CHAPTER – 3
THEORETICAL BACKGROUND

3.1 INTRODUCTION

Data analytics is a process of inspecting, cleaning, transforming and modeling data with the goal of discovering useful information with signifying conclusion and supporting decisions. It is extended with multiple facets and approaches, which encompasses diverse techniques for respective domains. Big Data is a collection of massive data with efficient data storing technique, which focus the process of analytics on the emerging data distributed across various fields. It consists of unstructured data in a form of words, images and videos on web. The unstructured data is basically impossible to analyze using traditional database techniques. Big Data usually include datasets with size beyond the ability of common software tools for analyzing, managing and processing of data. Such huge data is parallely distributed on clusters of commodity hardware. Generally, data are parallely distributed and processed for analytics using a framework called MapReduce [62]. This framework designed by Google is a parallel programming model for processing and generating massive data sets. The special features such as load balancing and fault tolerance make the model an attractive application for processing large scale data analytics [63]. It performs various functions like text processing, graph processing, web processing etc. MapReduce is an open source data processing model to handle Google file system. Hadoop extensively uses MapReduce model to process large scale data sets generated from Yahoo!, Facebook etc. This chapter explains in detail about various machine learning algorithms that facilities automation systems. It also states the importance of various frameworks that process the massive dataset.

3.2 MAP REDUCE MODEL

MapReduce is a data flow paradigm for data centric applications. It is a simple explicit data flow programming model and preferential over the traditional high level data base approaches [64]. MapReduce paradigm parallelizes huge data sets using clusters or grids. A MapReduce program comprises of two functions, a *Map*( ) Procedure and a *Reduce*( ) Procedure. These functions are used for processing large scale datasets on computer clusters.

The functions Map( ) and Reduce( ) are given below where K1 and K2 are two key pairs and V1 and V2 are value pairs.

$$\text{Map}(K_1, V_1) \rightarrow \text{list}(K_2, V_2) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad 3.1$$

$$\text{Reduce } (K_2, \text{list } (V_2)) \rightarrow \text{ list } (V_2) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 3.2$$

A *Map*( ) procedure performs filtering and sorting by managing the "infrastructure". The *Reduce*() procedure summarizes the operations designing the "framework". Therefore, the system is referred as "*infrastructure framework*". This system runs several tasks in parallel on distributed servers by managing data transfer across various parts of the system. Figure 3.1 illustrates various phases of processing data [64].

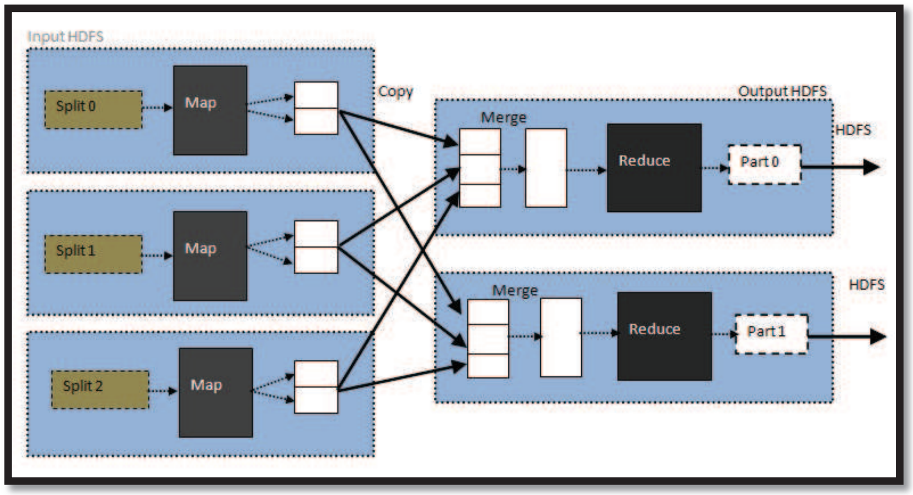


Figure – 3.1: Stages of MapReduce Programming Model [64]

MapReduce divides the input into fixed partitions called *input splits*, which runs in the user-defined map function on each record. Time required to process the map function on splits depends on number of maps used on a cluster. It takes minimum amount of time for higher number of map count with increased performance. Similarly, the performance decreases with lesser map count. The splits that occur minute are parallely processed for improving better load balancing. A swift system processes more number of splits in one course of action.

Machines existing in the cluster environment are prone to failure. If some process execution fails, then the job running on a map task is transferred to other map task to maintain load balancing in which number of splits are increased for fine grain. In other cases, if splits are too small it increases the overhead problem consequently by increasing the disk reads/writes, which results in low performance. This increases the overhead by escalating total execution time.

3.3 SPARK FRAMEWORK

Companies and technical sectors extensively use Hadoop for analytics based on the requirements featured from respective datasets. MapReduce is an efficient parallel programming tool enabling scalable, flexible, fault tolerance and cost effective processing. The main concern on these systems is the amount of computational time. Speed is a key component for large datasets, which are efficient for query processing and computation. The limitations faced by MapReduce programming model, which are high disk rate, low throughput and diminishing performance of a cluster are addressed by Spark [65].

Spark framework was introduced by Apache Foundations for parallel distributed Systems. It is an independent technology and is not part of Hadoop ecosystem. Spark has its own cluster management architecture for better storage [66]. The additional feature of Spark is it has powerful processing unit for fast computations.

Advanced methods like interactive query analysis and streaming are extended features of Spark framework over the MapReduce model. The significant feature of Spark is its in-memory cluster computing for enhancing the processing speed of an application. Spark covers wide range of workloads such as:

- Batch application
- Iterative algorithms
- Interactive queries
- Streaming

3.3.1 FEATURES OF SPARK FRAMEWORK

The following are the features of SPARK framework:

- ***Speed:*** It runs an application on Hadoop Cluster with 100 times faster in memory and 10 times faster on disks.
- ***Multiple languages:*** It supports various APIs written in Java, Scala, R and Python and maintains 80 high level operators for processing interactive queries.
- ***Advanced Analytics:*** Graph analytics, Streaming, Machine Learning and query analysis are advanced techniques in Spark Framework.

The following are the three ways of illustrating Spark deployment:

- ***Spark as Standalone*:** Spark is implemented on top of HDFS model for addressing all Spark jobs on a cluster.
- ***Hadoop Yarn Deployment*:** Spark runs on Yarn without pre installation and integrates with the ecosystem.
- ***Spark in MapReduce:***Launch of Spark job starts without additional administrative access.

3.3.2 CORE COMPONENTS OF APACHE SPARK

The following are the core components of Apache Spark:

- Spark is broad execution engine for in-memory computation and referencing datasets in exterior storage system [65].
- Spark SQL is a component that introduces data abstraction called Schema RDD for structured and semi-structure data.
- Spark Streaming influences fast scheduling capabilities by consuming parallel analyzed data after transformation.
- MLlib is a distributed Machine Learning library for alternative least squares and disk based versions of Mahout.
- GraphX is a built-in API for processing graph by using Pregel abstraction. This provides optimized runtime for abstraction.

3.3.3 RESILIENT DISTRIBUTED DATASETS

Resilient Distributed Datasets (RDDS) are immutable data structure collection of objects. The datasets are partitioned into logical blocks computed on different nodes in a cluster. This is implemented in APIs Java, Scala, R and Python [66]. The following are the features of RDD data structures:

- Read-only record structure
- Partitioned collection of records
- Deterministic operation
- Stable storage
- Fault tolerance

- Parallel processing
- Fast processing operations
- Immutable data structure
- Logical distribution

The two techniques involved in creation of RDDS are:

i. ***Distributing the collection:*** Parallelizing existing collection into corresponding driver program.
ii. ***Referencing a Dataset:*** It is an external storage system for sharing files in Hadoop with data sourcing input format.

3.3.4 DATA SHARING USING SPARK

Data sharing is a time consuming task for MapReduce due to replication, serialization and disk I/O. Ninety percent of Hadoop applications spend the time in read and write operation, which affects the performance [67]. This limitation is addressed by applying RDDS, a specialized data structure. RDD supports memory processing representing the state of memory across jobs in sharable network.

- ***Iterative Operation on Spark RDD:***The intermediate results are temporarily preserved in stable storage instead of moving to distributed memory.
- ***Interactive Operations on Spark RDD:***The different queries analyzed on same datasets are preserved on stable storage system for better execution. Each transformed RDD is recomputed during implementation. Spark maintains cluster documents, which are quickly accessible and persist even at a point of failure. It supports organizing more replicas across multiple nodes.

3.3.5 SPARK EXECUTION MODEL

Spark application execution involves runtime perceptions such as driver, executor, task, job and stage. Interpreting each component is significant in drafting resource efficient Spark code. Spark application maps to a single driver and to a combination of executors for processing distributed hosts on a cluster.

The driver manages the job and schedules the task for implementing the application at runtime. The driver initiates the jobs in the distributed system otherwise the shell takes up the role of driver process. The executors are responsible for executing the task in an organized form and in storing data in cache memory. The executor lifetime varies according to the dynamic allocation of data. This executor task maintains few additional slots for concurrent execution.

Invoking an action inside a Spark application triggers the launch of a job. It examines the dataset to set action and then formulates an execution plan. The execution plan assembles the dataset transformations into stages. A stage is a collection of tasks that run the same code each on a different subset of the data [67].

3.3.6 USING SPARK STREAMING

Spark Streaming is an extension of core Spark that enables scalable, high-throughput, fault-tolerant processing of data streams. Spark Streaming receives input data streams and divides the data into batches called DStreams. It is created either from sources such as Kafka, Flume and Kinesis or by applying operations on other DStreams. Every input DStream is associated with a receiver that receives the data from a source and stores it in executor memory.

3.3.7 USING SPARK SQL

Spark SQL is a processing tool for query structured data. It is used inside Spark programs by means of either SQL or through DataFrame API. The entry point for all Spark SQL functionalities is the SQLContext class or one of its descendants. A DataFrame is created from a RDD, Hive table or a data source using SQLContext. In Spark applications the stored data in Hive are processed by constructing a HiveContext, which inherits from SQLContext. With a HiveContext, tables are accessed in the Hive Metastore and queries are written using HiveQL.

3.4 HADOOP DISTRIBUTED FILE SYSTEM

The modern technological world faces a tremendous increase in data, which are complex to handle on a single machine. Therefore, it is essential to distribute the partitioned data blocks on multiple machines [68]. A file system distributed across network of machines is referred as distributed file system.

Data are scattered over various locations on network. It is typical to manage and control the flow of data with wide number of complications in network programming. For an instance, the biggest challenge is node failure in a distributed environment. The file system used in

Hadoop ecosystem is HDFS (Hadoop Distributed File System) [68]. HDFS key components are Name Node and Data Nodes. Hadoop Distributed File System uses master slave architecture. Data is distributed on racks in Hadoop Cluster. Job-tracker and task-tracker are two housekeeping components, which are scattered on the cluster [69]. Figure 3.2 describes the internal architectural view of Hadoop Distributed File System.

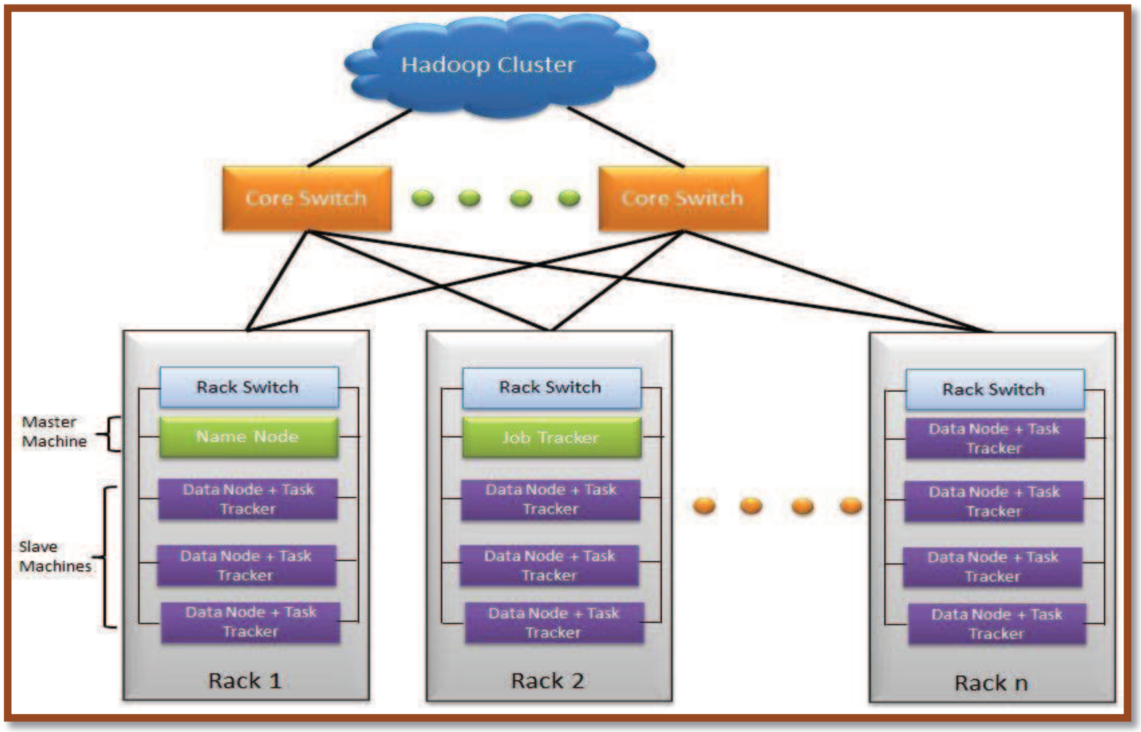


Figure – 3.2: Hadoop Distributed File System Architecture [69]

3.5 MACHINE LEARNING TECHNIQUES

Machine Learning is an application of science that creates logic from data by transforming them to knowledge [70]. Machine Learning consists of many powerful algorithms to learn patterns, acquire insight and prediction. Artificial Intelligence (AI) evolved as a subfield for development of self-learning algorithms for an insight. In the present age of rapid development, the availability of immersed data starves for knowledge.

Generally, data are represented in two forms (i) structured and (ii) unstructured. Structured data is the data arranged in tables and unstructured is the data with irregular form such as images, documents, text, audio and video. Machine Learning provides efficient analysis models to capture knowledge by improving prediction and data driven decisions [71]. It plays an eminent role in field of computer science especially for analyzing streaming data such as robust emails, spam filters, convenient text, voice recognition, web search engines, game developments and self – driving cars. Learning is an activity through which a model is tuned for solving various problems based on the parallel distribution of data.

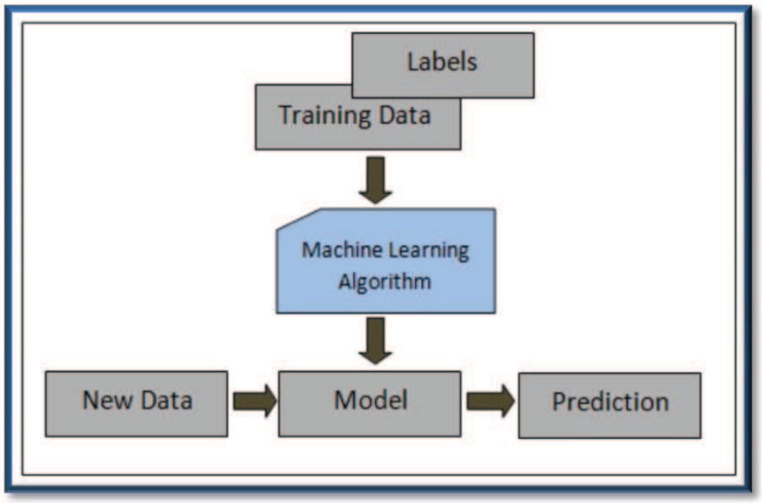


Figure – 3.3: Machine Learning Procedure for Processing Datasets [71]

Learning process involves following procedure for processing massive data sets. The steps involved in processing the datasets are demonstrated in figure 3.3.

Step1: Collection of data.

Step 2: Partition the data as trained datasets and test by identifying labels.

Step 3: Deploy the Machine Learning algorithms on models.

Step 4: Return output as per the requirement specified.

3.5.1 TYPES OF MACHINE LEARNING ALGORITHMS

Machine Learning methods are classified into three techniques. Each learning methods has its significance and dimensionality.

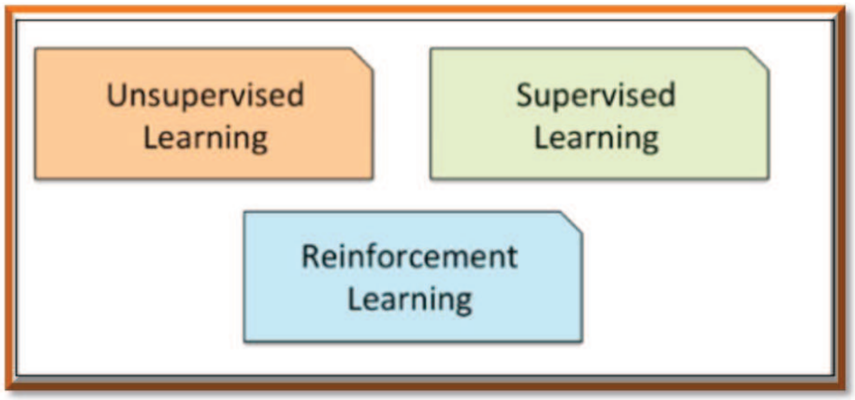


Figure – 3.4: Types of Machine Learning Methods [71]

The following are the prime categories of machine learning methods (Refer figure 3.4)

- Supervised Learning Method
- Unsupervised Learning Method
- Reinforcement Learning Method

3.5.1.1 *Supervised Learning Method*

In supervised learning method, a model from labelled training data is interpreted to make prediction for test data [72]. The term supervised learning method refers to labels which is known as a set of samples to attain desired outcome. Supervised learning are categorized into two tasks:

- Classification
- Regression

Consider an example of e-mail spam filtering, a model using an e-mail corpus is trained with supervised learning methods to identify the spam. This model is tested to detect the occurrence of spam.

- ***Classification:*** A learning task with discrete class labels that uses binary classification to predict a new instance from past observations is known as classification.
- ***Regression:*** An expected task with continuous unordered class labels to forecast on new instance and to find relationship between those variables is known as Regression. It is an associated method for prediction analysis. This research study mainly focuses on four prediction models as discussed in the following section.

3.5.1.1.1 *Linear Regression*

Linear Regression is one of the predictive modeling techniques. According to this technique, dependent variable is continuous and independent variable may or may not be continuous. To fit a straight line a relationship is established between independent and dependent variable. The straight line equation is given as $y=a+bx+e$ where, a is an intercept, b is a slope and e is an error term.

- The method of least squares is extended with multiple independent variables is represented in the following form:

$$y = c + b_1x_1 + b_2x_2 + \cdots + b_kx_k \text{ ....} \quad 3.3\ [73]$$

- The least square represents an equation for the line as shown below:

$$b_0 = b_1(x_1 - \overline{x_1}) + b_2(x_2 - \overline{x_2}\,) + \ldots + b_k(x_k - \overline{x_k}) \text{ ....} \quad 3.4\ [73]$$

- Thus equation is transformed into

$$y - b_0 = \sum_{j=1}^{k} b_j(x_j - \overline{x_j}) \text{....................................} \quad 3.5[73]$$

- Computing the values of b_j coefficients for which the sum of squares is expressed as minimum is represented as follows:

$$\sum_{i=1}^{n} (\hat{y}_i - y_i)^2 \quad \text{..} \quad 3.6[73]$$

- The coefficients b_j are computed as follows.

$$cov(y, x_j) = \sum_{m=1}^{k} b_m * cov(x_m, x_j) \text{ ....} \quad 3.7[73]$$

- Thus the population and sample covariance are calculated for variations between the variables by computing R square.

A straight line is assumed between the input variables (x) and the output variables (y) showing the relationship between the values. Statistics on the training data is required to estimate the coefficients. These estimates are used in model for prediction and data processing. The line of simple linear regression model is y= a1+a2*x where *a1* and *a2* are the coefficients of the linear equation. Generalized linear models follow Gaussian distribution for predictions which are drafted in canonical form of exponential distributions.

Estimating the coefficients is given as follows:

$$a1 = \frac{sum\left((x(i) - mean\ (x)) * (y(i) - mean\ (y))\right)}{sum\left((x(i) - mean\ (x))^2\right)} \quad \text{..................} \ 3.8$$

$$a0 = mean(y) - a1 * mean(x) \qquad \text{.........................} \ 3.9$$

3.5.1.1.2 *Decision Tree*

Decision tree classifiers are significant techniques for prediction by considering a special feature called interpretability. This model breaks data for appropriate decision making process and to answer series of queries to interpret the features. The decision tree model learns from features of training set to infer the class labels of the samples [74].

The algorithm starts from root node and split the set, which results in largest information gain. It is an iterative process by partitioning each child node until the leaves are pure. Samples at each node that belong to the same class at times results in overfitting.

Decision Tree is a classifier which concentrates on interpretability. It suggests a break down to make decisions by posing queries. A tree is pruned by fixing the limit for the maximal depth of the tree.

$$I_E = 1 - max\{p(i|t)\} \text{ .... } 3.\ 10\ [74]$$

Where I_G is Gini Index I_H is entropy and I_E is the classification error defined for all non-empty classes.

$$I_H(t) = -\sum_{i=1}^{c} p(i|t) \log_2 p(i|t) \text{ .... } 3.11\ [74]$$

$$I_G(D_P, f) = I(D_P) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j) \text{ .... } 3.12\ [74]$$

The decision tree is the process that begins at the root and followed by splitting the data using information gain. It is an iterative process that reaches the maximum depth of the tree using pruning. The procedure is illustrated as follows:

- An objective function is defined by splitting the nodes, which produce useful features via tree learning algorithm to optimize the requirement.
- The objective function maximizes the information gain at each split that is defined in the equation 3.10 and 3.11.
- Here *f* is the feature to perform the split and $D_p$and D_j are the dataset of the parent p^th node and j^th the child node.
- *I* is the impurity measure, whereas $N_p$and $N_j$are the number of samples in the parent and j^th the child node.
- Information gain is defined as difference between the impurity of the parent and summing up the impurities of the child nodes.
- Low ratio of impurity in child nodes results larger information gain. The parent node is split into two child nodes at each iteration, child left and child right.
- The measures to construct binary decision trees are Gini index (I_G), Entropy (I_H) and Classification error (I_E) for non-empty classes.

Gini index is a criterion to minimize the probability of misclassification. Entropy is to maximize the mutual information in the tree and error is the measure of impurity resulting in different pruning cut-offs [75].

The limitation of decision tree structure is overfitting. This issue is raised when algorithm continues to navigate in deeper level to reduce the training error but finally ends up in increased test set error. Ultimately this estimates poor prediction value. It generally happens when it builds numerous branches due to outliers and irregularities in dataset [76].

Two approaches which solve the issue of overfitting are *pre-pruning* and *post-pruning*:

- ***Pre-Pruning:***The tree structure is not further extended based on a threshold value. It terminates the tree construction little early as a perfect threshold value cannot be predicted [82]. The appropriate terminate point is difficult to choose for goodness measure, which is below a threshold value.
- ***Post-Pruning:***In post-pruning, a tree is built that navigates deeper for completing tree. Cross validation is used to check data that effect pruning. Test is conducted to check the expansion of a node for further improvement and best prediction. If the node marks the sign of improvement the tree is further expanded. If the accuracy is reduced, the tree is terminated and the node is converted into a leaf node [79].

3.5.1.1.3 *Random Forest*

A random forest is a combination of various decision tree classifiers that fits dataset as a meta-estimator for predictive accuracy and control over-fitting. This learning technique is applicable for classification and regression models [77]. This method operates on many decision trees as training set and it results a class of individual trees to form output. A random forest considers feature significance and these features are measured as the mean impurity. These impurities taken from decision trees are diminished [78]. The random forest algorithm is summarized in the following simple steps:

- Draw a random bootstrap sample of size n.
- Produce a decision tree from the sample extracted by bootstrap.
- At every node:
  - Randomly select d features without replacement.
  - Split the objective function capitalizing the information gain.

- Aggregate the predictions from each tree to assign the class label by majority vote.

In this model, training is given to extract individual decision tree instead of evaluating the features. These features are evaluated to determine best split at each node through random subset. Hyper parameter values are properly chosen and estimated using the bootstrapping.

Explicitly, pruning is not required as the system ensemble with required robust features eliminates noise [79]. Typically, the number of trees used, shows the better performance of the random forest at the expense of an increased computational cost. The '*n*' refers to the size of bootstrap sample '*d*' defines number of features for a random split.

Bias variance tradeoffs are controlled by large value of '*n*' and are decreased by the randomness which over fit. The degree of overfitting is reduced by choosing smaller values of n at the expense of the model performance. In this implementation, RandomForestClassifier method for a bootstrap sample is chosen to be equal. The original training set is obtained on executing d = $\sqrt{m}$ where m is the quantity of features in training set.

This is evaluated using n-estimators parameter and entropy criterion as an impurity measure to partition the nodes. Small random forests are grown from a very large training set with n-job parameters to parallelize the model training to process on multicore systems. Breiman [80] used statistical framework for recasting the Arcing – Adaptive Reweighting and Combining algorithm as classifiers and weighted inputs.

The pseudocode for random forest are partitioned into two stages, namely, *Random forest creation pseudocode by classification* and *Pseudocode to perform prediction from forest*.

i. ***Random Forest Algorithm for Basic Classification:*** The following are the steps:

   Step 1: Randomly select '*k*' features from total '*m*' features where k < m.

   Step 2: Among the '*k*' features compute the node '*d*' utilizing split point.

   Step 3: Split the nodes into child nodes using the best classification.

   Step 4: Repeat steps 1 to 3 till the single node is reached.

   Step 5: Creating '*n*' number of trees by repeating steps 1 to 4 'n' times.

ii. ***Random Forest Algorithm for Regression:*** The following are the steps:

   Step 1: Partition the data as trained dataset to run the algorithm.

   Step 2: Consider test features and apply the rules of randomly created decision trees.

Step 3: Each outcome of decision tree is predicted and stored as respective outcomes.

Step 4: Calculate votes for each predicted target trees.

Step 5: Consider the high voted target tree to be the best and unique prediction.

Step 6: This technique is referred to be majority voting.

3.5.1.1.4 *Gradient Boosting Tree*

Gradient boosting is a voracious algorithm that over fits training objects rapidly. It builds additive regression model with least squares method for each iteration function and is parameterized to present pseudo residuals. In the present step, the model evaluates the pseudo - residuals on trained data which are responsible for gradient loss function that needs to be minimized. For further improvement, the model incorporates randomization to prove the approximation precision and execution pace of gradient boosting [81].

The idea of gradient boosting tree is to transform a weak learner to a better learner by boosting. Michael Kearns articulated the goal as the "*Hypothesis Boosting Problem*" stating the goal from a practical standpoint as an efficient algorithm for converting relatively poor hypotheses into very good hypotheses. A weak hypothesis is described as performance improvement is shown by boosting instead choosing a random chance [82].

Specifically, this algorithm iterates at each draw on a subsample of the training data randomly (without replacement) from the full training data set. Gradient Boosting Tree approach increases robustness besides congestion of training sets. Further this algorithm is developed by adding prediction techniques by Friedman and referred as gradient tree boosting [82].

The statistical framework refers boosting as a numerical optimization problem that identify weak learners using gradient descent procedure to reduce the loss of the model. It presents two approaches (i) *stage-wise* and (ii) *step-wise* in which stage wise are additive model as it adds a new weak learner with the existing weak learners and step wise deals with formerly entered weak learners when new ones are added to the model.

The generalized technique uses arbitrary differential loss function that expands the method further as binary classification. With the necessary expansion the issues to support regression, multi-class classification is properly addressed by using Greedy Function Approximation.

Gradient Tree Boosting Algorithm involves three elements (i) loss function (ii) a weak learner and (iii) an additive model.

- ***Loss Function:***A loss function varies according to the dataset used to solve a particular problem. Standards are defined to control the loss function differentially. As an instance, mean square error is generated for regression models whereas logarithmic loss is applicable for classification models. An advantage of the gradient boosting framework need not derive a loss function for every new boosting algorithm. It is a generic framework used for differentiating loss functions.
- ***Weak Learners:***Decision trees are used in gradient boosting to interpret weak learners. While using regression trees the real values, which are produced as output from various splits is considered and added collectively. The output of the consequent models is added as an existing collection and the best prediction is measured through the rectification of the residual errors. Trees are constructed in a greedy manner by preferring the best split through computing the purity scores from Gini Index in order to minimize the loss. Basically AdaBoost method constructs very small decision trees for a single split and is referred as Decision stump. Generation of larger tree extends up to 4 – 8 levels in development. The basic intention is to refine the weak learners and it is the common method to constrain weak learners for better model. The specific way is to maximize number of layers, nodes, splits or leaf nodes of a learning environment. Therefore, the weak learners are trained till the tree is extended in a greedy way.
- ***Additive Model:***Trees are supplemented at a time and existing trees are not modified. Addition of tree helps gradient descent procedure to minimize the loss. Gradient descent is used to minimize a set of parameters such as the coefficients in a regression equation or weights in a neural network. Based on the error calculation, the weights are updated to minimize the error. The parameters are replaced with weak learner's sub-models or extra specific decision trees. Based on the appropriate loss calculation the gradient descent algorithm is applied to reduce the loss. Parameterized tree is to modify the parameters of the tree that frames right direction for reducing loss. This is referred to as *functional gradient descent*. One way to produce a weighted combination of classifier to optimize the cost and time using functional space is gradient descent. The output for the new tree is added to the output of the existing sequence of trees to improve the final model. A fixed

number of trees are collected once the loss reaches a threshold level where improvement cannot be extended.

Updating Basic Structure of a Gradient Boosting Tree: Gradient Boosting is a greedy algorithm that over fits a training dataset. Regularization method is applied to penalize the various parameters of the function to improve the performance and reduction of overfitting.

The requirements for model enhancements are given in the following structure:

- Tree Constraints
- Shrinkage
- Random Sampling
- Penalized Learning

Table – 3.1: Constraints in Constructing a Tree Model

Number of trees	**Adding more trees to the model are very slow to over fit. Therefore, add trees until no further improvement is observed.**
Tree depth	Deeper trees are more complex trees and shorter trees are preferred. Generally, better results are seen with 4-8 levels.
Number of nodes or number of leaves	This can constrain the size of the tree, but is not constrained to a symmetrical structure if other constraints are used.
Number of observations per split	This imposes a minimum constraint on the amount of training data at a training node before a split is considered.
Minimum improvement to loss	It is a constraint on the improvement of any split added to a tree.

Tree Constraints: In a skilled tree, weak learners remain weak due to some constraints in tree structure. A heuristic approach is that the tree creation is constrained to which more number of tree models are generated to complete the solution and this approach is vice-versa for building the tree models. Some of the constraints that are imposed on the construction for a model are listed below in table 3.1.

Weighted Updates: Prediction of each tree is added sequentially. The contribution of each tree to the sum is weighted to slow down by learning the algorithm. This weighting is called shrinkage or a learning rate. The effect makes the learning slowdown that results in construction of more trees. This takes longer amounts of training time that provides a configuration trade-off between number of trees and rate of learning.

$$Best\ Value = \frac{learning\ \ rate\ (v)}{number\ \ of\ trees\ (M)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots. \quad 3.13\ [83]$$

Learning Rate: is represented as V when decreased and number of trees is represented as M when increased for best value of the model. The value ranges from 0.1 to 0.3 and less than 0.1. Stochastic optimization works similar to learning rate where shrinkage reduces the influence of individual tree and leaves space for future trees to improve the model [83].

3.5.1.2 *Unsupervised Learning Method*

The Unsupervised learning method deals with unlabeled data or unknown structure. The data is explored for information extraction to analyze outcome variable [84]. For feature selection, different algorithms are used to reduce the dimensionality of a dataset. Another approach for dimensionality reduction is feature extraction.

Summarization and *Understanding* are the two techniques for N-Dimensional space in predefined structures [85]. This technique automatically returns amazing outcomes by implementing grouping and learning methods on data points. Learning is uncertain and unpredictable as they produce random results for different choice of features.

Clustering is an unsupervised learning technique to cluster similar features, a technique to group the similar objects on different metrics. The procedure for clustering is to collect information of different consent [86]. Each cluster is sorted by crafting the datasets that influence the model. Finally, these crafted clusters are campaigned for metric computations. It is an exploratory data analysis methodology, which organizes the data in meaningful and structural blocks devoid of any experience in group membership. Each block in a cluster analyze the groups and share degree of similarity between objects for classification [87]. This technique derives significant relationships among datasets. The following are the types of unsupervised learning:

- Clustering – K Means
    - Fuzzy Clustering
    - Hierarchical Tree based Clustering
- Principal Component Analysis
- Linear Discriminant Analysis.

3.5.1.3 *Reinforcement Learning Method*

A popular example of reinforcement learning is a Chess game on personal computers. Generally, the agent decides upon a series of moves depending on the state of the board (the environment) and the reward is defined as win or losesat the end of the game.

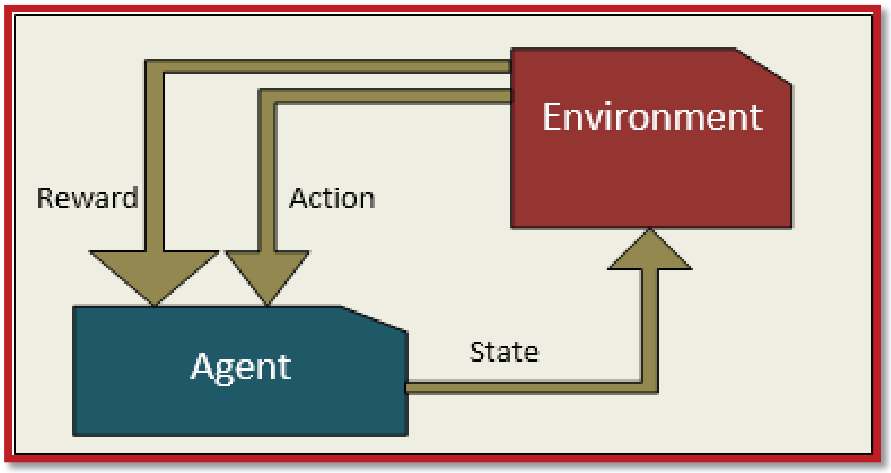


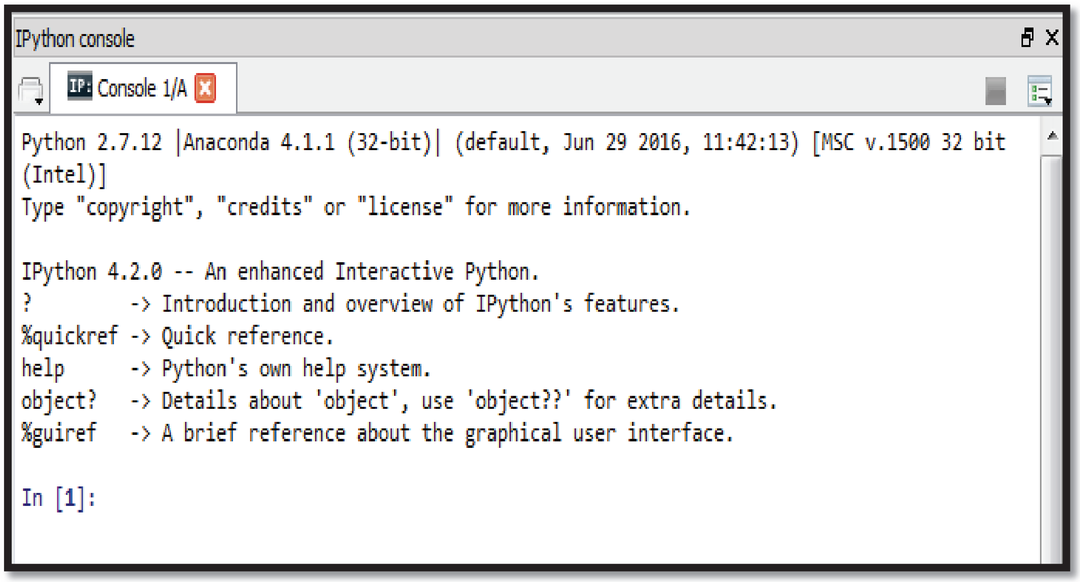
Figure – 3.5: Learning Process in Reinforcement Learning

In reinforcement learning, the goal is to develop a system (agent) that improves its performance based on interactions with the environment [88]. Since the information about the current state of the environment typically includes reward signal the reinforcement learning resembles similar to supervised learning as illustrated in Figure 3.5. The feedback is not the correct ground truth label or value but a measure of action calculated by a reward function. Based on the interaction with the environment an agent uses reinforcement learning to understand series of actions that maximizes the reward via an exploratory trial-and-error approach or deliberative planning.

The learning method allows a system to learn its behavior through the feedback received from the environment. It is an iterative process that updates every moment adapting to environment. The process is modeled with lot of attention so that it converges to the global optimum for maximize reward [89]. This automated learning scheme implies the need for a human expert for the application of domain features. The design for solutions demands less time. Experts with good Reinforcement learning knowledge is needed to address the issue. Reinforcement learning method estimates the outcome in each state. The predicted value is tuned over time by proliferating to the next reward. Among the optimum policies chose the action, this maximizes the value for the next state from previous state and action made to the system [89].

3.6 PYTHON

Python is an innovative and cognitive programming platform for the modern generation. It is efficient in maintaining high level data structures, classes and object oriented approaches. Python has an elegant IPython console for mathematical and statistical computations [90]. This platform uses interpreter for translation that supports language scripting and ideal for web applications like DJANGO. This is the best platform for mastering the Machine Learning algorithms [91] [92]. This language excels at string processing; the manipulation of strings lists a few languages with good string processing capabilities. Scikit Learn is a python Machine Learning library for implementing a robust analytical system. Figure 3.6 illustrates the screenshot of IPython console.



```
Python 2.7.12 |Anaconda 4.1.1 (32-bit)| (default, Jun 29 2016, 11:42:13) [MSC v.1500 32 bit
(Intel)]
Type "copyright", "credits" or "license" for more information.

IPython 4.2.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
%guiref   -> A brief reference about the graphical user interface.

In [1]:
```

Figure – 3.6:IPYTHON Console

The features of Scikit – Learn for implementing Machine Learning in Python are:

- Simple and efficient tools for data mining and data analysis.
- Accessible and reusable in various contexts.
- Built on NumPy, SciPy and matplotlib.
- Open Source and commercially usable – BSD License.
- IPython – enhanced interactive Python interpreter.
- Python Library – NumPy.
- SciPy – Signal and Image processing.
- Matplotlib – interactive 2D/3D plotting.

3.7 ORANGE

Orange is an open source machine learning with data visualization and interactive data analysis tool box. It is scripted in Python programming language. Orange has multiple components known as widgets [93]. Figure 3.7 illustrates the widgets incorporated in the Orange tool. The components of Orange tools are classified into [94]:

- ***Data:*** For fetching files from sources.
- ***Visualize:*** For generating graphical and chart representations.
- ***Classify:***For producing Tree based classification.
- ***Regression:*** For Prediction analysis using learning methods.
- ***Evaluate:*** For Testing and extracting computational metrics.
- ***Unsupervised:*** For Predefining Unsupervised learning methods.

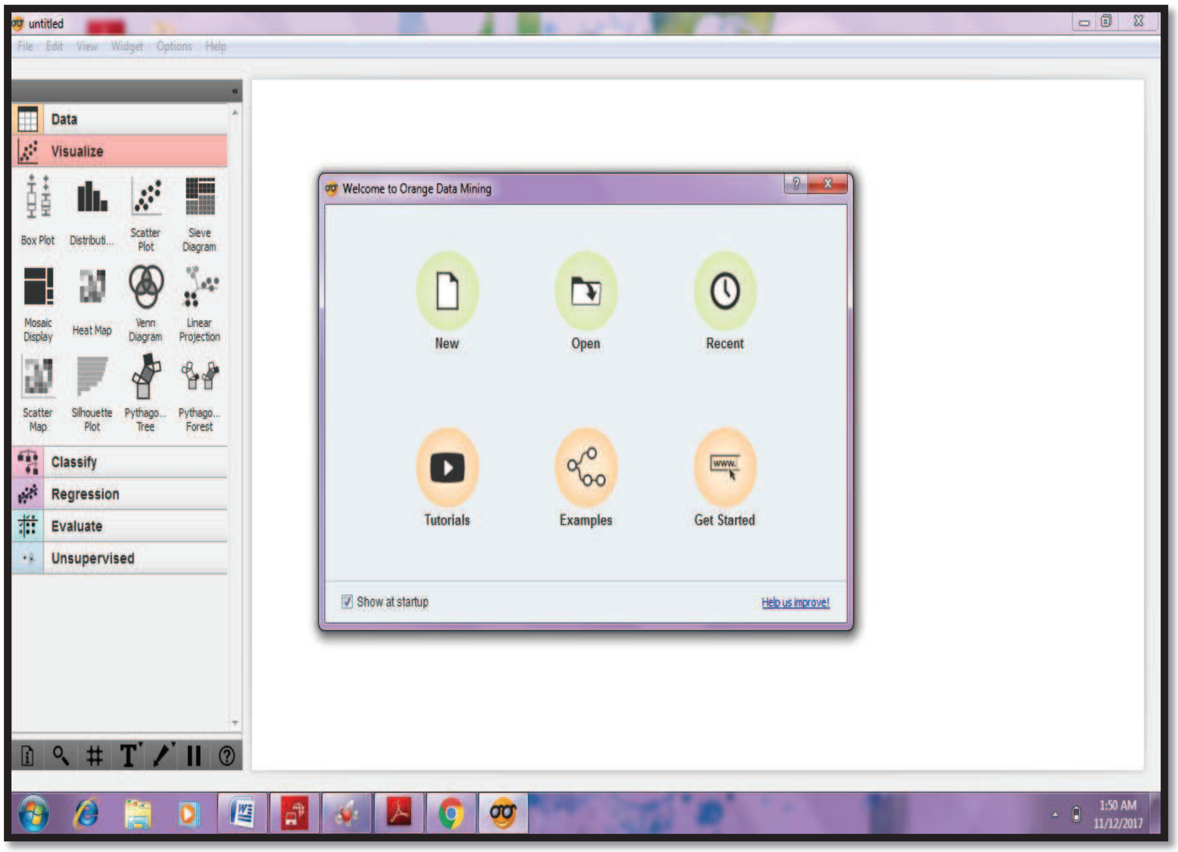


Figure – 3.7: Orange Canvas for Data Analysis

3.8 SUMMARIZATION OF PROCESSING TOOLS

Table 3.2 gives a detailed description on the processing tools used in this research work. The significance of the tools is narrated for easy understanding of the equipped tools.

Table – 3.2: Summarization of Processing Tools Utilized

Tools Utilized	Purpose
MapReduce	Frame work for data distribution
Spark	In Built distributed frame work
Machine Learning Techniques	Developing an analytical model
Supervised Learning Methods	Prediction
Python	Programming
Orange	Comparison

3.9 SUMMARY

An overview about the MapReduce and Spark with its related concepts is explained in this chapter. The importance of Machine Learning algorithms to handle the massive data is clearly illustrated in this chapter. An intricate knowledge about Python and Orange tools are given that supports the factors in the implementation phase. This chapter focused on the theoretical backgrounds of all conceptual ideas related to big data analytics.