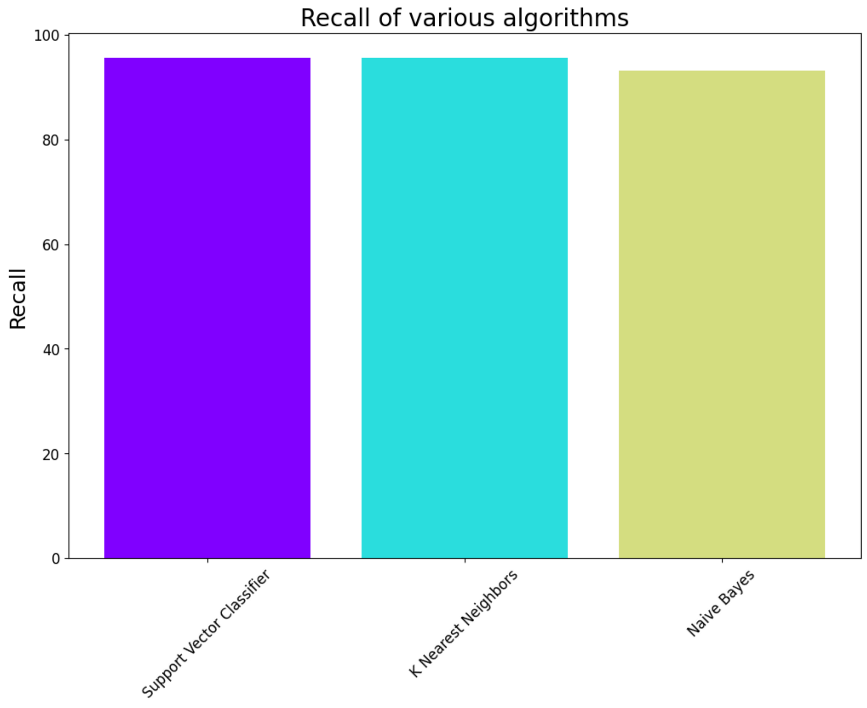
```
    K Nearest Neighbors Classifier accuracy: 95.55555555555556%
    Support Vector Classifier precision: 95.53571428571429%
    Support Vector Classifier precision: 95.53571428571429%
    Support Vector Classifier f1: 95.53571428571429%
```

```
# Naive Baye's Neighbors
clf = GaussianNB().fit(X_train, y_train)
prediction = clf.predict(X_test)
accuracy_scores[2] = accuracy_score(y_test, prediction)*100
print('Naive Bayes (NB) accuracy: {}%'.format(accuracy_scores[2]))
precision_scores[2]= precision_score(y_test, prediction, average='macro')*100
print('Support Vector Classifier precision: {}%'.format(precision_scores[2]))
recall_scores[2]=recall_score(y_test, prediction, average='macro')*100
print('Support Vector Classifier precision: {}%'.format(recall_scores[2]))
f1_scores[2]=f1_score(y_test, prediction, average='macro')*100
print('Support Vector Classifier f1l: {}%'.format(f1_scores[2]))
```

```
    Naive Bayes (NB) accuracy: 93.33333333333333%
    Support Vector Classifier precision: 93.51432880844645%
    Support Vector Classifier precision: 93.15476190476191%
    Support Vector Classifier f1l: 93.26599326599326%
```

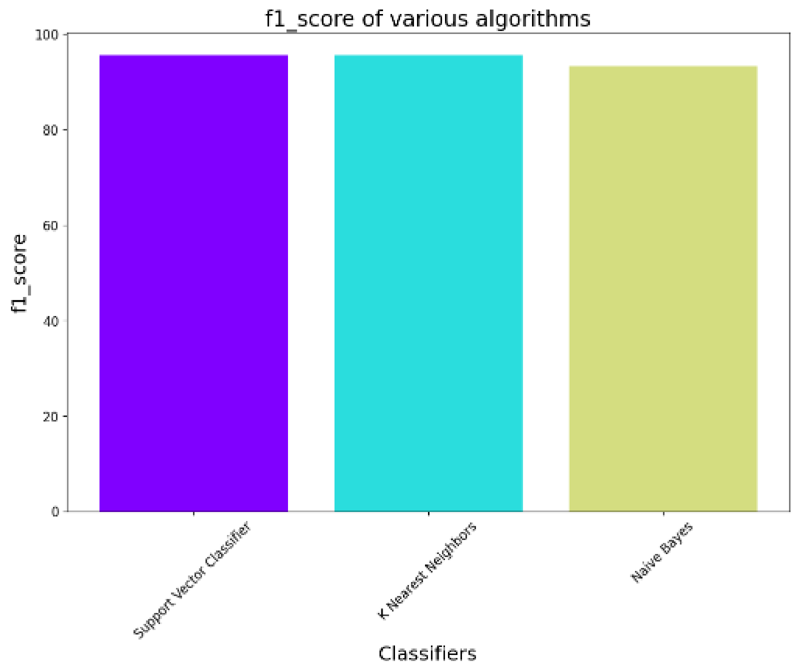
```
plt.figure(figsize=(12,8))
colors = cm.rainbow(np.linspace(0, 1, 4))
labels = ['Support Vector Classifier', 'K Nearest Neighbors','Naive Bayes' ]
plt.bar(labels,
recall_scores,
color = colors)
plt.xlabel('Classifiers',fontsize=18)
plt.ylabel('Recall',fontsize=18)
plt.title('Recall of various algorithms',fontsize=20)
plt.xticks(rotation=45,fontsize=12)
plt.yticks(fontsize=12)
```

```
(array([  0.,   20.,  40.,  60.,  80., 100., 120.]),
 [Text(0, 0.0, '0'),
  Text(0, 20.0, '20'),
  Text(0, 40.0, '40'),
  Text(0, 60.0, '60'),
  Text(0, 80.0, '80'),
  Text(0, 100.0, '100'),
  Text(0, 120.0, '120')])
```



```
plt.figure(figsize=(12,8))
colors = cm.rainbow(np.linspace(0, 1, 4))
labels = ['Support Vector Classifier', 'K Nearest Neighbors','Naive Bayes' ]
plt.bar(labels,
f1_scores,
color = colors)
plt.xlabel('Classifiers',fontsize=18)
plt.ylabel('f1_score',fontsize=18)
plt.title('f1_score of various algorithms',fontsize=20)
plt.xticks(rotation=45,fontsize=12)
plt.yticks(fontsize=12)
```

```
(array([  0.,   20.,  40.,  60.,  80., 100., 120.]),
 [Text(0, 0.0, '0'),
  Text(0, 20.0, '20'),
  Text(0, 40.0, '40'),
  Text(0, 60.0, '60'),
  Text(0, 80.0, '80'),
  Text(0, 100.0, '100'),
  Text(0, 120.0, '120')])
```


f1_score of various algorithms
f1_score
Support Vector Classifier
K Nearest Neighbors
Naive Bayes
Classifiers