

CHAPTER - 4

RESEARCH METHODOLOGY

4.1 INTRODUCTION

Data analysis is the process of analyzing the data to extract knowledge. The extracted information is useful for further analytics. Due to the rapid increase of data across various domains processing those using manual system is a challenging task. To overcome this issue automated systems are introduced to process real time data. This is achieved by implementing Machine Learning techniques. In this chapter the features of automation are enabled by using Spark an open source tool that support parallel distributed environment. The Machine learning algorithms are employed on Spark to generate prediction outcomes. The encapsulated resources that features the activities of proposed model is described in the following sections.

4.2 ENVIRONMENTAL SETUP

An environmental setup is established across cloud platforms by taking the instances from Amazon Web services on EC2. The virtual machine is equipped with the installation of Spark and Anaconda in which the Jupyter notebook are accessed by executing the Machine Learning algorithms. Orange is an open source tool that interprets the results of Machine Learning algorithms through cross validation. Table 4.1 describes the tools required for the implementation.

Table – 4.1: Tools Applicable for Implementation

Work Flow	Tools	Version
Job Distribution	Spark	Spark-2.1.0-bin-hadoop2.7
Prediction	Linear Regression Random Forest Decision Tree Gradient Boosting Tree	Anaconda3-4.3.1
Canvas	Orange	Orange 3.2

The work flow that potrays the establishment of virtual machine instances on cloud platforms are shown in Figure 4.1. Initial step consists of data set stored on the virtual cloud. During the preparation of cloud environment, *Putty* an open source key generator tool is used for exchanging the RSA keys on a cloud system. Based on the authetication approval an instance is made available on EC2. According to the need the required tools that benefits the implementation task are installed on the instance. Thus the setup is made available for implementing the learning algorithms on the virtual machine.

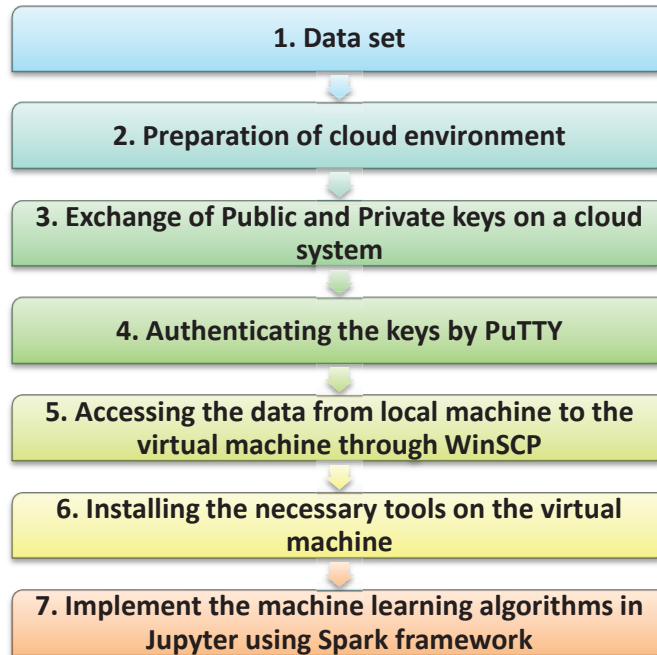


Figure – 4.1: Workflow of Environmental Setup on a Cloud Instance

WinSCP (Windows Secure Copy Protocol): WinSCP is an open source protocol based tool. The secured protocols are SFTP, SCP and FTP. These protocols are used to transfer files between clients. The main functionality is to provide a secured file transfer between local and remote computer (Refer Figure 4.2). File management and file synchronization are significant features of WinSCP.

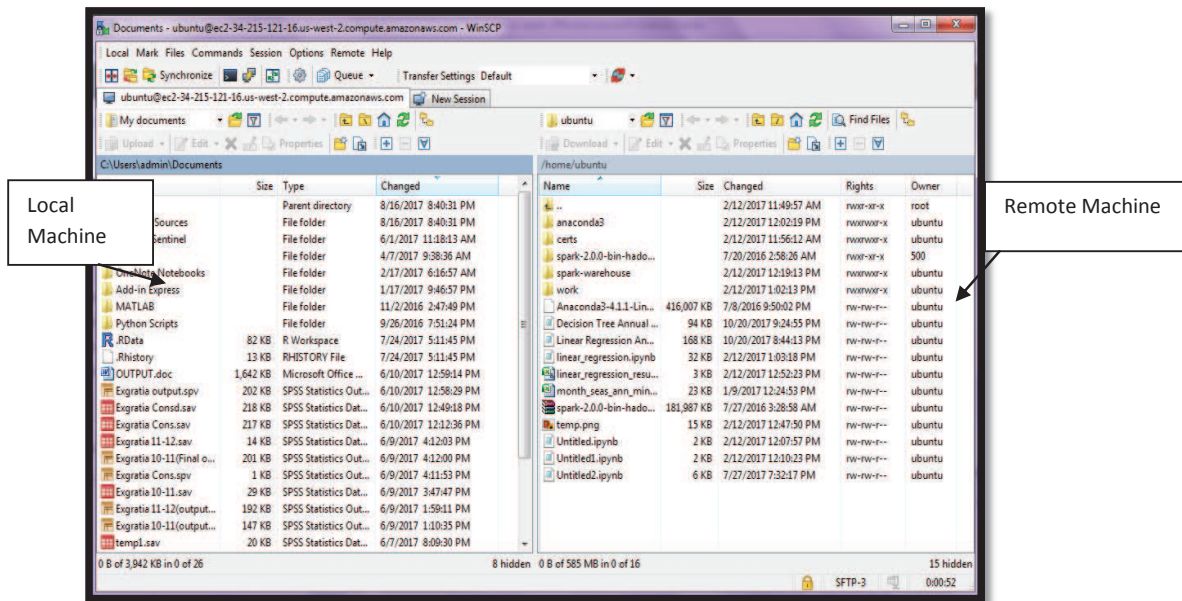


Figure – 4.2: WinSCP for File Transfer from Local to Remote Instance

PuTTY: PuTTY is an open source terminal emulator. The main features are serial control and network file transfer application. It supports several network protocols such as SCP, SSH, raw socket, Telnet and rlogin. Figure 4.3 demonstrates the environmental setup of PuTTY for RSA key exchange.

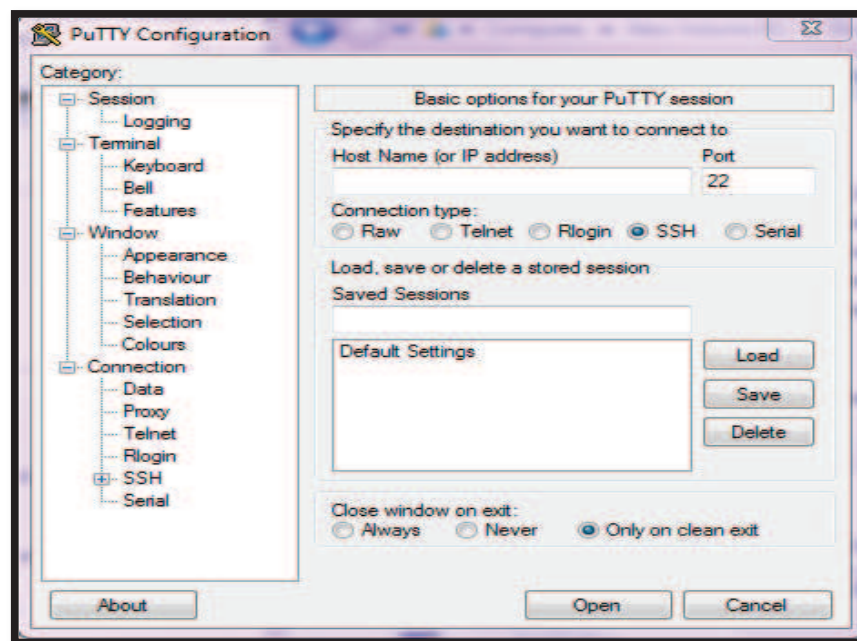
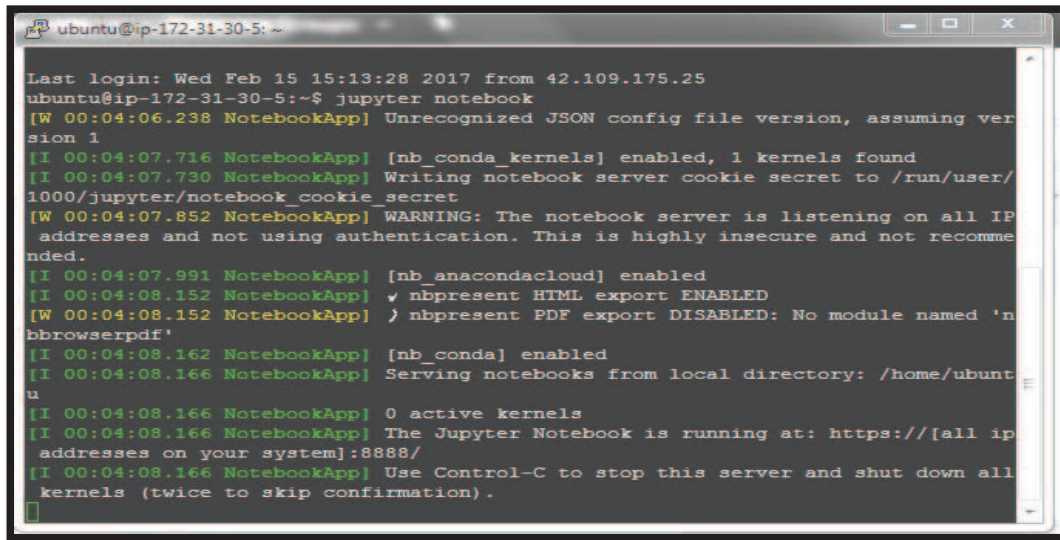


Figure – 4.3: PuTTY Environment for RSA Keys

A terminal window titled 'ubuntu@ip-172-31-30-5: ~' showing the output of the 'jupyter notebook' command. The logs indicate that the Jupyter Notebook server is starting, with messages about unrecognized JSON config file version, enabling nb_conda_kernels, writing the cookie secret, a warning about listening on all IP addresses, enabling nb_anacondacloud, enabling nbpresent HTML export, disabling nbpresent PDF export, enabling nb_conda, and serving notebooks from the local directory. The server is running at https://[all ip addresses on your system]:8888/.

```
ubuntu@ip-172-31-30-5: ~  
Last login: Wed Feb 15 15:13:28 2017 from 42.109.175.25  
ubuntu@ip-172-31-30-5:~$ jupyter notebook  
[W 00:04:06.238 NotebookApp] Unrecognized JSON config file version, assuming ver  
sion 1  
[I 00:04:07.716 NotebookApp] [nb_conda_kernels] enabled, 1 kernels found  
[I 00:04:07.730 NotebookApp] Writing notebook server cookie secret to /run/user/  
1000/jupyter/notebook_cookie_secret  
[W 00:04:07.852 NotebookApp] WARNING: The notebook server is listening on all IP  
addresses and not using authentication. This is highly insecure and not recomme  
nded.  
[I 00:04:07.991 NotebookApp] [nb_anacondacloud] enabled  
[I 00:04:08.152 NotebookApp] ✓ nbpresent HTML export ENABLED  
[W 00:04:08.152 NotebookApp] } nbpresent PDF export DISABLED: No module named 'n  
bbrowserpdf'  
[I 00:04:08.162 NotebookApp] [nb_conda] enabled  
[I 00:04:08.166 NotebookApp] Serving notebooks from local directory: /home/ubunt  
u  
[I 00:04:08.166 NotebookApp] 0 active kernels  
[I 00:04:08.166 NotebookApp] The Jupyter Notebook is running at: https://[all ip  
addresses on your system]:8888/  
[I 00:04:08.166 NotebookApp] Use Control-C to stop this server and shut down all  
kernels (twice to skip confirmation).
```

Figure – 4.4: Spark Connecting to Jupyter Notebook

PySpark is a framework that runs jobs on a Cluster and stores data in a distributed file system. Spark exposes a solitary data structure, the RDD (Resilient Distributed Dataset) which can be programmed in Python. The RDDs or the data-frame in Spark has numerous operations such as map, reduce, sort and filter etc. The results are stored in the memory to analyze the data rapidly. Figure 4.4 illustrates the connectivity setup of Spark tool with Jupyter notebook.



Figure – 4.5: Jupyter Notebook for Running Machine Learning Methods

Spark uses function pipeline to interpret the learning flow of Machine Learning techniques. This learning flow is implemented on datasets to process the training data for

attaining best test results. Figure 4.5 clearly explains the learning flow of machine learning methods through Jupyter Notebook.

4.3 TOOLS

The Machine Learning systems examine datasets and involuntarily capture the features while processing the test data. The training set consists of input data, features and desired output. The output of the function is a continuous value or prediction from a class label.

Scikit Learn is a Python Machine Learning library used for implementing the proposed work. It supports both supervised and unsupervised learning algorithms. Jupyter notebook is another tool that runs in Apache Spark framework. This Spark framework distributes jobs on virtual machines that runs Jupyter notebook implementing Machine Learning techniques.

Orange is open source software that helps to load and transform data. It is useful in applications like data mining, Machine Learning, predictive analytics and Statistical Modelling. This software evaluates the comparative results of the algorithms drawn from the proposed model.

4.4 PHASES OF PROPOSED MODEL

The proposed model consists of three important phases. Initial phase concentrates on data collection which encompasses of information and brief description of dataset. The algorithm representation followed by the work flow and implementation are described in phase II. The phase III exhibits the result and error depiction between the learning algorithms with a prime focus on the comparative result analysis.

- Phase I: Data Collection
- Phase II: Proposed Model and Algorithm
- Phase III: Comparative Result analysis

4.4.1 PHASE I: DATA COLLECTION

The dataset consists of Temperature data ‘All India seasonal Annual Temperature’ series extracted from https://data.gov.in/catalogs/ministry_department/india-meteorological-department-imd. It consists of annual temperature reading over 100 years which helps in predicting the temperature for the upcoming years. The extracted dataset contains both annual seasonal data with minimum and maximum temperature recorded over years. It comprises of

seasonal reading, Month-wise reading and annual reading which benefits the proposed work for processing the data to predict the future temperature. Table 4.2 represents the trained data based on the labeled features.

Table – 4.2: Temperature Dataset

YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	JAN-FEB	MAR-MAY	JUN-SEP	OCT-DEC
1904	17.77	19.39	22.95	26.73	27.83	27.85	26.84	26.73	25.84	24.36	21.07	18.84	23.86	18.66	25.84	26.83	21.42
1910	18.14	19.72	22.9	25.96	28.36	27.72	26.93	26.61	25.98	24.04	20.72	18.05	23.77	18.93	25.74	26.81	20.96
1911	18.52	19.18	22.05	26.0	28.55	28.02	27.44	27.04	26.22	24.57	21.1	18.76	23.96	18.85	25.53	27.18	21.48
1912	18.6	20.84	22.93	26.21	28.3	28.53	27.49	26.68	25.81	24.44	21.0	18.44	24.11	19.72	25.81	27.13	21.29
1922	18.3	20.5	23.65	26.55	29.8	28.03	27.26	26.95	26.04	23.94	21.28	18.39	24.05	19.4	25.88	27.07	21.2
1924	18.06	19.97	24.12	26.87	27.43	28.6	27.5	26.91	25.72	24.5	21.04	18.97	24.15	19.02	26.14	27.18	21.52
1940	18.1	19.88	22.23	25.88	28.38	28.29	27.42	26.44	26.19	24.74	22.03	18.97	24.06	19.03	25.49	27.08	21.91
1942	18.07	20.06	23.87	26.79	28.82	28.81	27.28	26.62	26.03	24.56	21.85	18.75	24.22	19.07	26.49	27.18	21.72
1943	18.32	19.47	23.31	25.56	27.82	27.8	27.16	26.66	26.26	24.2	21.81	19.02	23.93	18.89	25.57	26.97	21.68
1948	18.25	19.36	22.72	26.68	28.71	28.16	27.15	26.72	26.28	24.84	21.58	18.79	24.11	18.83	26.04	27.08	21.73
1960	18.31	21.13	22.65	26.17	28.29	28.5	27.33	27.2	26.33	24.58	21.35	19.59	24.29	19.72	25.7	27.34	21.84
1967	18.15	20.73	22.69	26.04	27.92	28.36	27.36	26.7	26.35	24.22	21.4	19.41	24.11	19.44	25.55	27.19	21.68
1980	18.81	20.47	23.24	27.55	28.94	28.25	27.33	26.93	26.55	25.12	22.15	19.23	24.55	19.64	26.58	27.26	22.17
1982	18.76	19.52	22.37	26.15	27.72	28.2	27.68	26.98	26.4	24.89	21.84	19.33	24.15	19.14	25.41	27.31	22.02
1986	18.24	20.0	23.53	26.53	27.75	28.19	27.3	26.75	26.43	24.48	21.89	19.38	24.2	19.12	25.94	27.17	21.9
1988	18.58	20.83	23.51	26.6	28.6	28.37	27.12	26.91	26.48	24.79	21.63	19.64	24.42	19.7	26.24	27.22	22.0
1991	17.99	20.43	23.58	26.13	28.35	28.17	27.59	26.88	26.65	24.56	21.6	19.44	24.28	19.21	26.02	27.32	21.87
2002	18.78	20.52	24.44	27.66	29.56	28.77	28.47	27.27	26.43	25.44	23.17	19.87	25.0	19.65	27.22	27.71	22.58
2006	19.96	23.02	23.91	26.83	28.82	28.13	27.92	27.15	26.42	25.41	22.59	19.99	25.06	21.33	26.52	27.4	22.66
2008	18.49	19.83	24.43	26.54	28.42	28.1	27.5	27.0	26.44	25.47	22.51	20.62	24.61	19.16	26.46	27.26	22.87

4.4.2 PHASE II: PROPOSED MODEL AND ALGORITHM

The proposed model illustrates task of the machine learning in predicting the value of the function for a valid input object having a large number of training data.

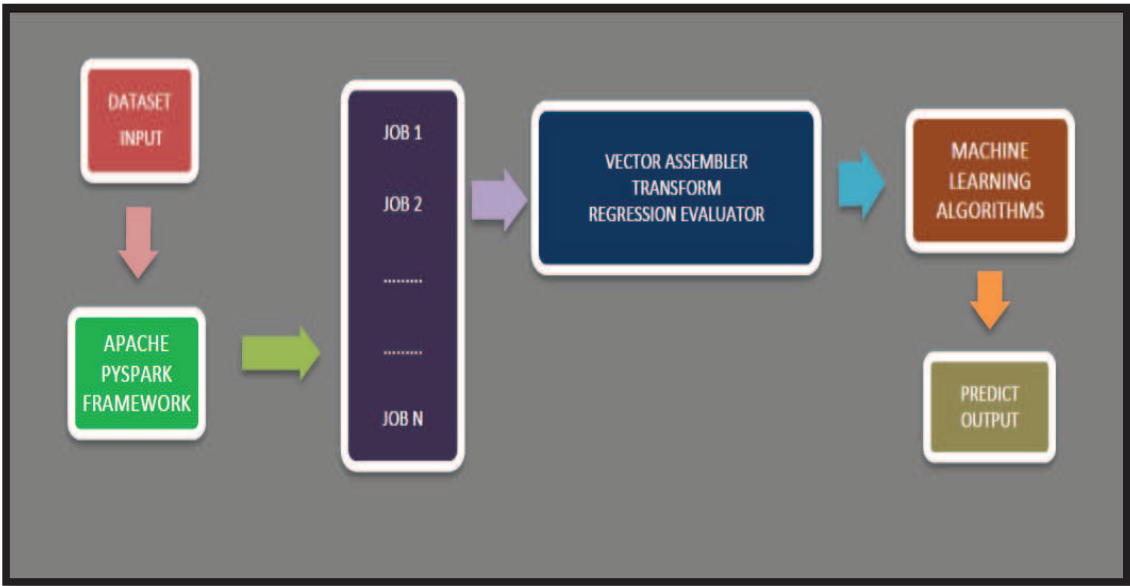


Figure – 4.6: Framework Model for Prediction on Datasets using ML Techniques

Analyzing such large datasets is a challenging task. Apparently to resolve these anonymities, the proposed system “**A Framework Model on Big Data Analytics Using Machine Learning Techniques for Prediction on Datasets**” incorporates the entities as depicted in figure 4.6

The prescribed model and algorithm shows the implementation of four Machine Learning regression algorithms for predictive analysis on data sets. The following are the machine learning regression algorithms:

- Linear Regression
- Decision Tree
- Random Forest
- Gradient Boosting Tree

4.4.2.1 *Algorithm for Machine Learning using PySpark Framework*

The following are the various steps in the algorithm for machine learning:

Step 1: Read the data into Spark data-frame and invoke time.

Step 2: Split the data as test data using randomSplit function from Spark library.

Step 3: Vector Assembler converts the data into terms of vectors.

Step 4: Transform function modifies the vectors into necessary data-frames.

Step 5: Map the labelcol and featurecol using Machine Learning algorithms from MLlib.

- Linear Regression
- Decision Tree
- Random Forest
- Gradient Boosting Tree

Step 6: Pipeline consists of stages transformer and estimator to process Machine Learning workflow.

Step 7: Fit the training data into the model for prediction.

Step 8: Regression Evaluator is used to evaluate the prediction on the featured data.

Step 9: RMSE is calculated to find the mean square error.

Step 10: Accurate prediction value is observed from different learning methods.

Step 11: Time and Space are the parameters experimental on Machine Learning algorithms.

4.4.2.2 Flowchart for the Proposed System

Flowchart in Figure 4.7 depicts the entire functionality of the proposed model. Primarily the dataset is distributed on a cluster, which consists of four machines. The represented data is converted into vector columns and transformed. Following the transformation step is the process of regression evaluator, which initiates the prediction outcome. Each prediction is taken from learning algorithms and evaluated for accuracy. The two frameworks MapReduce and Spark are examined on time and Space complexities for each learning methods.

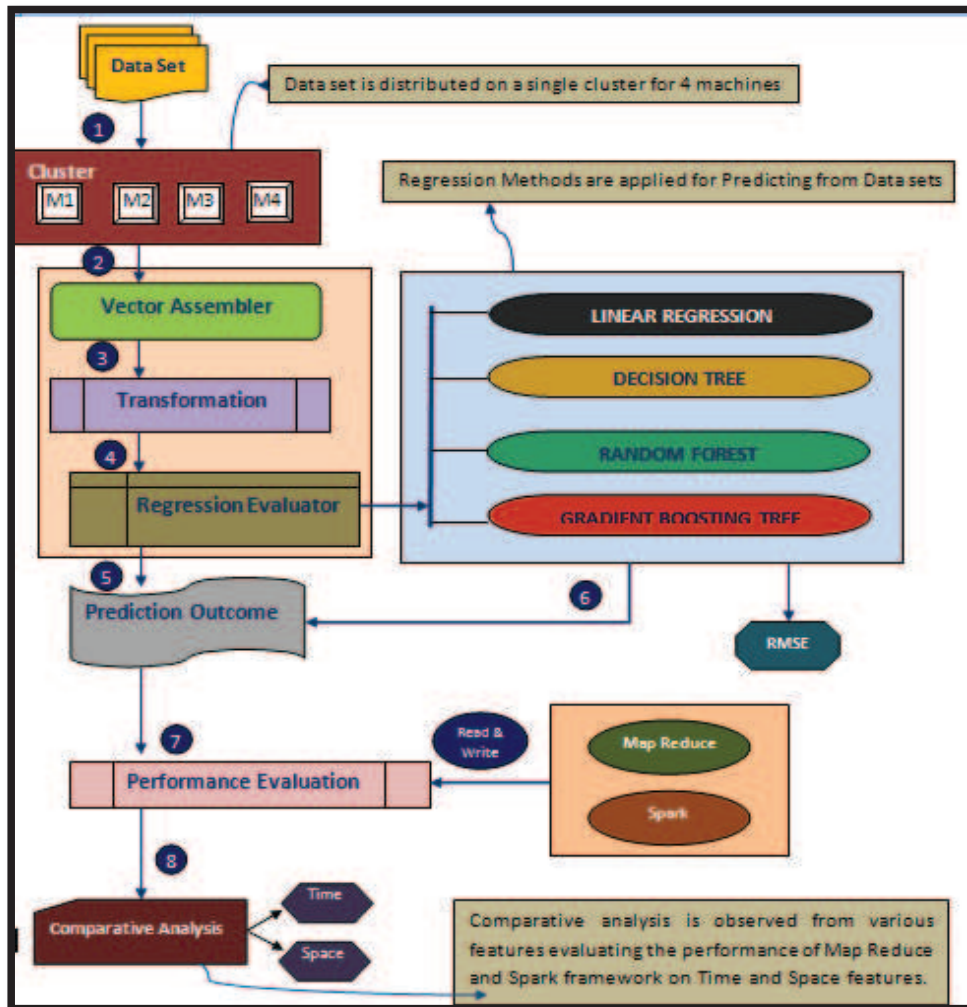


Figure – 4.7: Flowchart for the Functionality of the Proposed System

4.4.2.3 *Process of Transformation*

Dataset is distributed on the cluster of four machines. Data is preprocessed using transformation. The stages of transformation are:

- i. **Vector Assembler:** Vector Assembler class creates a copy of instance with unique identification and special parameters for processing. Basic parameters for the vector assembler method are input and output columns. The data which is read as vectors is processed into numerical values by vector assembler. This transforms multiple columns into vector columns by using assembler function. The function 'VectorAssembler' and required packages are imported from Spark Machine Learning library (MLLib). Link: *org.apache.spark.ml.feature.VectorAssembler*.
- ii. **Pipelining:** A pipeline has sequence of stages and each stage is either a transformer or an estimator. These stages run in order and input data-frame. This data-frame modifies itself as it completes each stage. The *transform()* method is called in the initial stage on the data-frame. *Fit()* method is called in the estimator stage to produce a transformer on the data-frame.
- iii. **Transformer:** A transformer comprises of transform method, which converts Data-frame into another by appending one or more columns. The model understands data-frames, which contains feature vectors and predictive labels for every feature vector. The output data-frame is appended as a column to generate predictive labels. This data is partitioned in this step using *randomSplit* as trained test data.
- iv. **Estimator:** An estimator abstracts the concept of learning algorithm that fits on training data. It fits the data-frame for the Machine Learning model to produce fine estimator from the test dataset. This implements a method *fit()*, which accepts data-frame and produce an appropriate model.
- v. **Regression Evaluator:** The regression evaluator computes the Random Mean Square Error, Mean Square Error and r^2 for coefficient of correlation for prediction or observations on a data set.

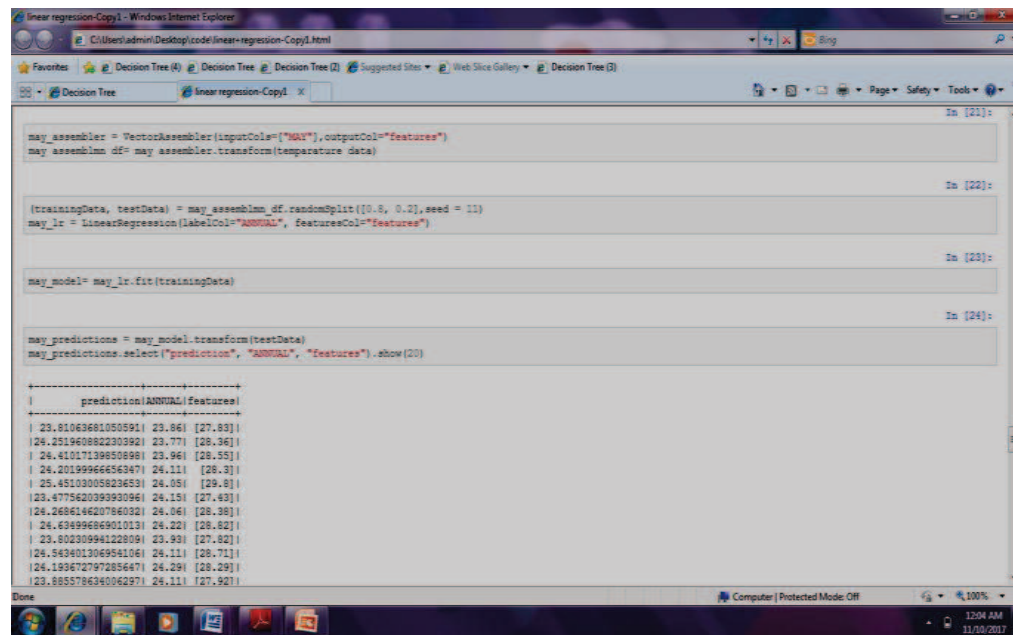
org.apache.spark.ml.evaluation

RegressionEvaluator(labelCol="ANNUAL",predictionCol="prediction",metricName="rmse")

4.4.2.4 Machine Learning Models

Machine Learning models incorporate statistical techniques for handling regression and classification tasks with multiple dependent and independent variables. The following methods are some of the supervised learning regression techniques used for predicting outcomes:

- i. **Linear Regression:** Linear Regression is one of the prediction evaluated by supervised learning method. A detailed description on Linear Regression algorithm is discussed in chapter 3. Figure 4.8 illustrates the Jupyter notebook for Linear Regression realization.



```
may_assembler = VectorAssembler(inputCols=["MAX"], outputCol="features")
may_assemblm_df = may_assembler.transform(temperature_data)

(trainingData, testData) = may_assemblm_df.randomSplit([0.8, 0.2], seed = 11)
may_lr = LinearRegression(labelCol="ANNUAL", featuresCol="features")

may_model = may_lr.fit(trainingData)

may_predictions = may_model.transform(testData)
may_predictions.select("prediction", "ANNUAL", "features").show(20)
```

	prediction	ANNUAL	features
1	23.81063661050591	23.86	[27.83]
2	26.25196082230592	23.77	[28.36]
3	24.41017139650698	23.96	[28.55]
4	24.2019996656347	24.11	[28.3]
5	25.45103005823653	24.05	[29.8]
6	23.47756200993096	24.15	[27.63]
7	24.268614620786032	24.06	[28.38]
8	24.63499686901013	24.22	[28.62]
9	23.80230996122808	23.93	[27.62]
10	26.56360130495406	24.11	[28.71]
11	24.183672797285647	24.29	[28.29]
12	23.885378634006297	24.11	[27.92]

Figure – 4.8: Jupyter Notebook for Linear Regression

- ii. **Decision Tree:** Decision tree is a tree based learning algorithm for prediction. It has two variations, namely, *Classification* and *Regression*. Classification generates a tree structure by interpreting the label features. Regression is used for prediction analytics. An elaborate description on decision tree is explained in chapter 3. Figure 4.9 describes the implementation of decision tree on a Jupyter notebook.

```

may_assembler = VectorAssembler(inputCols=["WAT"], outputCol="Features")
may_assemblm_df= may_assembler.transform(temperature_data)

(trainingData, testData) = may_assemblm_df.randomSplit([0.8, 0.2], seed = 11)
dt = DecisionTreeRegressor(labelCol="ANNUAL", featuresCol="Features")

dt.model = dt.fit(trainingData)

m1 = dt.model.toDebugString

m1

'DecisionTreeRegressionModel (uid=DecisionTreeRegressor_458780240b71a0043b54) of depth 5 with 20 nodes\n
If (feature 0 <= 27.04)\n  Predict: 15.825733333333333\n
Else (feature 0 > 27.04)\n  If (feature 0 <= 28.41)\n    If (feature 0 <= 28.07)\n      If (feature 0 <= 27.57)\n        If (feature 0 <= 27.31)\n          Predict: 24.080000000000002\n        Else (feature 0 > 27.31)\n          Predict: 24.169999999999999\n      Else (feature 0 > 28.07)\n        If (feature 0 <= 27.7)\n          Predict: 2\n        Else (feature 0 > 27.7)\n          Predict: 24.079523809523806\n    Else (feature 0 > 28.41)\n      If (feature 0 <= 28.13)\n        If (feature 0 <= 28.12)\n          Predict: 24.256666666666666\n        Else (feature 0 > 28.12)\n          Predict: 26.276666666666667\n      Else (feature 0 > 28.13)\n        If (feature 0 <= 28.1801)\n          Predict: 23.8818\n        Else (feature 0 > 28.1801)\n          Predict: 24.143700000000004\n  Else (feature 0 > 28.41)\n    If (feature 0 <= 29.06)\n      If (feature 0 <= 28.82)\n        If (feature 0 <= 28.48)\n          Predict: 24.375000000000004\n        Else (feature 0 > 28.48)\n          Predict: 24.406399999999999\n      Else (feature 0 > 28.82)\n        If (feature 0 <= 28.97)\n          Predict: 24.043333333333333\n        Else (feature 0 > 28.97)\n          Predict: 24.5033\n    Else (feature 0 > 29.06)\n      If (feature 0 <= 29.19)\n        Predict: 24.726666666666663\n      Else (feature 0 > 29.19)\n        Predict: 24.783333333333328\n'

```

Figure –4.9: Jupyter Notebook for Decision Tree

iii. **Random Forest:** Group of decision trees form Random forest. Random Forest is applied for deep analysis on the input data for accurate outcome. Learning process of this algorithm is discussed in chapter 3. Figure 4.10 describes the implementation of Random Forest on a Jupyter notebook.

```

may_assembler = VectorAssembler(inputCols=["WAT"], outputCol="Features")
may_assemblm_df= may_assembler.transform(temperature_data)

(trainingData, testData) = may_assemblm_df.randomSplit([0.8, 0.2], seed = 11)
#RandomForestRegressor in Spark

rf = RandomForestRegressor(labelCol="ANNUAL", featuresCol="Features")

rf_model= rf.fit(trainingData)

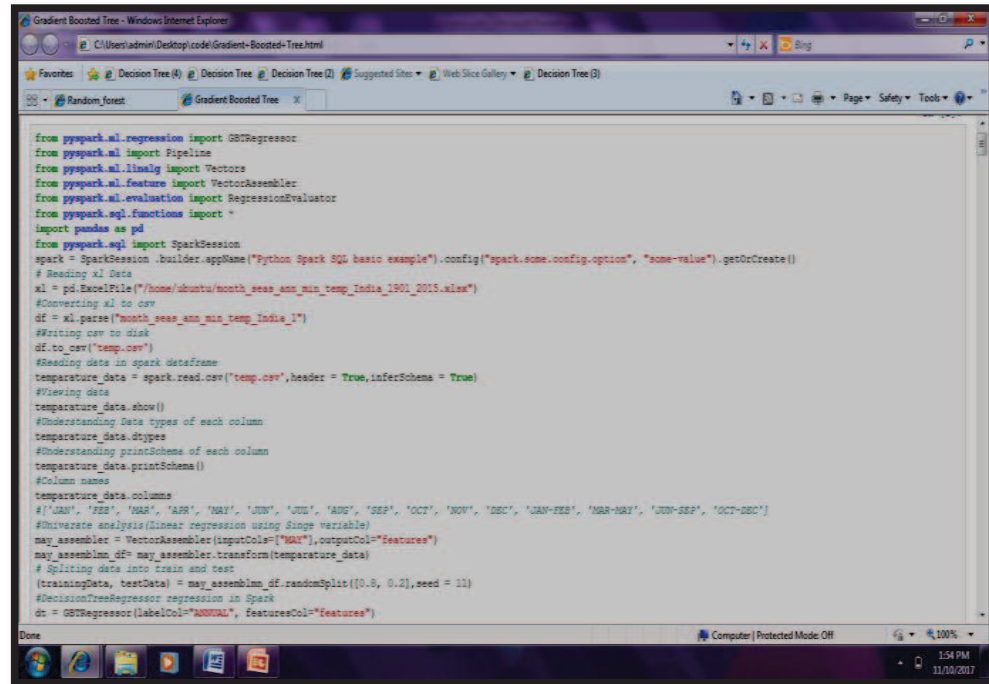
rf_predictions = rf_model.transform(testData)
rf_predictions.select("prediction", "ANNUAL", "Features").show(5)
# Checking model result
rf_evaluator = RegressionEvaluator(labelCol="ANNUAL", predictionCol="prediction", metricName="rmse")
rf_one_feature_rmse = rf_evaluator.evaluate(rf_predictions)
print("Root Mean Squared Error (RMSE) on test Data = %g" % rf_one_feature_rmse)

-----+-----+-----+
| prediction|ANNUAL|Features|
-----+-----+-----+
| 19.14476839680194| 19.22| [23.02]|
| 19.272464645939492| 19.01| [23.22]|
| 19.434326191051298| 19.31| [23.34]|
| 19.289731388164056| 19.27| [23.3]|

```

Figure – 4.10: Jupyter Notebook for Random Forest

- iv. **Gradient Boosting Tree:** Gradient boosting is a voracious algorithm which over fits training objects rapidly. This robustness increases in congestion with training sets. A detailed venture about the Gradient boosting is explained in chapter 3. A weak hypothesis described as performance improvement is shown by boosting instead choosing a random chance. Figure 4.11 describes the implementation of Random Forest on a Jupyter notebook.



```
from pyspark.ml.regression import GBRegressor
from pyspark.ml import Pipeline
from pyspark.ml.linalg import Vectors
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.sql.functions import *
import pandas as pd
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("Python Spark SQL basic example").config("spark.some.config.option", "some-value").getOrCreate()

# Reading xl Data
xl = pd.ExcelFile("/home/ubuntu/month_wise_ann_min_temp_india_1901_2015.xlsx")

#Converting xl to csv
df = xl.parse("month_wise_ann_min_temp_india_1")
#Writing csv to disk
df.to_csv("temp.csv")

#Reading data in spark dataframe
temperature_data = spark.read.csv("temp.csv", header = True, inferSchema = True)

#Viewing data
temperature_data.show()

#Understanding Data types of each column
temperature_data.dtypes

#Understanding printSchema of each column
temperature_data.printSchema()

#Column names
temperature_data.columns

#Univariate analysis(Linear regression using single variable)
may_assembler = VectorAssembler(inputCols=["MAY"], outputCol="Features")
may_assembled_df = may_assembler.transform(temperature_data)

# Splitting data into train and test
(trainingData, testData) = may_assembled_df.randomSplit([0.8, 0.2], seed = 11)

#DecisionTreeRegressor regression in Spark
dt = GBRegressor(labelCol="ANNUAL", featuresCol="Features")
```

Figure – 4.11: Jupyter Notebook for Gradient Boosting Tree

4.4.3 PHASE III: COMPARATIVE RESULT ANALYSIS

The jobs are split in the Apache Spark environment and the Machine Learning algorithms are applied on the jobs for prediction based on the training datasets. Each algorithm is tested by computing time as a parameter. Eventually, time, space efficiencies and best prediction are obtained for further evaluations. The prediction and time parametric constraints are calculated for efficient learning methods.

Table 4.3 shows the prediction value of random forest method considering annual as feature label for various years.

Table – 4.3: Annual and Prediction Values for Various Years

prediction	ANNUAL	features
23.81063681050591	23.86	[27.83]
24.251960882230392	23.77	[28.36]
24.41017139850898	23.96	[28.55]
24.20199966656347	24.11	[28.3]
25.45103005823653	24.05	[29.8]
23.477562039393096	24.15	[27.43]
24.268614620786032	24.06	[28.38]
24.63499686901013	24.22	[28.82]
23.80230994122809	23.93	[27.82]
24.543401306954106	24.11	[28.71]
24.193672797285647	24.29	[28.29]
23.885578634006297	24.11	[27.92]
24.734919300343975	24.55	[28.94]
23.719041248449884	24.15	[27.72]
23.744021856283346	24.2	[27.75]
24.45180574489808	24.42	[28.6]
24.243634012952572	24.28	[28.35]
25.251185195568837	25.0	[29.56]
24.63499686901013	25.06	[28.82]
24.301922097897315	24.61	[28.42]

The prediction is generated from the proposed model based on the training dataset. The efficiency of the model is evaluated based on the generated outcomes. It is achieved by computing the Random Mean Square error. If the error is less, then the prediction drawn is effectual and accurate.

Figure 4.12 shows the status of jobs executed successfully. It indicates the jobs and various stages of data partitioning on the cluster. Job Group describes the jobs on the cluster and unique identification. To accomplish a task in a machine the created multiple jobs are distributed on the maps, which run parallelly.



Figure – 4.12: Spark Cluster for the Job Status

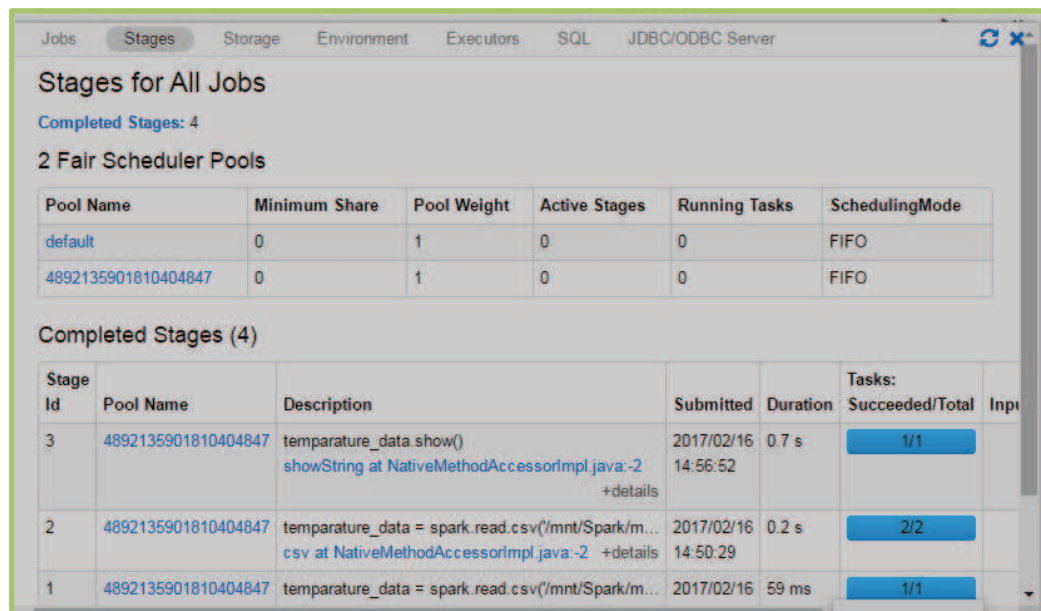


Figure – 4.13: Stages of Jobs on a Machine

Fair scheduler distributes the jobs evenly on machines to avoid overloading. Figure 4.13 illustrates various stages and distribution of jobs on a machine. This interprets the time used for various tasks assigned on a cluster.

In this research work, the proficiency of computed results is based on the time and space parametric values on various Machine Learning techniques. MapReduce programming tool is compared with Spark framework. The various hindrances such as high disk rates, low

efficiency and unexpected failures faced in MapReduce tool are addressed by Spark Framework. These faults are handled by the Spark framework for an improvised performance when compared with MapReduce.

4.5 COMPARISON OF LEARNING ALGORITHMS

The observed values of learning algorithms are compared across various attributes using Orange tool. Linear Regression, Random Forest and Decision tree algorithms are compared using two major techniques (i) cross validation and (ii) random sampling.

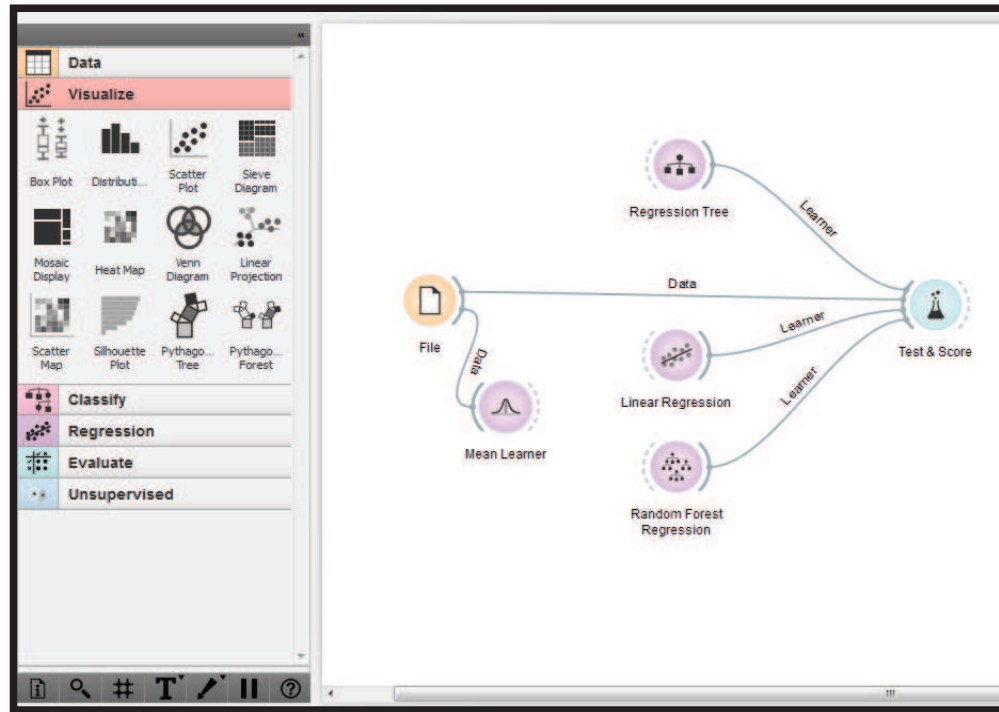


Figure – 4.14: Comparison of Algorithms using Orange Tool

Figure 4.14 illustrates the canvas of orange tool that considers the test data for comparing the algorithms. Test scores computed assigns the number of folds as 10.

Cross validation is evaluated on Linear Regression, Random Forest Regression and Regression Tree (Decision Tree) for a stratified sampling method. The computations are carried out for MSE, RMSE, MAE and R2.

Linear Regression result in perfect positive correlation between the annual and prediction features from the model by computing $R^2=1.00$. Random Forest Regression shows no correlation between the features and lastly regression tree shows negative correlation.

Mean Square error is more for regression tree when compared with Random forest. Figure 4.15 demonstrates the test score of various algorithm based on the cross validation.

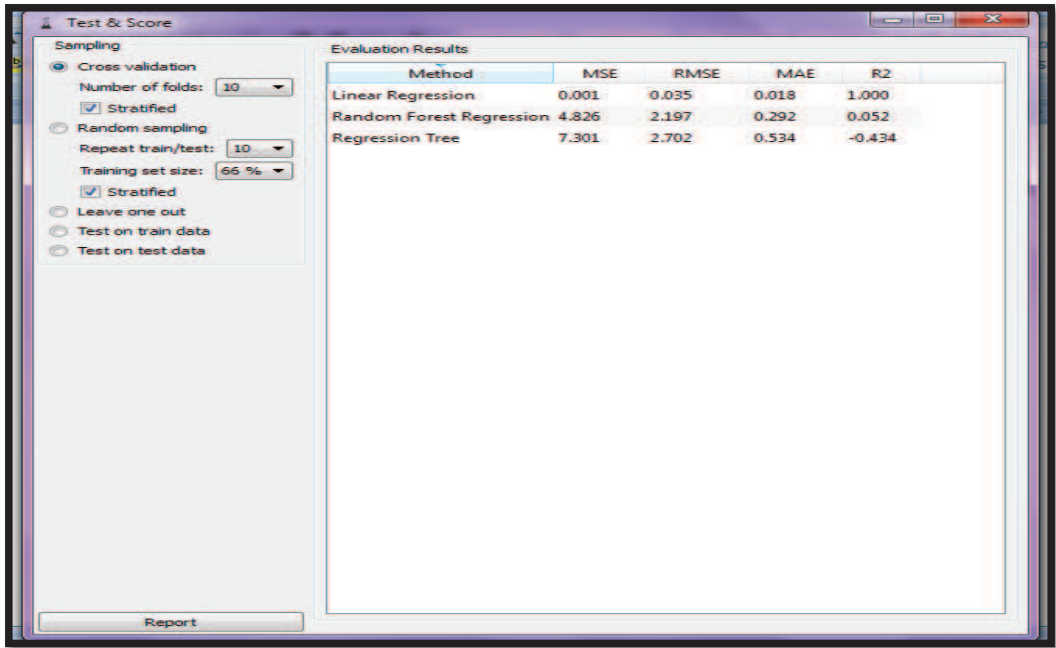


Figure – 4.15: Test Score of Algorithms based on Cross Validation

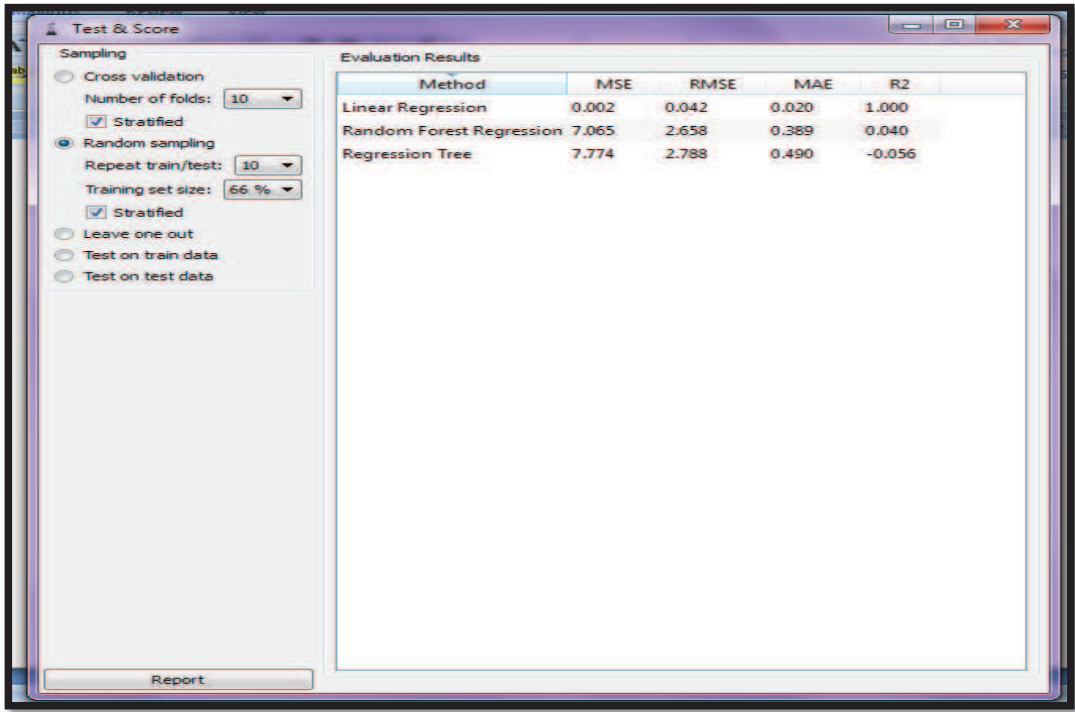


Figure – 4.16: Test Score of Algorithms based on Random Sampling

Random sampling is the other category for analyzing the learning algorithms using stratified sampling. MSE, RMSE, MAE and R2 are the measures taken for considerations.

Random forest and Regression tree evaluates MSE and RMSE with 7.065, 7.774 and 2.568, 2.788 respectively. Figure 4.16 shows the test score of various algorithms based on Random Sampling.

The calculated error rate from the prescribed model shows that Linear Regression shows minimum error rate whereas Regression tree proves to compute at high error rate.

	ANNUAL	YEAR	Regression Tree	Linear Regression	Random Forest Regression
1	23.710	1905	23.990	23.520	24.029
2	23.960	1911	23.870	23.961	24.046
3	24.110	1912	24.150	24.111	24.046
4	23.940	1927	23.903	23.947	23.976
5	23.980	1937	24.040	23.950	24.002
6	24.650	1941	24.437	24.677	24.530
7	23.930	1943	23.823	23.937	23.978
8	23.920	1945	24.085	23.885	24.098
9	24.130	1954	24.080	24.111	24.119
10	24.410	1973	24.407	24.406	24.446
11	24.200	1986	24.140	24.196	24.152
12	25.290	1995	25.100	25.284	24.883

Figure – 4.17: Prediction Values from Learning Algorithms

Figure 4.17 clearly shows the Prediction values drawn from Regression Tree, Linear Regression and Random forest regression which are compared with annual values of respective years. Accurate values nearest to the annual are Random Forest Regression values. Hence the best prediction values are computed through Random Forest regression algorithms.

4.6 SUMMARY

This chapter explained in detail an appropriate model for prediction using Machine Learning algorithms. An elaborate narration on various active phases that support the functionality of the model is discussed in detail. A comparative analysis is conducted and evaluated across various learning algorithms. The annual features and the learning algorithms are examined for accurate prediction values. The Spark and MapReduce framework are examined across typical job distribution for time and space complexities.

CHAPTER - 5

RESULTS AND ANALYSIS

5.1 INTRODUCTION

This chapter discusses the experimental results and observations of the devised model, which predicts the future values through learning algorithms. The developed model is the amalgamation of different phases constructed upon different platforms. Each implemented phase drives the model for accurate prediction. The accuracy of each learning method used in the model has different functionality. Therefore, the prediction results and observations obtained reflect the performance at each phase is clearly explained in this chapter. Regression techniques determine the accuracy of the prediction model through various dimensions. The dimensions for evaluating the performance of the developed model are prediction outcome, Time and Space complexity and Random Mean Square error (RMSE).

5.2 SCRUTINY OF THE SYSTEM

To examine the accuracy of the system major regression models for prediction are equipped on two different frameworks such as MapReduce and Spark by analytical approach. The overall performance is evaluated by computing the complexities on time and space.

5.3 DATASET ANALYSIS

The dataset used for this research work contains temperature related information. It comprises of annual temperature computed from the month of January to December. It also includes seasonal variations occurred in various months of a year. These are categorized as January to February, March to May, June to September and October to December. The dataset measured for analysis has hundred and fifteen years of temperature data from the year 1904 to 2016. This information is applied for computing and predicting the future years. Table 5.1 consists of temperature dataset that is used as training data set. Machine Learning algorithms are trained on this data set to predict the values for later years. Figure 5.1 graphically represents the annual temperature data for various years.

Table – 5.1: Annual Temperature Data for Various Years

Year	ANNUAL	1960	19.64
1904	19.51	1964	19.34
1908	19.44	1968	19.02
1912	19.25	1972	19.02
1916	19.22	1976	19.37
1920	19.03	1980	19.07
1924	19.51	1984	19.54
1928	19.08	1988	19.32
1932	19.09	1992	19.36
1936	19.13	1996	19.52
1940	19.01	2000	19.24
1944	19.31	2004	19.37
1948	19.27	2008	19.3
1952	19.09	2012	19.61
1956	19.41	2016	19.4

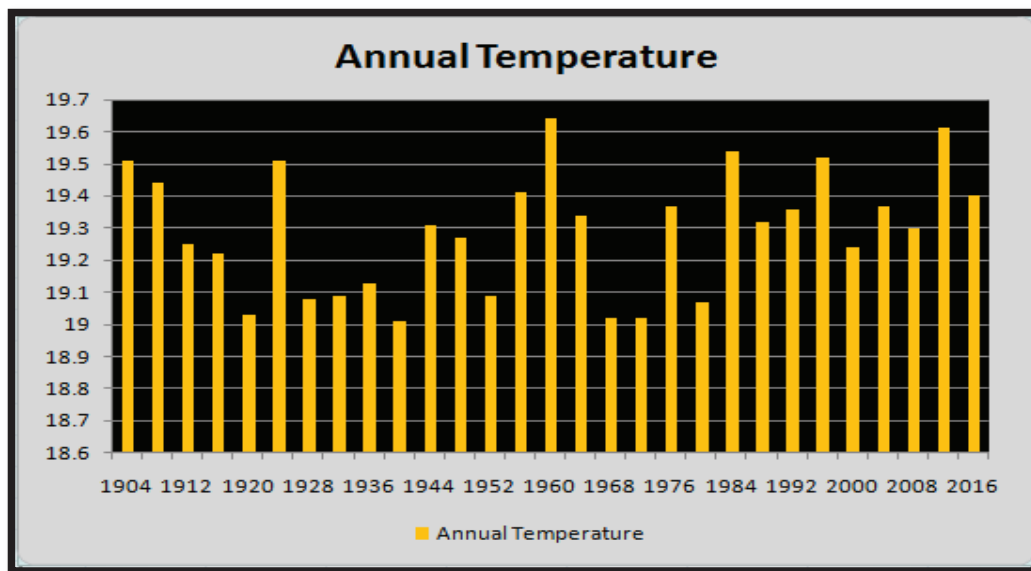


Figure – 5.1: Year-wise Annual Temperature Data

5.4 MACHINE LEARNING ALGORITHMS FOR PREDICTION ON DATASETS

The model proposed in this research is implemented on various Machine Learning algorithms. The predictions obtained from the learning algorithms are illustrated through graphical representations. The various Machine Learning algorithms used for prediction are:

- Linear Regression
- Decision Tree
- Random Forest
- Gradient Boosting Tree

5.4.1 LINEAR REGRESSION ALGORITHM

The model is implemented using the linear regression algorithm. The graphical representation of data by comparing the annual temperature with prediction is shown in the figure 5.2.

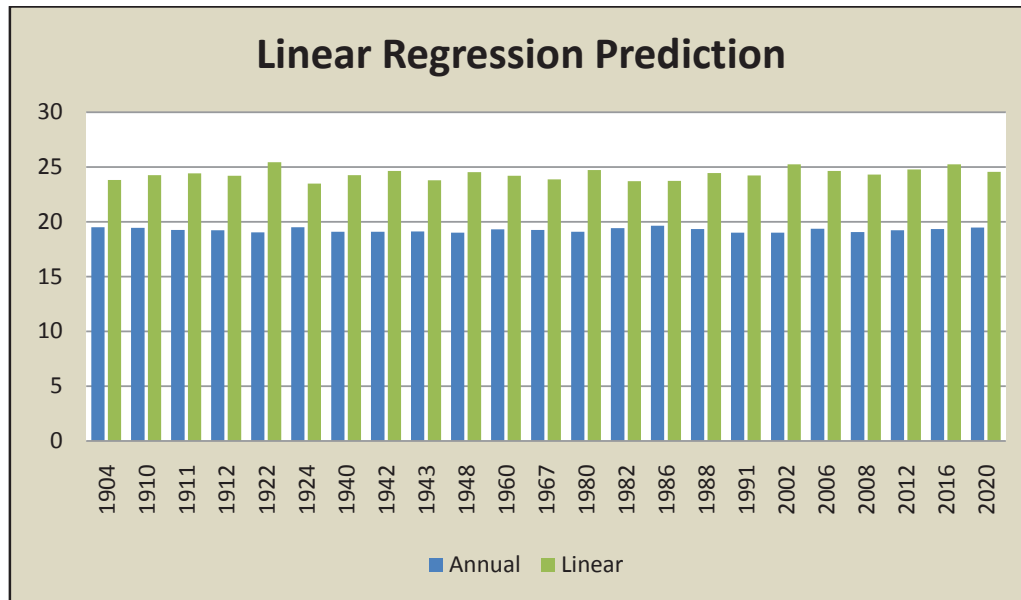


Figure – 5.2: Annual Temperature and Prediction Representation using Linear Regression Algorithm

Observation: Annual Temperature is compared against the prediction using linear regression with an accuracy of 72.5%. The computations drawn from Linear Regression algorithms are predictions and RMSE. The obtained prediction measures are less accurate and the graphical mapping shows wide deviations, which results in higher error rate with minimal utilization of time and space. The complexities faced by this algorithm are discussed in the section 5.5

Figure 5.3 illustrates the annual temperature and prediction for various years. It represents the annual temperature taken from the dataset where the prediction values are computed from the linear regression algorithm. The prediction values deviates showing large variance which results in higher error rate.

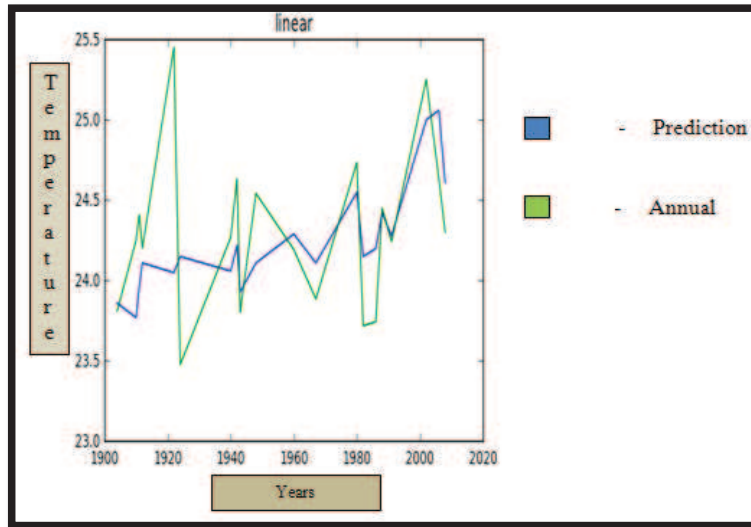


Figure – 5.3: Annual and Prediction Value for Linear Regression Algorithm

5.4.2 DECISION TREE

The tree structure is obtained through the prediction outcomes using Decision Tree algorithm. The graphical representation as in figure 5.4 illustrates the variations.

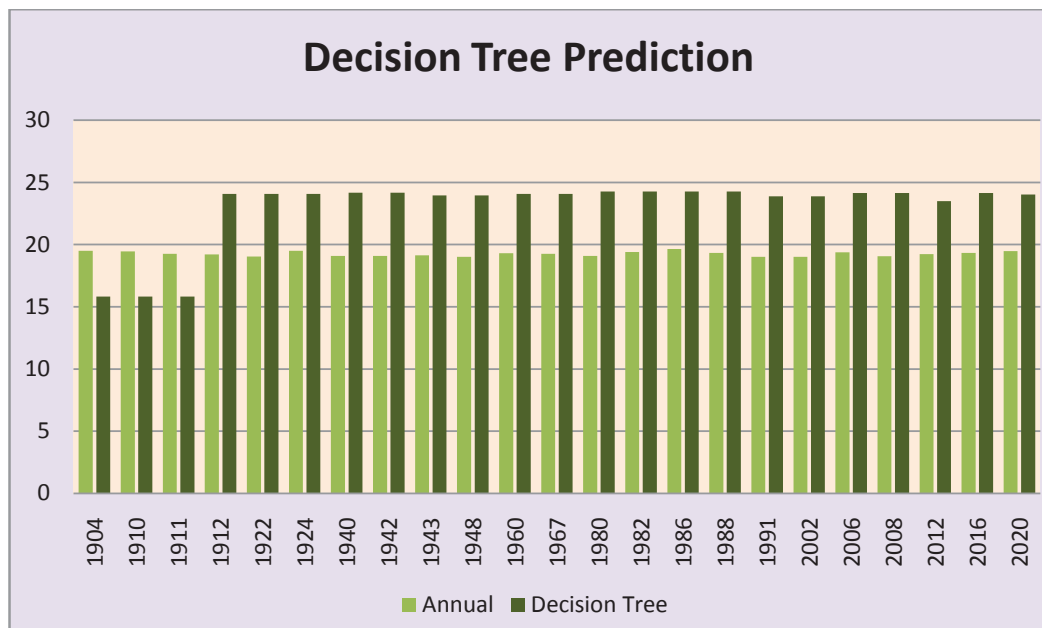


Figure – 5.4: Annual Temperature and Prediction Representation using Decision Tree Algorithm

Observation: Annual Temperature is compared against the prediction using Decision Tree Regression algorithm with an accuracy of 82.7%. The computations drawn from Decision

Tree algorithms are predictions and RMSE. The obtained prediction measures are nearer to accuracy and graphical representation shows deviations with minimal error rate. Large space is utilized by the algorithms. The complexities faced by this algorithm are discussed in the section 5.5. Figure 5.5 shows small branch of the tree with depth 5 and 27 nodes.

```

m1 = my_features_model.toDebugString
m1

Out[18]:
'DecisionTreeRegressionModel (uid=DecisionTreeRegressor_45f
989823a43ff027f4e) of depth 5 with 37 nodes\n If (feature
13 <= 24.89)\n If (feature 0 <= 17.33)\n Predict: 0.00
72\n Else (feature 0 > 17.33)\n If (feature 0 <= 18.1
6)\n Predict: 23.56\n Else (feature 0 > 18.16)\n
Predict: 23.87\n Else (feature 13 > 24.89)\n If (feature
12 <= 19.22)\n If (feature 15 <= 21.77)\n If (featur
e 13 <= 25.81)\n If (feature 11 <= 18.86)\n Pred
ict: 23.825945454545458\n Else (feature 11 > 18.86)\n
Predict: 23.98375\n Else (feature 13 > 25.81)\n If
(feature 12 <= 19.04)\n Predict: 24.013333333333335\n
Else (feature 12 > 19.04)\n Predict: 24.1599999999999
86\n Else (feature 15 > 21.77)\n If (feature 13 <= 2
5.93)\n If (feature 13 <= 25.34)\n Predict: 23.9
833333333333334\n Else (feature 13 > 25.34)\n Pre
dict: 24.1175\n Else (feature 13 > 25.93)\n If (fe
ature 1 <= 20.2)\n Predict: 24.24\n Else (featur
e 1 > 20.2)\n Predict: 24.460000000000008\n Else (f
eature 12 > 19.22)\n If (feature 8 <= 26.56)\n If (f
eature 13 <= 25.58)\n If (feature 3 <= 25.76)\n
Predict: 24.013333333333332\n Else (feature 3 > 25.76)
\n Predict: 24.210000000000008\n Else (feature 13
> 25.58)\n If (feature 11 <= 19.55)\n Predict: 2
4.35818181818182\n Else (feature 11 > 19.55)\n P
redict: 24.554285714285715\n Else (feature 8 > 26.56)\n
If (feature 13 <= 26.85)\n If (feature 10 <= 22.14)\n
Predict: 24.566666666666663\n Else (feature 10 > 22.1
4)\n Predict: 24.747000000000003\n Else (feature
13 > 26.85)\n If (feature 0 <= 18.87)\n Predict:
24.74\n Else (feature 0 > 18.87)\n Predict: 25.1
766666666666666\n'

```

Figure – 5.5: Decision Tree Computed from Temperature Data

The tree shows the prediction values through various comparisons. Based on the seasonal and monthly temperature data the learning algorithms comprehend the features of the data set. These trained features are used for the comparison between annual temperature and prediction.

5.4.3 RANDOM FOREST

The tree structure is produced from the prediction analysis using Random Forest algorithm. The following graphical representation shows the variations. The forest is drawn from several decision trees. Using random forest method features are measured as the average impurities. These impurities decrease on computing output from all decision trees in the forest. It is achieved without any assumptions related to data. Ranks are computed from different features, which are normalized with their relative significance. It implements a transformation method that select features based on threshold specified after model fitting.

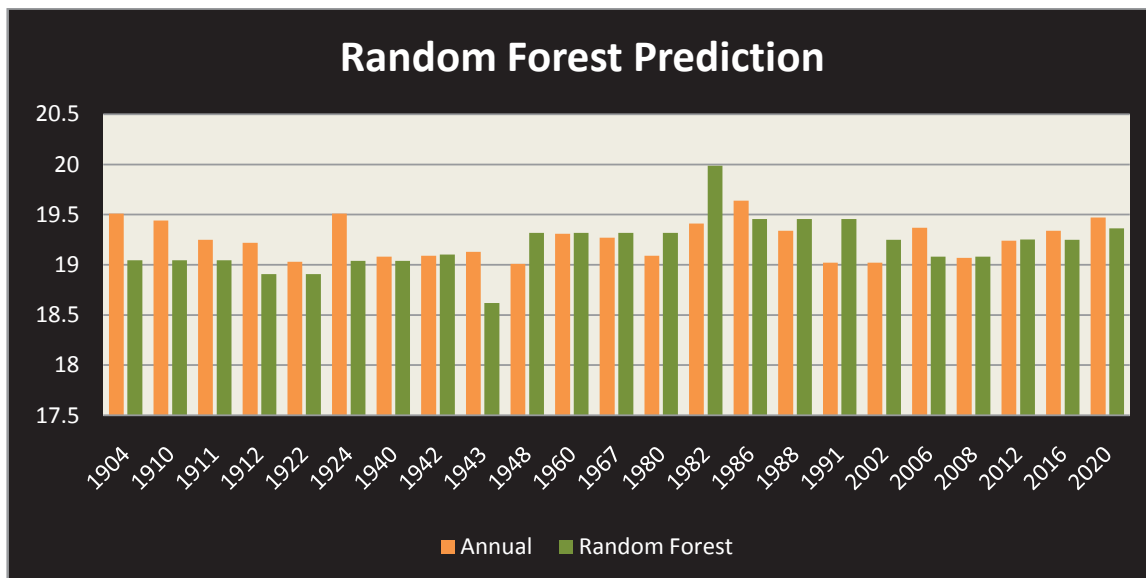


Figure – 5.6: Annual Temperature and Prediction Representation using Random Forest Algorithm

Observations: Annual Temperature is compared against the prediction using Random Forest with an accuracy of 95.33%. Random forest that shows pertinent prediction to the annual temperature is illustrated in figure 5.6. The prediction outcome of random forest is obtained after constructing a forest of 20 decision trees. Space complexity is higher in ratio for Random Forest technique.

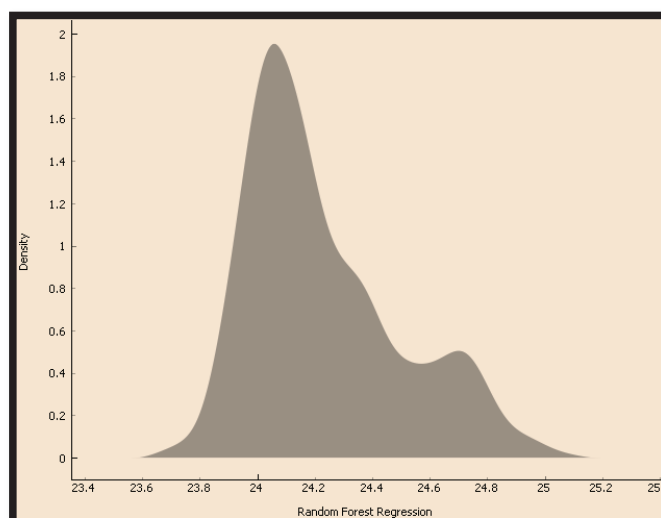


Figure – 5.7: Random Forest Regression

Figure 5.7 represents the frequency range of the prediction outcomes. Figure 5.8 illustrates the trees of random forest along with the prediction outcome.

```

Out[22]: RandomForestRegressionModel (uid=rfr_09249f98671e) with 20 trees\n Tree 0 (weight 1.0):\n If (feature 0 <= 22.51)\n If (feature 0 <= 22.41)\n If (feature 0 <= 22.13)\n Predict: 19.045\n Else (feature 0 > 22.13)\n Predict: 18.987999999999998\n Else (feature 0 > 22.41)\n If (feature 0 <= 22.47)\n Predict: 19.04\n Else (feature 0 > 22.47)\n Predict: 19.183333333333335\n Else (feature 0 > 22.51)\n Predict: 18.520000000000005\n Else (feature 0 > 22.66)\n If (feature 0 <= 23.5)\n If (feature 0 <= 23.41)\n If (feature 0 <= 23.03)\n If (feature 0 <= 22.98)\n Predict: 19.317999999999998\n Else (feature 0 > 22.98)\n Predict: 19.988333333333333\n Else (feature 0 > 23.03)\n If (feature 0 <= 23.18)\n Predict: 19.456666666666667\n Else (feature 0 > 23.18)\n Predict: 19.249166666666666\n Else (feature 0 > 23.41)\n If (feature 0 <= 23.46)\n Predict: 19.086666666666666\n Else (feature 0 > 23.46)\n Predict: 19.249999999999998\n Else (feature 0 > 23.5)\n If (feature 0 <= 23.54)\n Predict: 19.720000000000002\n Else (feature 0 > 23.54)\n If (feature 0 <= 23.63)\n Predict: 19.283333333333335\n Else (feature 0 > 23.63)\n If (feature 0 <= 24.01)\n Predict: 19.536923876923874\n Else (feature 0 > 24.01)\n Predict: 19.614999999999995\n Tree 1 (weight 1.0):\n If (feature 0 <= 23.3)\n If (feature 0 <= 22.66)\n If (feature 0 <= 22.3)\n If (feature 0 <= 22.13)\n Predict: 19.02\n Else (feature 0 > 22.13)\n Predict: 19.328333333333337\n Else (feature 0 > 22.3)\n If (feature 0 <= 22.41)\n Predict: 18.83\n Else (feature 0 > 22.41)\n If (feature 0 <= 22.47)\n Predict: 19.21\n Else (feature 0 > 22.47)\n Predict: 19.017777777777778\n Else (feature 0 > 22.66)\n If (feature 0 <= 23.18)\n If (feature 0 <= 23.1)\n If (feature 0 <= 22.92)\n Predict: 19.342727272727272\n Else (feature 0 > 23.1)\n If (feature 0 <= 23.17)\n Predict: 19.655\n Else (feature 0 > 23.17)\n Predict: 19.605000000000004\n Else (feature 0 > 23.18)\n If (feature 0 <= 23.26)\n Predict: 19.083333333333332\n Else (feature 0 > 23.26)\n Predict: 19.16\n Else (feature 0 > 23.3)\n If (feature 0 <= 23.54)\n If (feature 0 <= 23.41)\n If (feature 0 <= 23.33)\n Predict: 19.511666666666667\n Else (feature 0 > 23.33)\n Predict: 19.566666666666666\n Else (feature 0 > 23.41)\n If (feature 0 <= 23.5)\n Predict: 19.369999999999997\n Else (feature 0 > 23.5)\n Predict: 19.485454545454544\n Else (feature 0 > 23.54)\n If (feature 0 <= 23.56)\n Predict: 19.25\n Else (feature 0 > 23.56)\n Predict: 19.253000000000003\n Else (feature 0 > 23.63)\n If (feature 0 <= 23.96)\n If (feature 0 <= 23.91)\n If (feature 0 <= 23.68)\n Predict: 19.789999999999994\n Else (feature 0 > 23.91)\n Predict: 19.04\n Else (feature 0 > 23.68)\n Predict: 19.789999999999994\n Else (feature 0 > 23.91)\n Predict: 19.400000000000001\n Else (feature 0 > 23.96)\n If (feature 0 <= 24.01)\n Predict: 20.15\n Else (feature 0 > 24.01)\n Predict: 19.589999999999999\n Tree 2 (weight 1.0):\n If (feature 0 <= 23.66)\n If (feature 0 <= 22.3)\n If (feature 0 <= 22.13)\n Predict: 19.

```

Figure – 5.8: Random Forest for Prediction Outcome

5.4.4 GRADIENT BOOSTING TREE

The tree structure is produced from the prediction analysis using Gradient Boosting Tree algorithm. The following graphical representation shows the variations.

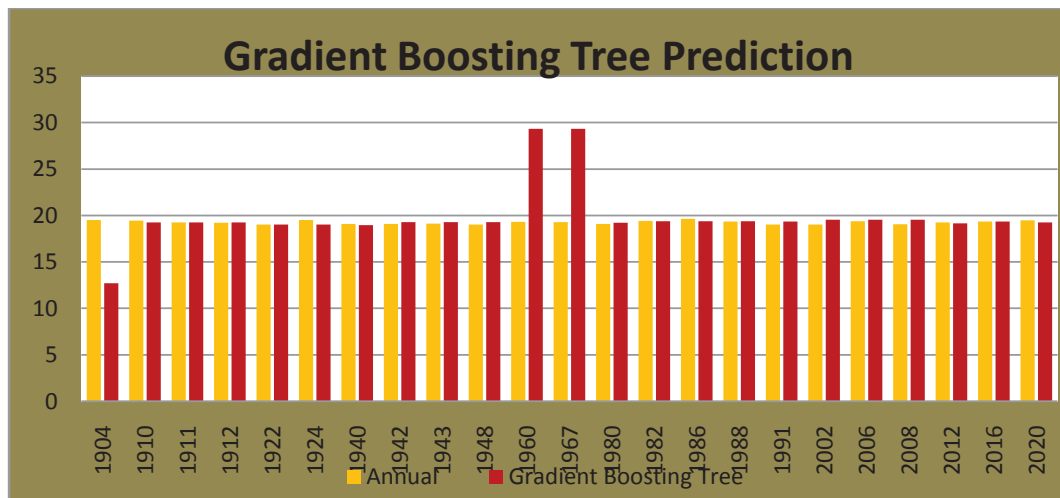


Figure – 5.9: Annual Temperature and Prediction Representation using Gradient Boosting Tree

Observations: Annual Temperature is compared against the prediction using Gradient Boosting Tree with an accuracy of 93.56% (Refer Figure 5.9). The tree diagram that represents the algorithmic output is shown in figure 5.10.

```

Out[5]: 'GBRegressor (uid=GBRegressor_4f36814e93c4fc654d3a) with 20 trees\n Tree 0 (weight 1.0):\n   If (feature 1 <= 12.99)\n     Predict: 0.003\n     Else (feature 1 > 12.99)\n       If (feature 0 <= 12.16)\n         Predict: 18.76\n         Else (feature 0 > 12.16)\n           Predict: 18.62\n           Else (feature 6 > 23.1)\n             If (feature 14 <= 23.47)\n               If (feature 13 <= 20.29)\n                 If (feature 14 <= 22.94)\n                   If (feature 10 <= 15.87)\n                     Predict: 18.799999999999997\n                     Else (feature 10 > 15.87)\n                       Predict: 18.9375\n                       Else (feature 14 > 22.94)\n                         If (feature 15 <= 16.39)\n                           Predict: 19.074166666666667\n                           Else (feature 15 > 16.39)\n                             Predict: 19.277142857142852\n                             Else (feature 13 > 20.29)\n                               If (feature 12 <= 14.32)\n                                 If (feature 15 <= 16.8)\n                                   Predict: 19.28723513513513\n                                   Else (feature 15 > 16.8)\n                                     Predict: 19.478333333333335\n                                     Else (feature 12 > 14.32)\n                                       If (feature 3 <= 21.74)\n                                         Predict: 19.543333333333333\n                                         Else (feature 3 > 21.74)\n                                           Predict: 19.71\n                                           Else (feature 14 > 23.47)\n                                             If (feature 15 <= 17.2)\n                                               If (feature 4 <= 22.98)\n                                                 If (feature 0 <= 13.88)\n                                                   Predict: 19.42\n                                                   Else (feature 0 > 13.88)\n                                                     Predict: 19.370000000000005\n                                                     Else (feature 4 > 22.98)\n                                                       If (feature 12 <= 13.84)\n                                                         Predict: 19.59\n                                                         Else (feature 12 > 13.84)\n                                                           Predict: 19.791249999999998\n                                                           Else (feature 15 > 17.2)\n                                                             If (feature 7 <= 23.82)\n                                                               If (feature 0 <= 13.51)\n                                                                 Predict: 19.96\n                                                                 Else (feature 0 > 13.51)\n                                                                   Predict: 19.92\n                                                                   Else (feature 7 > 23.82)\n                                                                     If (feature 7 <= 23.62)\n                                                                       If (feature 0 <= 13.62)\n                                                                         Predict: 20.15\n                                                                         Else (feature 0 > 13.62)\n                                                                           Predict: 20.39\n                                                                           Tree 1 (weight 0.1):\n                                                                           If (feature 12 <= 13.6813)\n                                                                           If (feature 7 <= 23.3)\n                                                                           If (feature 7 <= 23.21)\n                                                                           If (feature 0 <= 12.9)\n                                                                           If (feature 0 <= 12.39)\n                                                                           Predict: 0.015789961389964042\n                                                                           Else (feature 0 > 12.39)\n                                                                           Predict: -0.07464684684684168\n                                                                           Else (feature 0 > 12.9)\n                                                                           Predict: -0.3144702702702631\n                                                                           Else (feature 7 > 23.21)\n                                                                           If (feature 7 <= 23.25)\n                                                                           If (feature 0 <= 12.27)\n                                                                           Predict: -0.17447027027026252\n                                                                           Else (feature 0 > 12.27)\n                                                                           Predict: -0.2483333333333485\n                                                                           Else (feature 7 > 23.25)\n                                                                           If (feature 13 <= 20.07)\n                                                                           Predict: -0.25428571428570734\n                                                                           Else (feature 13 > 20.07)\n                                                                           Predict: -0.36348513513513225\n                                                                           Else (feature 7 > 23.3)\n                                                                           If (feature 2 <= 18.35)\n                                                                           If (feature 3 <= 21.49)\n                                                                           If (feature 6 <= 23.64)\n                                                                           Predict: 0.042830939510947984\n                                                                           Else (feature 6 > 23.64)\n                                                                           Predict: -0.10182489060488606\n                                                                           Else (feature 3 > 21.49)\n                                                                           If (feature 0 <= 12.27)\n                                                                           Predict: 0.18552972972973691\n                                                                           Else (feature 0 > 12.27)\n                                                                           Predict: 0.10552972972973862\n                                                                           Else (feature 2 > 18.35)\n                                                                           Predict: 0.30552972972974146\n                                                                           Else (feature 12 > 13.6813)\n                                                                           If (feature 11 <= 12.29)\n                                                                           Predict: -0.2544702702702608\n                                                                           Else (feature 11 > 12.29)\n                                                                           If (feature 12 <= 14.16)\n                                                                           If (feature 13 <= 21.14)\n                                                                           If (feature 12 <= 13.94)\n                                                                           Predict: 0.04245831794527903\n                                                                           Else (feature 12 > 13.94)\n                                                                           Predict: 0.171114

```

Figure – 5.10: Prediction Tree of Gradient Boosting Tree

5.5 COMPARATIVE ANALYSIS

The various machine learning algorithms such as Linear Regression, Random Forest, Decision Tree and Gradient Boosting tree algorithms are used in the prescribed model for prediction. Table 5.2 produces the comparative results of the various learning algorithms.

Table – 5.2: Prediction Analysis from Various Machine Learning Methods

Year	Linear Regression	Random Forest	Gradient Booster	Decision Tree
1905	23.8106	19.045	12.697	15.825
1910	24.2519	19.045	19.2367	15.825
1915	24.4107	19.045	19.2367	15.825
1920	24.2019	18.9079	19.2367	24.08
1925	25.451	18.9079	19.0267	24.08
1930	23.4775	19.04	19.0267	24.08
1935	24.2686	19.04	18.967	24.169
1940	24.6349	19.103	19.28	24.169
1945	23.8023	18.62	19.28	23.963
1950	24.5434	19.317	19.28	23.963
1955	24.1936	19.317	29.33	24.0795

1960	23.8855	19.317	29.33	24.0795
1965	24.7349	19.317	19.225	24.2567
1970	23.719	19.988	19.3716	24.2567
1975	23.744	19.456	19.3716	24.276
1980	24.4518	19.456	19.3716	24.267
1985	24.2436	19.456	19.337	23.8818
1990	25.2511	19.2493	19.554	23.8818
1995	24.6349	19.08	19.554	24.1435
2000	24.3019	19.08	19.554	24.1435
2005	23.5327	19.457	19.475	24.517
2010	24.7825	19.251	19.147	23.486
2015	25.2378	19.249	19.357	24.143
2020	24.5673	19.364	19.264	24.0245

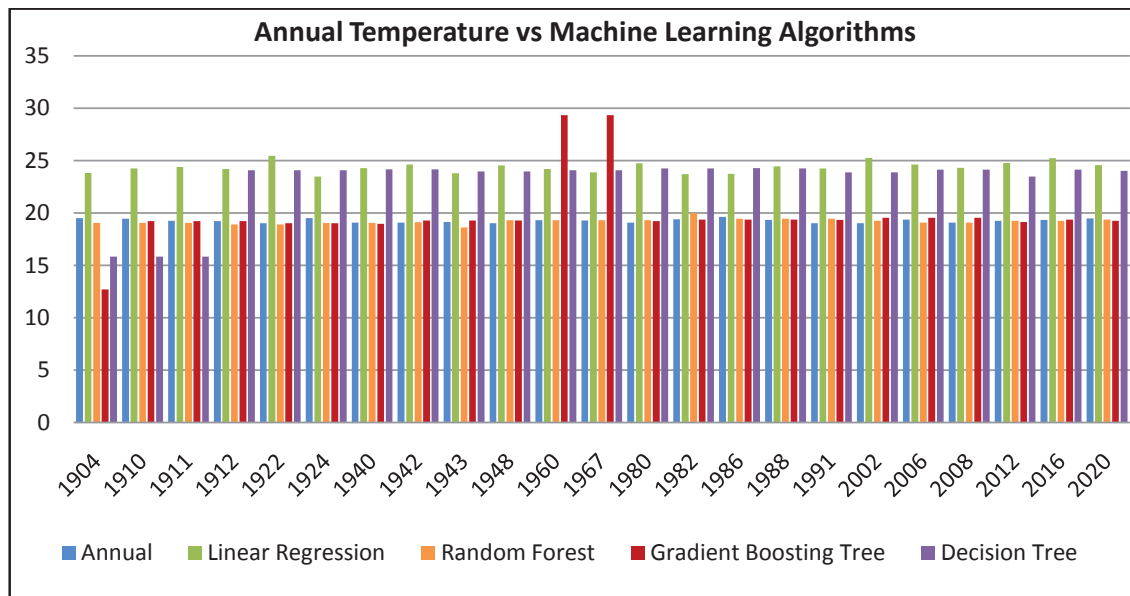


Figure – 5.11: Predictive Values based on Various Machine Learning Techniques

Observation: Figure 5.11 shows the comparative results of annual temperature with prediction values of Machine Learning algorithms. The observed value discloses the accuracy of Random forest for annual temperature. It embarks high accuracy when compared with other learning methods.

5.6 EFFICIENCY

Each Machine Learning method is applied on top of the developed model with its own significance and specialization. The efficiency of the learning methods is described in the table 5.3.

Table – 5.3: Efficiency of Machine Learning Methods

Machine Learning Methods	Efficiency
Linear Regression	Low Space Utilization
Gradient Boosting Tree	Low Time Utilization
Random Forest	Accurate Prediction
Decision Tree	Less Error Rate (RMSE)

Table 5.4 illustrates the analysed data that Predict values over annual temperature.

Table - 5.4: Prediction Analysis over Annual Temperature

Year	Annual	Linear Regression	Random Forest	Gradient Booster	Decision Tree
1904	19.51	23.8106	19.045	12.697	15.825
1910	19.44	24.2519	19.045	19.2367	15.825
1911	19.25	24.4107	19.045	19.2367	15.825
1912	19.22	24.2019	18.9079	19.2367	24.08
1922	19.03	25.451	18.9079	19.0267	24.08
1924	19.51	23.4775	19.04	19.0267	24.08
1940	19.08	24.2686	19.04	18.967	24.169
1942	19.09	24.6349	19.103	19.28	24.169
1943	19.13	23.8023	18.62	19.28	23.963
1948	19.01	24.5434	19.317	19.28	23.963
1960	19.31	24.1936	19.317	29.33	24.0795
1967	19.27	23.8855	19.317	29.33	24.0795
1980	19.09	24.7349	19.317	19.225	24.2567
1982	19.41	23.719	19.988	19.3716	24.2567
1986	19.64	23.744	19.456	19.3716	24.276

1988	19.34	24.4518	19.456	19.3716	24.267
1991	19.02	24.2436	19.456	19.337	23.8818
2002	19.02	25.2511	19.2493	19.554	23.8818
2006	19.37	24.6349	19.08	19.554	24.1435
2008	19.07	24.3019	19.08	19.554	24.1435
2012	19.24	24.7825	19.251	19.147	23.486
2016	19.34	25.2378	19.249	19.357	24.143
2020	19.47	24.5673	19.364	19.264	24.0245

Random Mean Square Error (RMSE) is calculated for the various algorithms (Refer table 5.5). RMSE for Decision Tree = 0.223611 is minimum when compared to remaining models. Linear Regression shows the high error rate of 0.453279.

Table – 5.5: RMSE over ML Techniques

Algorithm	Random Mean Square Error
Decision Tree	0.223611
Random Forest	0.247448
Gradient Boosting Tree	0.264829
Linear Regression	0.453279

The Machine Learning techniques reveal the efficient use of time and space. These methods train the machine with the dataset for further processing. Table 5.6 and Figure 5.12 provides the gist of techniques used to analyze the complexities. This shows the collective measures of supervised learning techniques.

Table – 5.6: Time and Space Efficiency Measures

Machine Learning Techniques	Time	Space
linear Regression	12.564	3.452
Gradient Boosted tree	6.744	8.675
Random Forest	6.754	9.632
Decision Tree	7.675	8.912

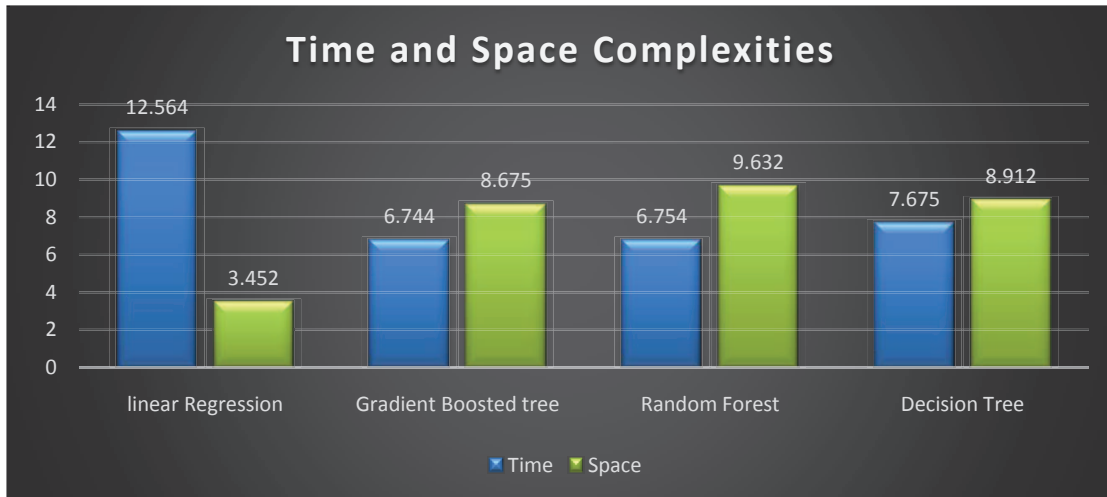


Figure – 5.12: Time and Space Complexity for Various Techniques

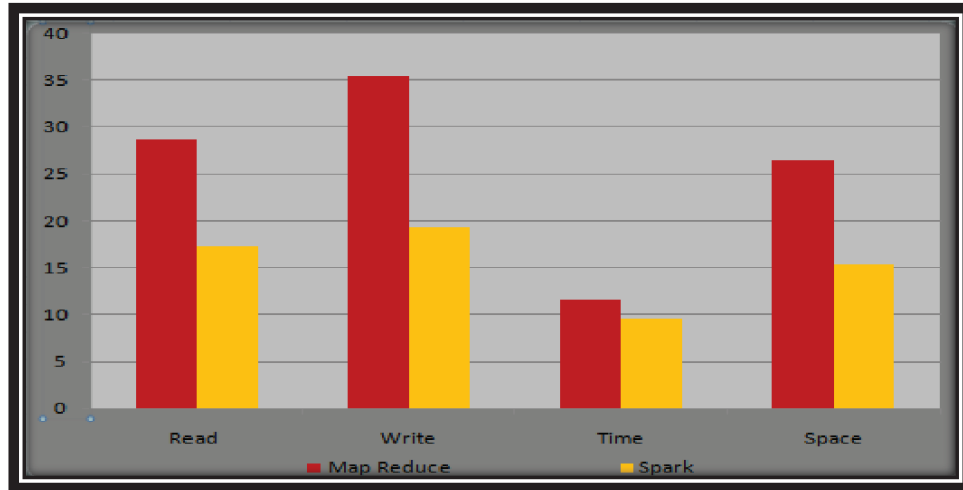
Time and Space complexity is identified for various Machine Learning techniques. From the resultant output it is evident that Machine Learning techniques provides more accurate measures. Comparisons between these learning algorithms reveal that the transparency is required for storage and analysis.

For space complexity linear regression shows the best efficiency and with respect to time complexity Gradient Boosting Tree shows better performance. In terms of prediction accuracy Random Forest explicitly shows better results. The Decision Tree computations are low in error rate (RMSE). This discussion extends when the job distribution happens on cluster.

Table – 5.7: Comparative Results between MapReduce and Spark based on Various Features

Features	MapReduce	Spark
Read	28.65	17.23
Write	35.43	19.41
Time	11.57	9.47
Space	26.54	15.43

The efficiency is observed by comparing MapReduce with Spark framework. The performance is evaluated on time and space complexity for better interpretations of results.



**Figure – 5.13: Read, Write, Time and Space Complexity
Comparison between Spark and MapReduce**

The MapReduce model exhibits high disk rates for read operations with a result of 28.65 MB/Sec and for write operations with 35.43 MB/Sec. It shows reduced performance on a processor. While executing the jobs on Spark platform, the results are 17.23 MB/Sec on read and 19.41 MB/Sec for write operations. Table 5.7 and Figure 5.13 shows the combined measures of learning techniques considering the time, space, read and write operations. Hence Spark shows 65.21% of improvement on MapReduce framework.

5.7 SUMMARY

This chapter elaborately explained the results obtained using learning algorithm. It described the outcomes by analyzing the efficiency through various regression models. The work based on comparative analysis of learning algorithms is relatively remarkable. Furthermore, the supervised way of identifying the regression models is a coherent attempt made in interpreting the prediction on datasets using adaptable tools. Thus this chapter portrayed the predictive models preferred for data analytics which are examined and analyzed in respective domains.