Machine Learning with Scikit-Learn

scikit-learn

Simple and efficient tools for data mining and data analysis

Accessible to everybody, and reusable in various contexts.

Built on **NumPy, SciPy, and matplotlib**

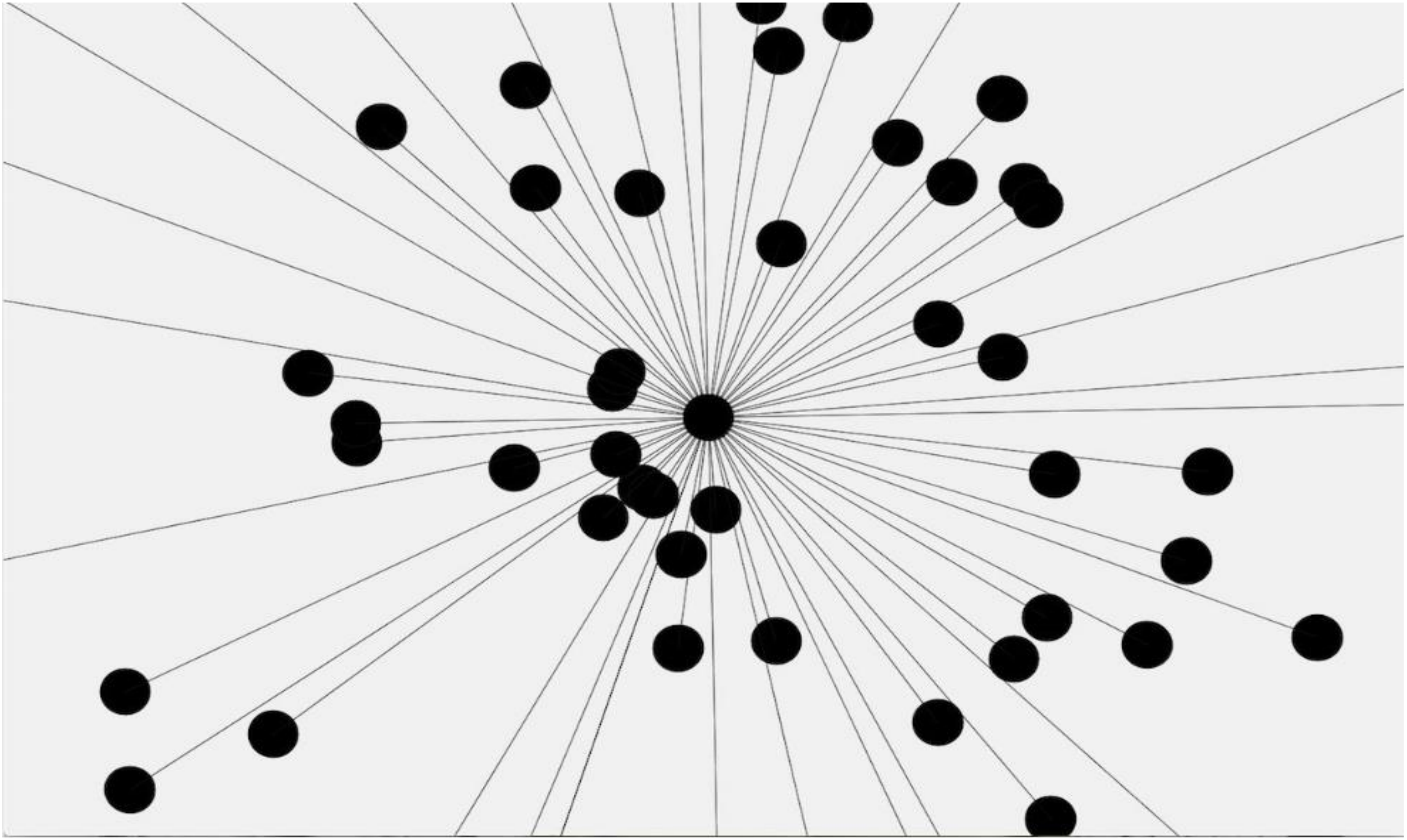
Representation of Data in Scikit-learn

Machine Leraning is all about cretaing models from data.

[n_sample,n_feature]

https://archive.ics.uci.edu/ml/datasets/iris

Nearest Neighbors Classification



sklearn.neighbors.KNeighborsClassifier

n_neighbors :int, optional (default = 5)
Number of neighbors to use

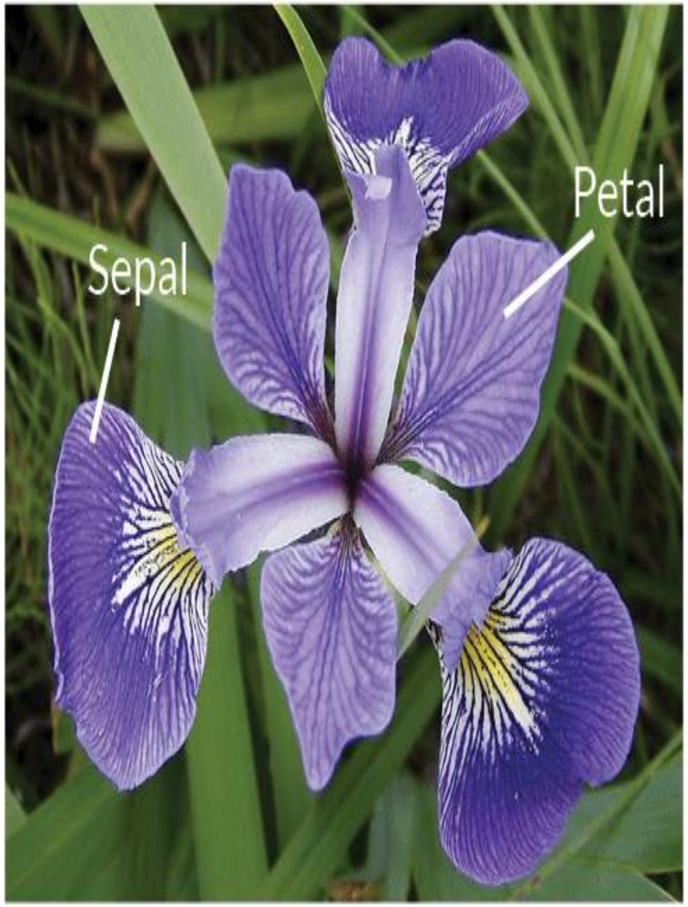
weights : str or callable, optional (default = ‘uniform’)
weight function used in prediction. Possible values:
‘**uniform**’ **:** uniform weights. All points in each neighborhood are weighted equally.
‘distance’ : weight points by the inverse of their distance.

fit(X, y)	Fit the model using X as training data and y as target values
predict(X)	Predict the class labels for the provided data

Iris flower data set

The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor).

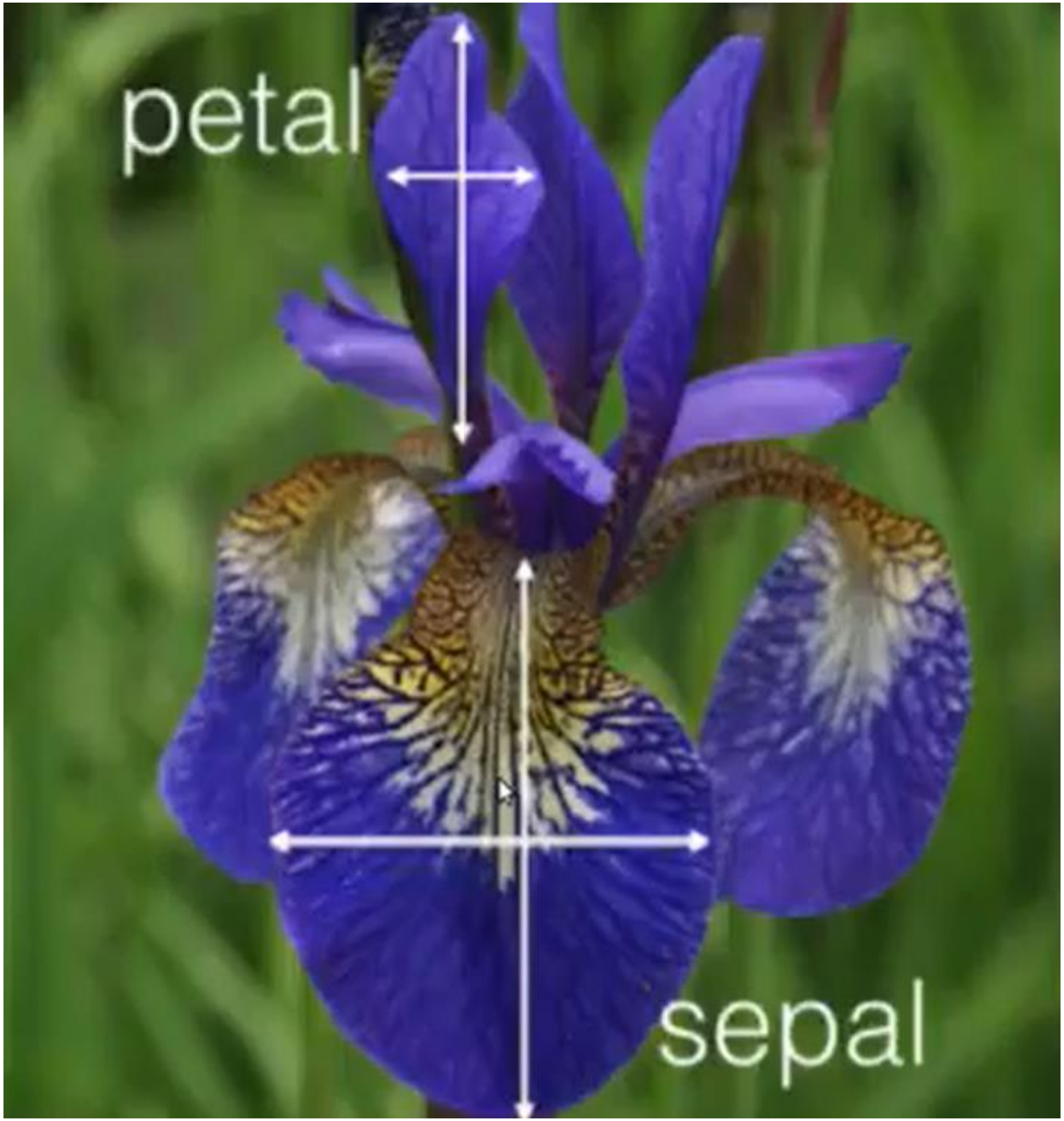
Classes	3
Samples per class	50
Samples total	150
Dimensionality	4
Features	real, positive

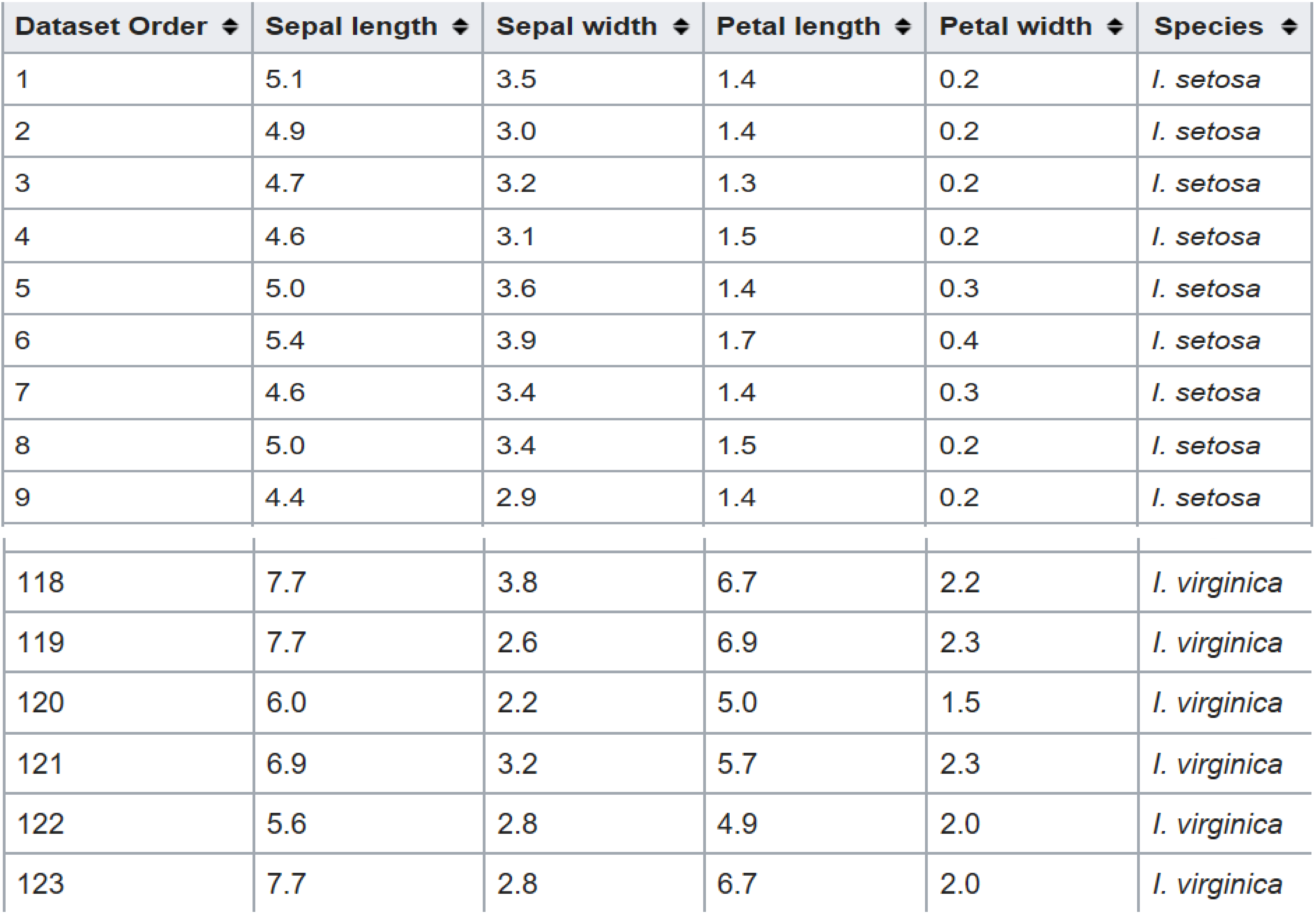


Iris Versicolor

Iris Setosa

Iris Virginica


petal
sepal



Dataset Order	Sepal length	Sepal width	Petal length	Petal width	Species
1	5.1	3.5	1.4	0.2	*I. setosa*
2	4.9	3.0	1.4	0.2	*I. setosa*
3	4.7	3.2	1.3	0.2	*I. setosa*
4	4.6	3.1	1.5	0.2	*I. setosa*
5	5.0	3.6	1.4	0.3	*I. setosa*
6	5.4	3.9	1.7	0.4	*I. setosa*
7	4.6	3.4	1.4	0.3	*I. setosa*
8	5.0	3.4	1.5	0.2	*I. setosa*
9	4.4	2.9	1.4	0.2	*I. setosa*
118	7.7	3.8	6.7	2.2	*I. virginica*
119	7.7	2.6	6.9	2.3	*I. virginica*
120	6.0	2.2	5.0	1.5	*I. virginica*
121	6.9	3.2	5.7	2.3	*I. virginica*
122	5.6	2.8	4.9	2.0	*I. virginica*
123	7.7	2.8	6.7	2.0	*I. virginica*

```
from sklearn.datasets import load_iris
iris=load_iris()
print(type(iris))
print(iris.data)
print(iris.data.shape)
print(iris.feature_names)
print(iris.target_names)
print(type(iris.data))
print(type(iris.target))
```

Each row is an Observation

(also know as: sample , example ,instance , record)

Each Column is a feature

(also Know as: predictor , attribute , independent variable)

```
from sklearn.datasets import load_iris
iris=load_iris()
iris.keys()
iris['data']
iris['feature_names']
iris['data'].shape
print(iris["target"])
print(iris["target_names"])
```

Predicting the Iris Flower

1.Import the class

```
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
```

2.Instantiate an Estimator

```
knn=KNeighborsClassifier(n_neighbors=1)
```

3.Model Training

Learning is the Relationship Between Data and Traget

```
iris=load_iris()
knn.fit(iris.data,iris.target)
```

Predict The Response for New Observation

The new observation are called "out-of-sample" Data

```
knn.predict([[3,5,4,2],])
print(iris.target_names[_])
```

```
iris_new=[[3,5,4,2],[5,4,3,2]]
knn.predict(iris_new)
print(iris.target_names[_])
```

Regression

The data will be split into a trainining and test set. Once we have the test data, we can find a **best fit line** and **make predictions**.

Load Data
Split the data into training/testing sets
Split the targets into training/testing sets
Create linear regression object
Train the model using the training sets

```
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
import pandas as pd
```

```
df =
pd.read_csv(r"C:\Users\Shubh_Ram\Desktop\Workshop_
AI_CODE\data\Housing _Data.csv")

df
```

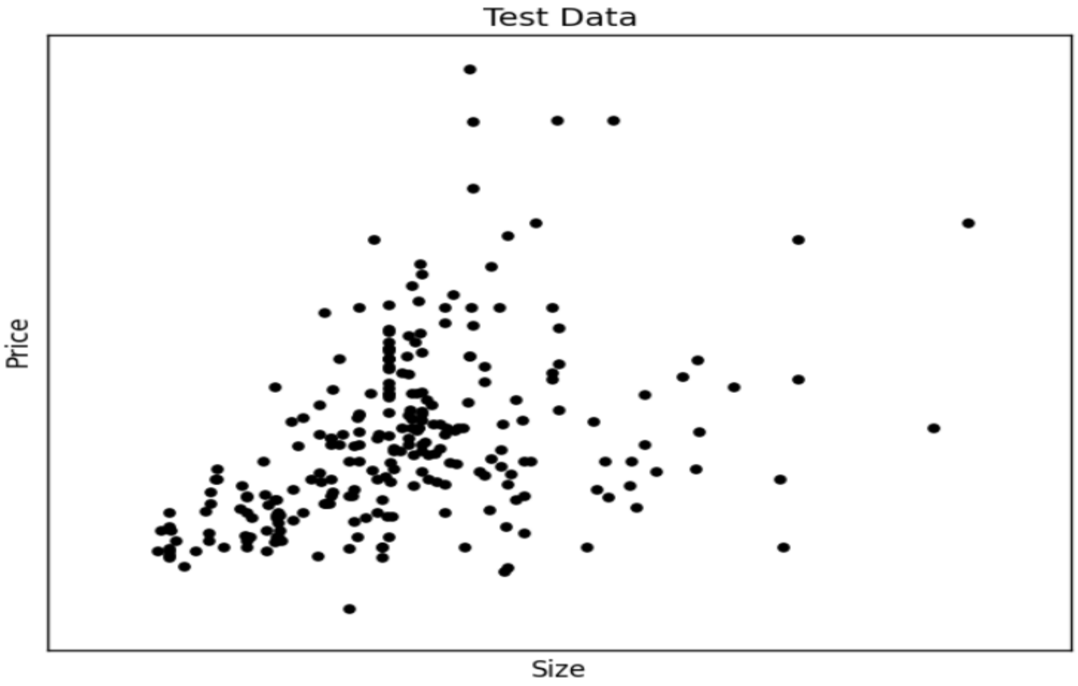
```
Y = df['price']
X = df['lotsize']
```

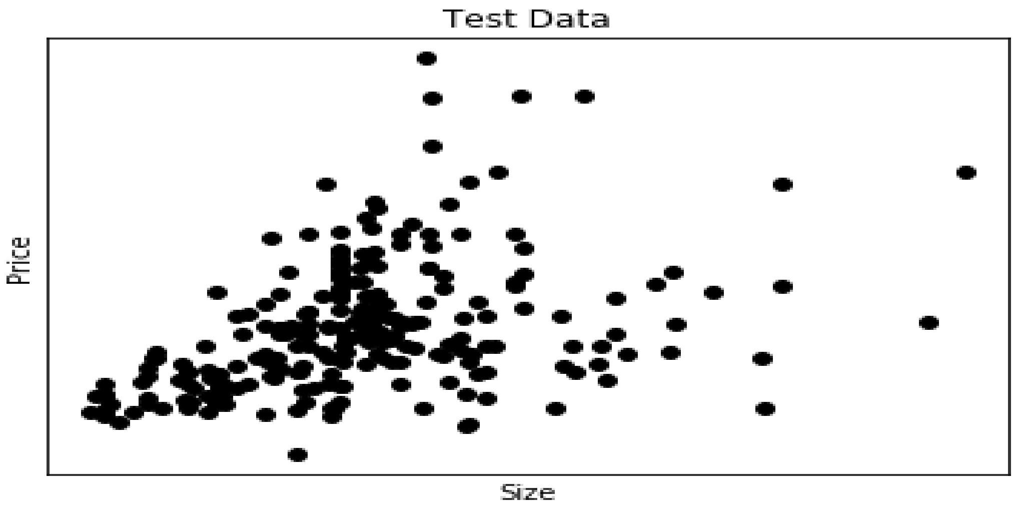
```
X=X.reshape(len(X),1)
Y=Y.reshape(len(Y),1)
```

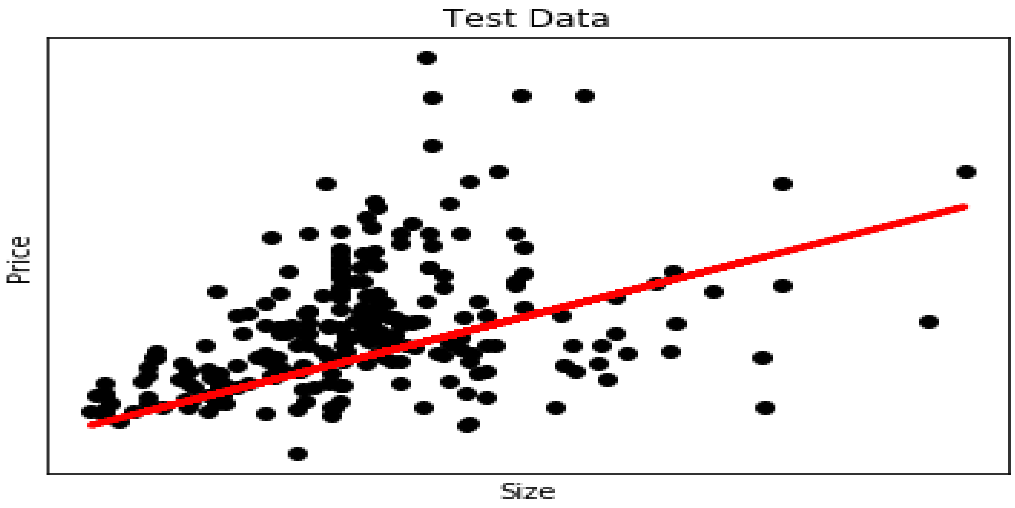
```
# Split the data into training/testing sets
 X_train = X[:-250]
X_test = X[-250:]
```

```
# Split the targets into training/testing sets
Y_train = Y[:-250]
Y_test = Y[-250:]
```

```
plt.scatter(X_test, Y_test,  color='black')
plt.title('Test Data')
plt.xlabel('Size')
plt.ylabel('Price')
plt.show()
```




Test Data
Price
Size


Test Data
Price
Size

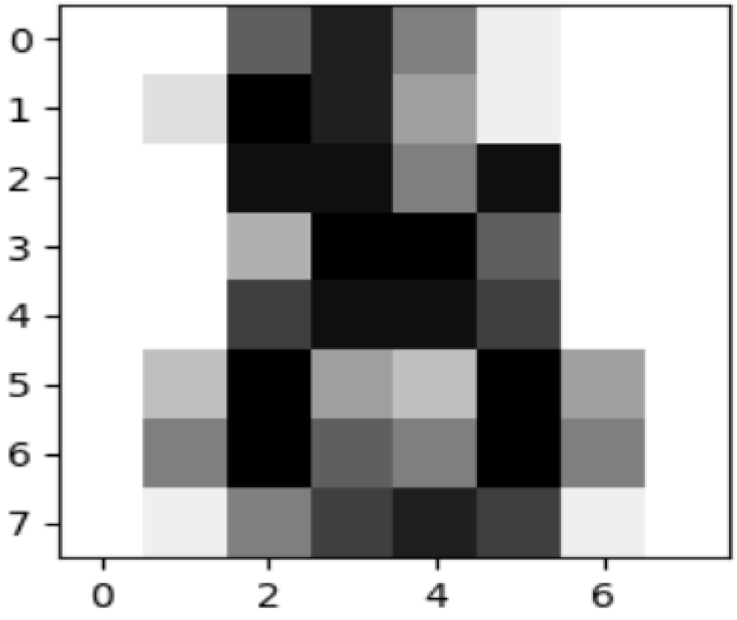
```
regr = linear_model.LinearRegression()
regr.fit(X_train, Y_train)
plt.plot(X_test, regr.predict(X_test),
color='red',linewidth=3)
plt.show()
```

```
print( str((regr.predict(5000))) )
```

```
# Create linear regression object
regr = linear_model.LinearRegression()
# Train the model using the training sets
regr.fit(X_train, Y_train)
# Plot outputs
plt.scatter(X_test, Y_test,  color='black')
plt.title('Test Data')
plt.xlabel('Size')
plt.ylabel('Price')
plt.xticks(())
plt.yticks(())
plt.plot(X_test, regr.predict(X_test), color='red',linewidth=3)
```

The Digit Dataset

This dataset is made up of 1797 8x8 images. Each image, like the one shown below, is of a hand-written digit. In order to utilize an 8x8 figure like this, we'd have to first transform it into a feature vector with length 64.



```
from sklearn import datasets
digits=datasets.load_digits()
x=digits.data
y=digits.target
print(x)
print(y)
```

Neural network models (supervised

Class MLPClassifier implements a multi-layer perceptron (MLP) algorithm that trains using Backpropagation.

MLP trains on two arrays: array X of size (n_samples, n_features), which holds the training samples represented as floating point feature vectors; and array y of size (n_samples,), which holds the target values (class labels) for the training samples

MLPClassifier

1.Hidden Layer

2.Batch size

3.Slover

fit(X, y)

After fitting (training), the model can predict labels for new samples

The Boston Housing Dataset

The dataset contains a total of **506** cases.

13 attributes or parameters

The goal of this exercise is to predict the housing prices in boston region using the features given.

```
from sklearn.datasets import load_boston
import pandas as pd
bostan=load_boston()
print(bostan.keys())
print(bostan.data.shape)
print(bostan.feature_names)
print(bostan.DESCR)
bos=pd.DataFrame(bostan.data)
print(bos.head())
bos.columns=bostan.feature_names
print(bos.head)
print(bostan.target[:5])
bos['PRICE']=bostan.target
print(bos.head)
```