Python Libraries for Machine Learning

NumPy – Fundamental package for **scientific computing** with Python.

Scipy – Built on Numpy and used for **mathematics,** science**,** and engineering.

Matplotlib – 2D plotting library that can be used to generate plots, histograms, power spectra, bar charts, scatter plots, etc.

Pandas – Provide high-performance, easy-to-use **data structures** and **data analysis** tool.

Scikit Learn – **Machine learning** tool in python for data analysis and **data mining**, built on Numpy, Scipy and Matplotlib.

Python Basics for Data Analysis

- Python is a simple, powerful and efficient interpreted language.
- Python Installation – Anaconda is open source data science platform which comes with preinstalled libraries.
- Together with Numpy, Scipy, Matplotlib, Pandas and Scikitlearn, it provides a very nice environment for data science work.
- Python is open source and cross platform.
- Python support modules and packages.

Python Syntax

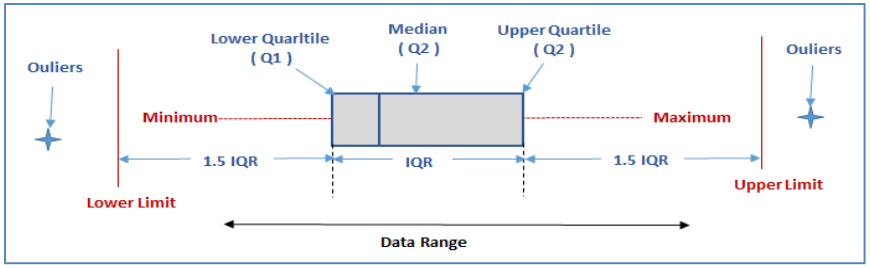
Syntax

- Comments are indicated with "#"
- No semicolon at the end of lines.
- Two statements on the same line are separated with a semicolon ";“.
- Grouping is obtained through indentation.
- Assignment uses the equal sign "=“.
- Identifiers are case sensitive, first character should be a letter or ‘_’.
- Assignments create references not values.

Exploratory Data Analysis

Exploratory Data Analysis using Pandas, Numpy, Scipy, Matplotlib

- Data Profiling to summarize the input data to asses the data quality and accordingly correct, discard and handle your data differently.
  - Data cleaning and missing values handling
  - Outlier Handling
  - Data Transformation



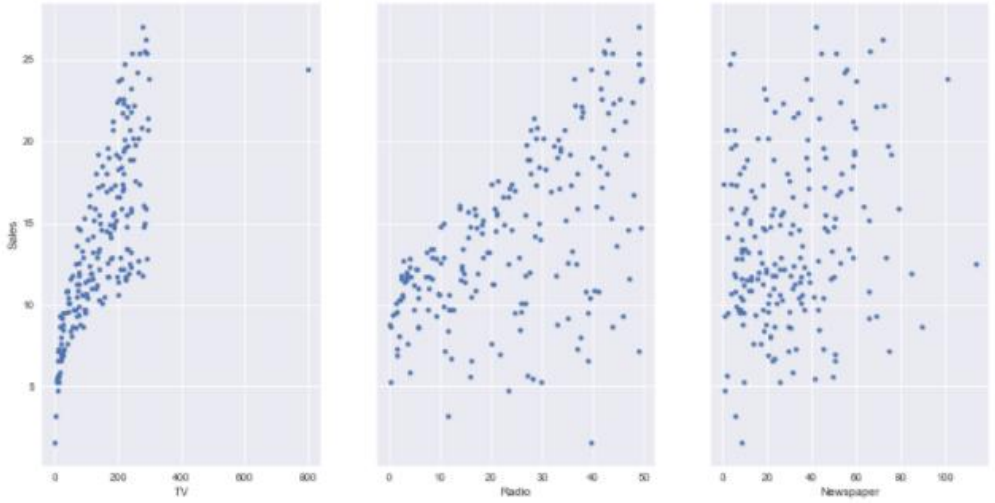
- Explore and compare different variables to search systematic patterns in data.

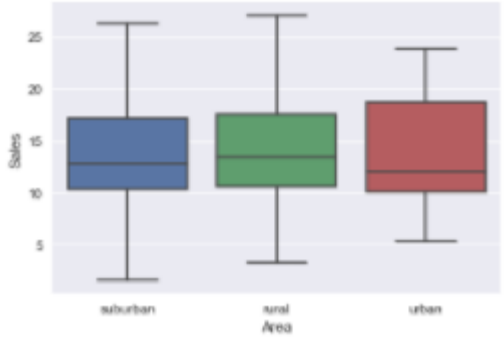
Exploratory Data Analysis

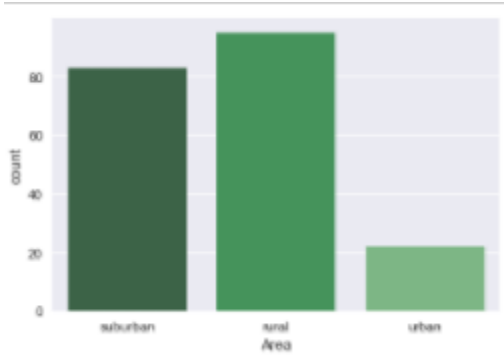
Exploratory Data Analysis using Pandas, Numpy, Scipy, Matplotlib

• Exploratory Visualization : Matplotlib is a core package for visualization, Seaborn is extension of matplotlib to make graphic look nicer.

Scatter Plot, Box Plot, Bar Chart, Histogram.







Scikit Learn Machine Learning Library

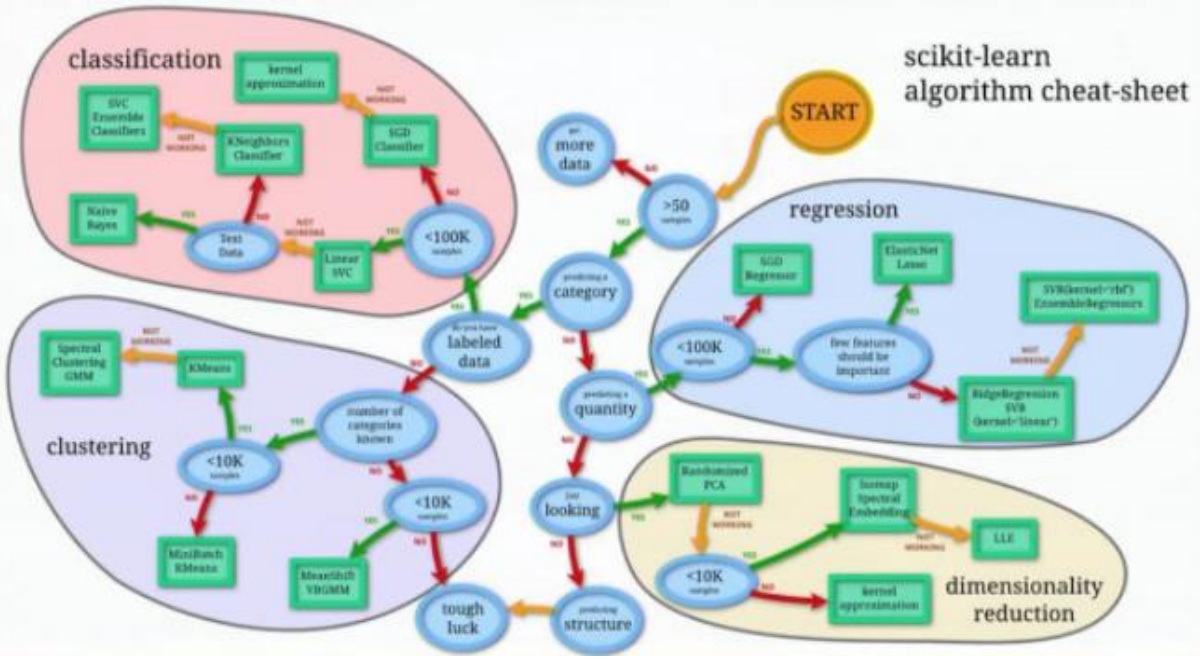
ScikitLearn : Python Machine Learning library, default installed with Anaconda.

Benefits:

- Consistent Interface to Machine Learning Models.
- Has lot many tuning parameters with good default values.
- Rich set of functionality available.

Drawbacks:

- Less emphasis on Model interpretability.



Scikit Learn Machine Learning Library

Iris Flowers Classification: Create a **classification model** to predict the iris flower types based on sepal and petal lengths/widths for a new flower.

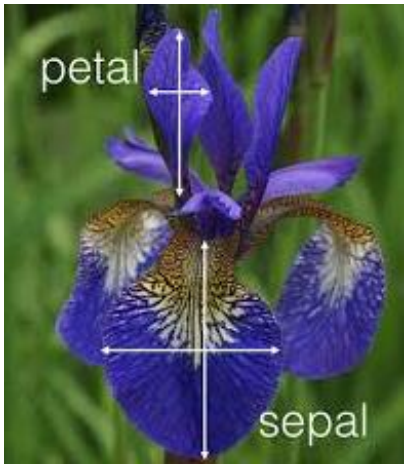
The Iris data set consists of the measurements of four attributes of 150 iris flowers from three types of irises.
The typical task for the Iris data set is to classify the type of iris based on the measurements

The data dimensions are as follows:

- sepal length in cm;
- sepal width in cm;
- petal length in cm;
- petal width in cm;

Class:

- Iris Setosa
- Iris Versicolour
- Iris Virginica



	sepal_length	sepal_width	petal_length	petal_width	response
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Scikit Learn Machine Learning Library

Scikit 6 steps process to build the predictive model (Consistent Interface)

#1: Import the model class that is gong to be used.

```
from sklearn.neighbors import KNeighborsClassifier
```

#2: Create the instance of the Model Estimator (instantiate the model).

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

#3: Split the whole data into training and testing data.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.35, random_state=2)
```

#4: Train the model based on training data.

```
knn_fit = knn.fit(X_train,y_train)
```

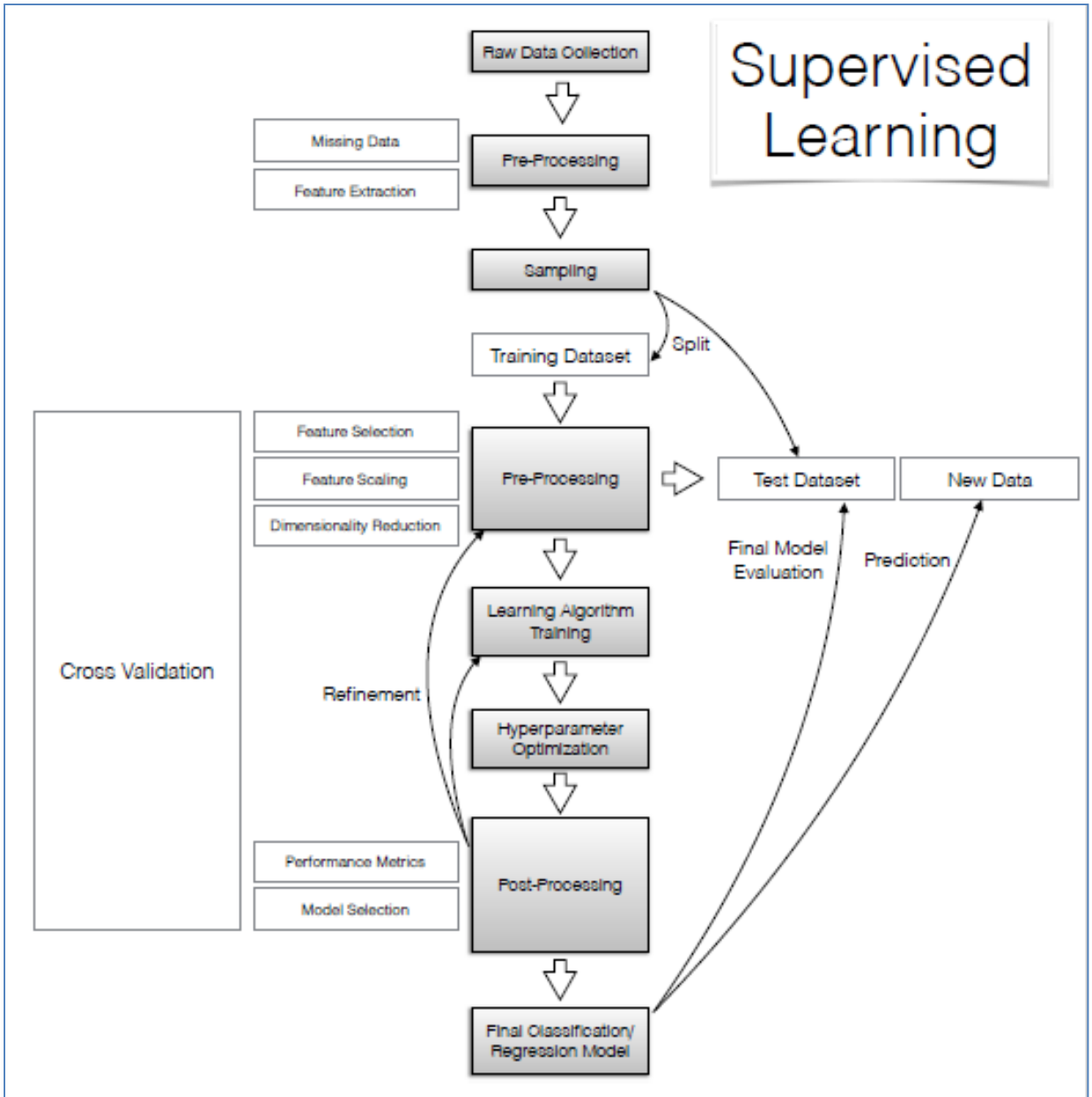
#5: Make the predictions on the test data.

```
y_pred = knn.predict(X_test)
```

#6: Compare the predicted response(y_pred) vs actual response(y_test)

```
from sklearn import metrics
print(metrics.accuracy_score(y_test, y_pred))
```

Supervised Learning Model Workflow



Scikit Learn Machine Learning Library

Model Evaluation Process: Estimate the Model predictive performance on out of sample data.

- Maximizing the **Training Accuracy** will lead to complex model that over fit the model.
- Split the dataset into two parts **Training** and **Test** datasets so model can be trained on training data and tested on Test data.
- **Testing Accuracy** is better estimates than training accuracy for out of sample predictive performance.
- But **train-test** technique provides **high variance**.
- **K-fold Cross validation** can over come of high variance of train-test strategy.
- K-fold cross validation can be used the **tuning parameters** and choosing between the models.

K-fold Cross validation Steps:

(i) Split the complete datasets into K equal partitions.
(ii) Use One partition for testing set and union of remaining other partitions for training set.
(iii) Calculate the Testing Accuracy
(iv) Repeat the step Two and Three K times using different partitions.
(v) Calculate the average accuracy from all the partitions that would be used for out of sample accuracy.