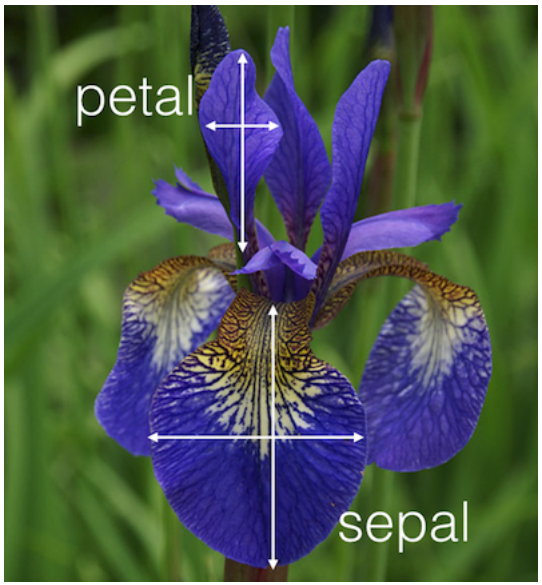
KNN (K Nearest Neighbors) Classification

```
x
```

```
     array([0])
```

```
## Loading the necessary packages

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from sklearn.datasets import load_iris
iris = load_iris()
%matplotlib inline
```




```
## Displaying the feature names

iris.feature_names
```

```
     ['sepal length (cm)',
      'sepal width (cm)',
      'petal length (cm)',
      'petal width (cm)']
```

```
## Displaying the class labels

iris.target_names
```

```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

```
## Creating a dataframe out of the dataset loaded

df = pd.DataFrame(iris.data,columns=iris.feature_names)
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
## Adding class labels to the dataframe

df['target'] = iris.target
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

```
df['flower_name'] = df.target.apply(lambda x: iris.target_names[x])
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
0	5.1	3.5	1.4	0.2	0	setosa
1	4.9	3.0	1.4	0.2	0	setosa
2	4.7	3.2	1.3	0.2	0	setosa
3	4.6	3.1	1.5	0.2	0	setosa
4	5.0	3.6	1.4	0.2	0	setosa

```
df.tail()
```

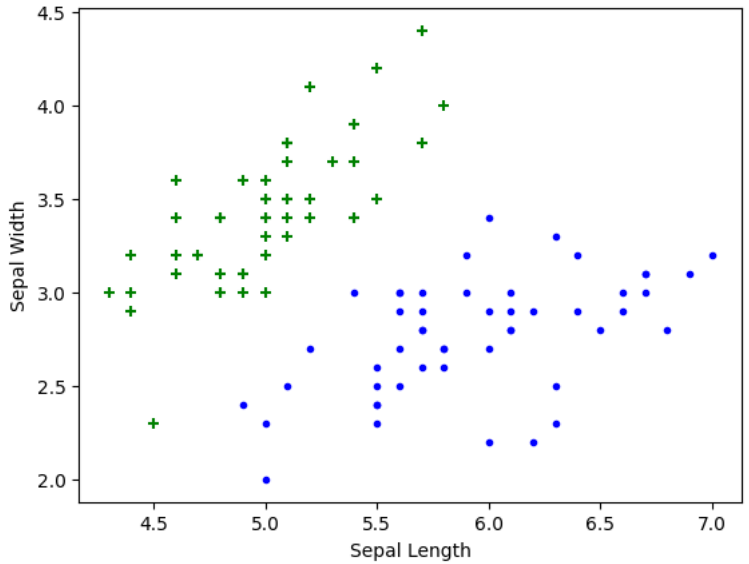
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	flower_name
145	6.7	3.0	5.2	2.3	2	virginica
146	6.3	2.5	5.0	1.9	2	virginica
147	6.5	3.0	5.2	2.0	2	virginica

```
## Creating three dataframes corresponding to three class labels from the main dataframe df

df0 = df[:50]
df1 = df[50:100]
df2 = df[100:]
```

Sepal length vs Sepal Width (Setosa vs Versicolor)

```
## Visulaizing the instances of Setosa and Versicolor wrt. Sepal Length & Width

plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.scatter(df0['sepal length (cm)'], df0['sepal width (cm)'],color="green",marker='+')
plt.scatter(df1['sepal length (cm)'], df1['sepal width (cm)'],color="blue",marker='.')
```

```
<matplotlib.collections.PathCollection at 0x7e1628b84e20>
```

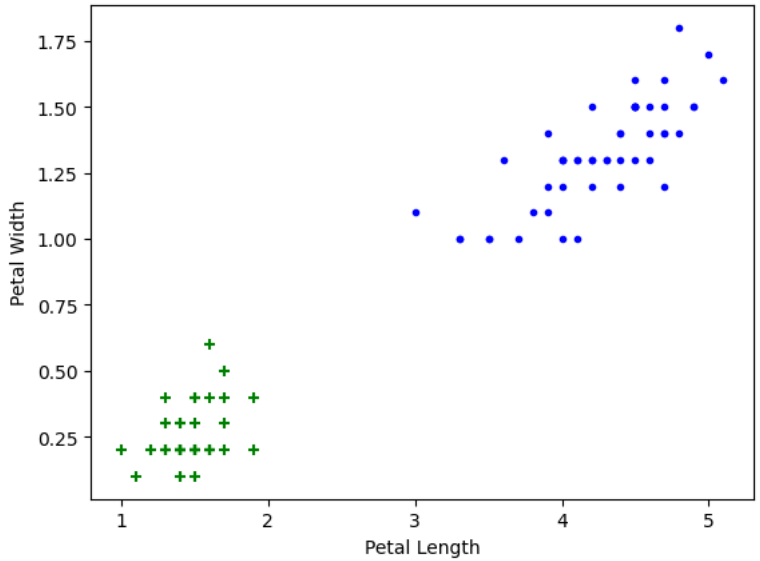


Petal length vs Petal Width (Setosa vs Versicolor)

```
## Visulaizing the instances of Setosa and Versicolor wrt. Sepal Length & Width

plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
```

```
plt.scatter(df0['petal length (cm)'], df0['petal width (cm)'],color="green",marker='+')
plt.scatter(df1['petal length (cm)'], df1['petal width (cm)'],color="blue",marker='.')
```

```
<matplotlib.collections.PathCollection at 0x7e1628c0ab60>
```



Train test split

```
from sklearn.model_selection import train_test_split

X = df.drop(['target','flower_name'], axis='columns')
y = df.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

print(len(X_train), len(X_test))
```

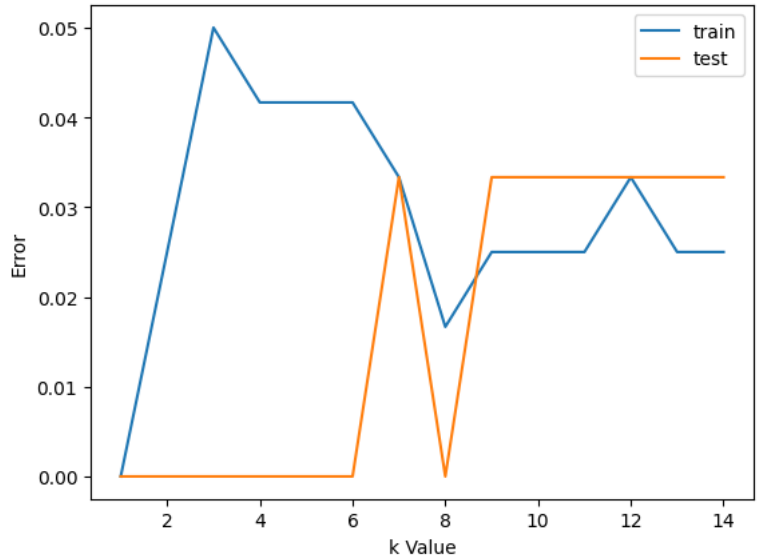
```
120 30
```

Create KNN (K Neighrest Neighbour Classifier)

```
## Finetuning the hyperparameter K

from sklearn.neighbors import KNeighborsClassifier

error1= []
error2= []
for k in range(1,15):
    knn= KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train,y_train)
```

```
    y_pred1= knn.predict(X_train)
    error1.append(np.mean(y_train!= y_pred1))
    y_pred2= knn.predict(X_test)
    error2.append(np.mean(y_test!= y_pred2))

# plt.figure(figsize(20,10))
plt.plot(range(1,15),error1,label="train")
plt.plot(range(1,15),error2,label="test")
plt.xlabel('k Value')
plt.ylabel('Error')
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7e1628e06560>
```




```
## Model Creation

knn = KNeighborsClassifier(n_neighbors=8)

## Model Training

knn.fit(X_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=8)
```

```
knn.score(X_test, y_test)
```

```
1.0
```

```
## Prediction for a sample

x = knn.predict([[4.8,3.0,1.5,0.3]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KNeighborsClassifier was fitted with feature names
  warnings.warn(
```

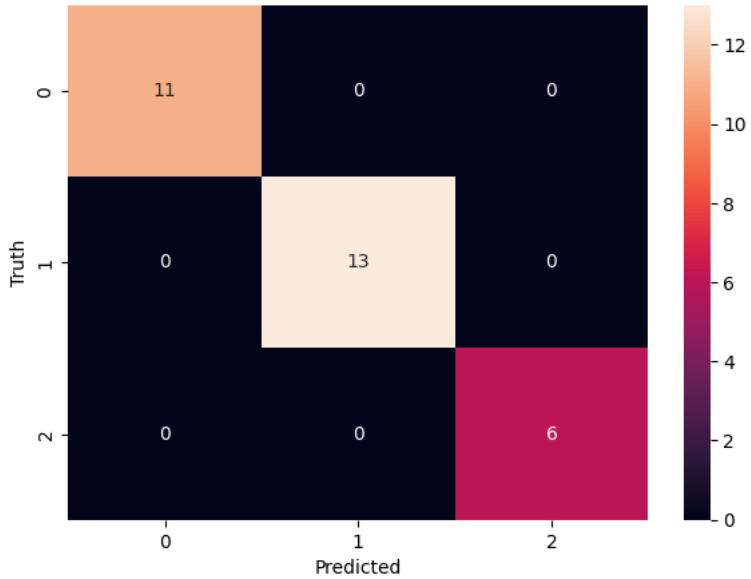
Plot Confusion Matrix

```
from sklearn.metrics import confusion_matrix
y_pred = knn.predict(X_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[11,  0,  0],
       [ 0, 13,  0],
       [ 0,  0,  6]])
```

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sn
plt.figure(figsize=(7,5))
sn.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

```
Text(58.222222222222214, 0.5, 'Truth')
```



Print classification report for precesion, recall and f1-score for each classes

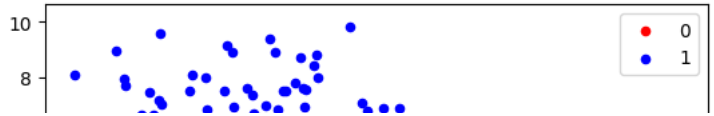
```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
                   precision    recall  f1-score   support

                0       1.00      1.00      1.00        11
                1       1.00      1.00      1.00        13
                2       1.00      1.00      1.00         6

         accuracy                           1.00        30
        macro avg       1.00      1.00      1.00        30
     weighted avg       1.00      1.00      1.00        30
```

Generating Datasets

Blobs Classification Problem

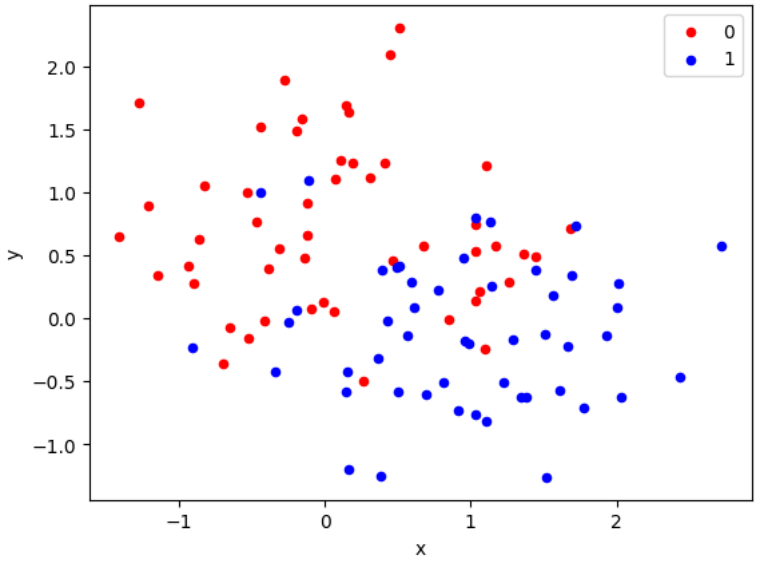
```
from sklearn.datasets import make_blobs
from matplotlib import pyplot
from pandas import DataFrame


# generate 2d classification dataset
X, y = make_blobs(n_samples=100, centers=2, n_features=2)


# scatter plot, dots colored by class value
df = DataFrame(dict(x1=X[:,0], x2=X[:,1], label=y))
colors = {0:'red', 1:'blue'}
fig, ax = pyplot.subplots()
grouped = df.groupby('label')
for key, group in grouped:
    group.plot(ax=ax, kind='scatter', x='x1', y='x2', label=key, color=colors[key])
pyplot.show()
```



Moons Classification Problem

```
from sklearn.datasets import make_moons

# generate 2d classification dataset
X, y = make_moons(n_samples=100, noise=0.5)
```

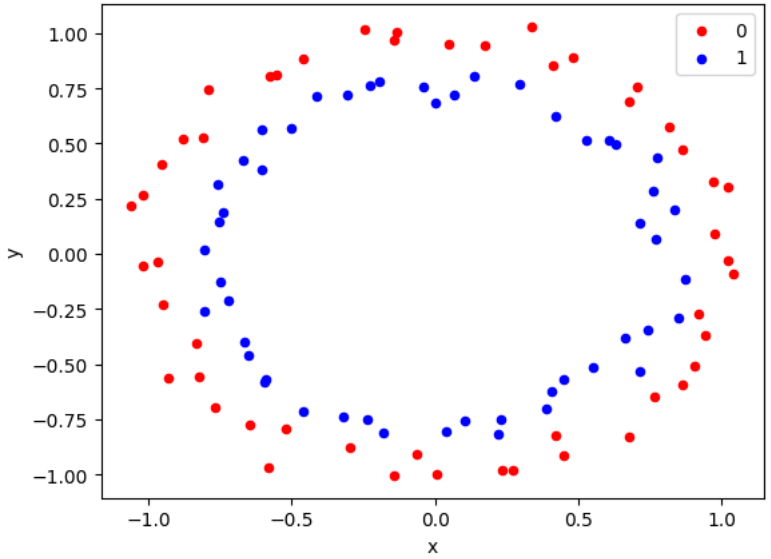
```
# scatter plot, dots colored by class value
df = DataFrame(dict(x=X[:,0], y=X[:,1], label=y))
colors = {0:'red', 1:'blue'}
fig, ax = pyplot.subplots()
grouped = df.groupby('label')
for key, group in grouped:
    group.plot(ax=ax, kind='scatter', x='x', y='y', label=key, color=colors[key])
pyplot.show()
```



Circles Classification Problem

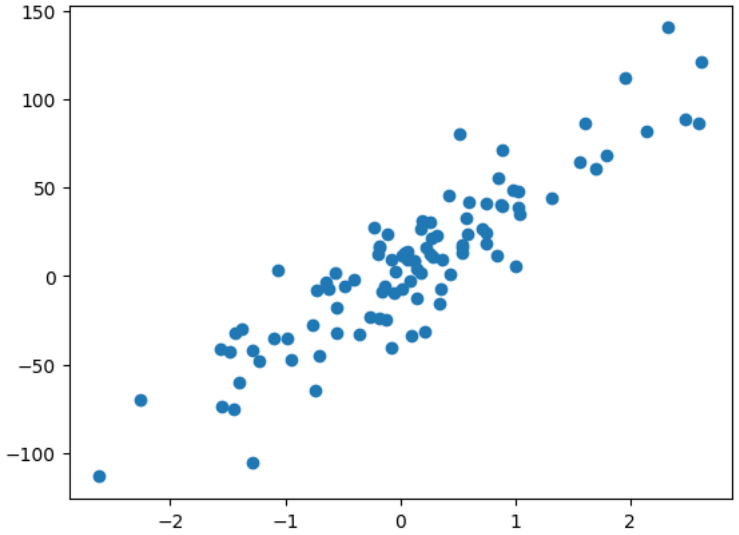
```
from sklearn.datasets import make_circles

# generate 2d classification dataset
X, y = make_circles(n_samples=100, noise=0.05)
```

```
# scatter plot, dots colored by class value
df = DataFrame(dict(x=X[:,0], y=X[:,1], label=y))
colors = {0:'red', 1:'blue'}
fig, ax = pyplot.subplots()
grouped = df.groupby('label')
for key, group in grouped:
    group.plot(ax=ax, kind='scatter', x='x', y='y', label=key, color=colors[key])
```



Regression Test Problems

```
from sklearn.datasets import make_regression

# generate regression dataset
X, y = make_regression(n_samples=100, n_features=1, noise=20)

# plot regression dataset
pyplot.scatter(X,y)
pyplot.show()
```


150
100
50
0
−50
−100
−2
−1
0
1
2