SVKM's NMIMS

Mukesh Patel School of Technology Management and Engineering

A REPORT ON:

AUGMENTED DATA ANALYTICS

&

COMMUNITY FOREST ALGORITHM

By:

VARUN PRAGNESH SHAH

Course: MBA Tech Computers | Semester: VII

Roll No: N073 | SAP ID: 70471015051

Project Time Frame: 25th April 2019 – 30th June 2019

Organisation:

Reliance Jio Infocomm Limited

Ghansoli, Navi Mumbai – 400 701

A REPORT ON:

AUGMENTED DATA ANALYTICS

&

COMMUNITY FOREST ALGORITHM

By

VARUN PRAGNESH SHAH

(EP NO: EP5700095871)

A Report submitted in partial fulfilment of the requirements of 5 years Integrated MBA (Tech) Program of Mukesh Patel School of Technology Management & Engineering, NMIMS

Distribution List:

Ms. Binny Khanna (Faculty Mentor, M.P.S.T.M.E)
&
Mr. George Cherian (Industry Mentor, Reliance Jio)

Acknowledgements

Mr. Raghuram Velega (Head of Big Data CoE)

Mr. Raghuram Velega leads the efforts of the Big Data Center of Excellence. He has provided many insights into the world of ML and AI and its future applications. Mr. Raghuram has been very approachable and has helped me understand how AI has been growing in India and how Jio is leveraging it to transform their business.

Mr. George Cherian (Industry Mentor, Head of Data Science Team)

Mr. George Cherian has been my project mentor throughout my time at Reliance Jio. It has been under him that I have worked on the DAM and Community Forest Algorithm. Mr. George has been an amazing mentor and under him I have learnt a lot. He has taught me key lessons that are important in the corporate world. On many occasions Mr. George has shared his experiences at various companies like Goldman Sachs, HP and JP Morgan Chase which provided a deeper insight into how multinational companies assess and solve problems.

Ms. Binny Khanna (Faculty Mentor, M.P.S.T.M.E)

Ms. Binny has guide me throughout the internship and has always attended my calls whenever I had questions. Through her co-operation this internship has been one of the most rewarding learning experiences of my life.

Dr. Shailesh Kumar (Chief Data Scientist at Jio)

Dr. Shailesh Kumar is the chief data scientist at Reliance Jio and it is his project that I have been working on. The Community Forest algorithm was originally theorised by him. He has been extremely helpful in helping me implement the algorithm and has taken the time out of his busy schedule to help me understand the algorithm and the vision behind it.

Mr. Anurag Sahoo (AI CoE)

Mr. Anurag Sahoo is part of the Artificial Intelligence Centre of Excellence at Hyderabad. He works closely with Dr. Shailesh and it is with him that I have implemented the Community Forest Algorithm. He has been extremely instrumental in explaining how code must be structured. We have also spent a lot of time discussing how to optimise the Community Forest code.

Mr. Vishal Naidu (Analytics CoE)

Mr. Vishal Naidu is part of the Analytics Center of Excellence. He is part of the JDSP team and has been extremely helpful in making me understand the inner working of Jio. Vishal has helped me settle in and was always the go to guy for any doubts with JDSP.

Mr. Shivam Runthala (Analytics CoE)

Mr. Shivam Runthala is part of the Analytics Center of Excellence. He is also part of the JDSP team and has been extremely helpful in making me understand the inner working of Jio. Shivam has helped me settle in and was always the go to guy for any doubts with JDSP.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ABSTRACT

Augmented Data Analytics

Augmented Analytics automates data insight by utilizing Machine Learning to automate data preparation and enable data sharing. This advanced use, manipulation and presentation of data simplifies data to present clear results and provides access to sophisticated tools so business users can make day-to-day decisions with confidence. Users can go beyond opinion and bias to get real insight and act on data quickly and accurately.

Augmented Analytics also empowers business users with access to meaningful data to test theories and hypotheses without the assistance of data scientists or IT staff. It allows users access to crucial data and Information and allows them to connect to various data sources (personal, external, Cloud, and IT provisioned). Users can mash-up and integrate data in a single, uniform, interactive view and leverage auto-suggested relationships, JOINs, type casts, hierarchies and clean, reduce and clarify data so that it is easier to use and interpret, using integrated statistical algorithms like binning, clustering and regression for noise reduction and identification of trends and patterns. The ideal solution should balance agility with Data Governance to provide Data Quality and clear watermarks to identify the source of data.

"With Augmented Data Analytics we can truly Democratize AI"

Community Forest Algorithm

To enhance the process of Augmented Analytics, algorithm selection plays an important role in the quality of insights generated by the Augmented Analytics System. Thus, to complement the Analytics Augmented System a new algorithm temporarily named Community Forest was theorised by Dr. Shailesh Kumar as part of his PhD thesis. This approach is a patented approach held between Dr. Shailesh and Google.

The Community Forest algorithm is an advanced community detection algorithm that detects communities from graphs. This algorithm is mainly used in order to gain such insights from heterogenous data that wouldn't be otherwise obvious at the human level. The Community Forest algorithm in a way grants complete autocracy to the system to generate insights based on data correlation and other factors.

The Community Forest algorithm is an Artificial Intelligence approach to problem solving that uses machine learning techniques in its implementation. The projects that I have worked on in my time at Reliance Jio combine augmented data analytics with the community forest algorithm and can been used to generate valuable insights for Jio.

Jio Data Science Platform

JDSP is responsible for designing and implementing processes and layouts for complex, large-scale data sets used for modelling, data mining, and research purposes.

It focuses on business case development, planning, coordination / collaboration with multiple teams & project, managing the life-cycle of an analysis project, and interface with business sponsors to provide periodic updates.

Data scientists perform data analysis, develop algorithms and predictive models, and deliver reports for a broad audience. They apply algorithms and modelling tools to a variety of data assets and develop critical solutions for multiple areas of the business.

Jio Big Data Lake

The Jio Big Data Lake (JBDL) is been created by Reliance Jio in order to serve as a data lake for all Reliance's subsidiaries. A data lake is a storage repository that holds a vast amount of raw data in its native format until it is needed. Reliance uses a HDFS to store their data in the JBDL. A Hadoop Distributed File System (HDFS) is the primary data storage system used by Hadoop applications.

It is through the JBDL that I have been able to test my algorithms and code on real world production data. My project mainly revolved around telecom data that was generated by Jio's mobile towers. The Jio Big Data Lake is the largest of its kind in India and processes 2.5 petabytes of data every day.

PROJECT 1: MARKOV CHAINS

INTRODUCTION

Note:

This project was my pilot project and was completed within a week. The main reason of this project was to understand how to write data agnostic models.

Markov chains are a fairly common, and relatively simple, way to statistically model random processes. They have been used in many different domains, ranging from text generation to financial modelling.

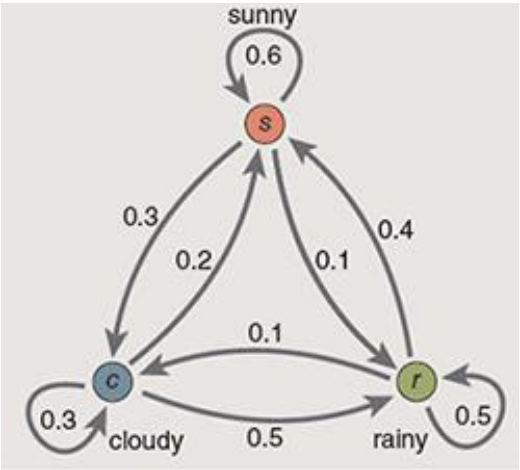
A Markov chain is a mathematical system usually defined as a collection of random variables, that transition from one state to another according to certain probabilistic rules. These set of transition satisfies the Markov Property, which states that the probability of transitioning to any particular state is dependent solely on the current state and time elapsed, and not on the sequence of state that preceded it. This unique characteristic of Markov processes renders them memoryless.

The Markov Chain can be understood easily by using the weather example.

Let us assume 3 states of the weather: cloudy, rainy and sunny. Associated with each of these states is a probability score which takes into account the present weather. This can be represented in a N x N matrix format where N represents the number of states. For this example, let us assume the probability matrix P to be

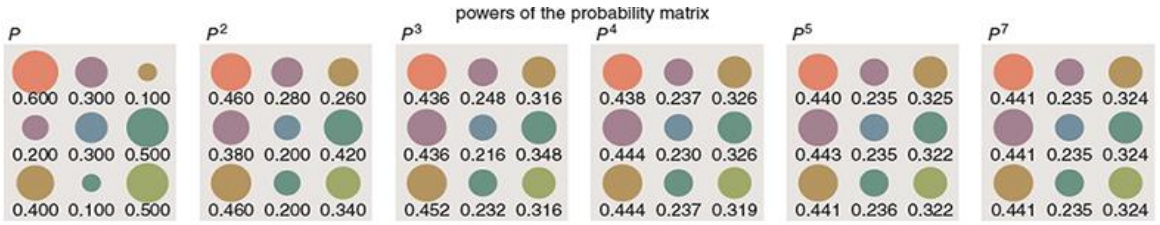
P =

	S	C	R
S	0.6	0.3	0.1
C	0.2	0.3	0.5
R	0.4	0.1	0.5



The chain can also answer questions such as, “If it’s cloudy today, what is the probability of rain two days from now?” The answer is found by summing the contributions of all pathways that lead from the *cloudy* state to the *rainy* state in exactly two steps. This sounds like a tedious exercise, but there’s an easy way to organize the computation, based on the arithmetic of matrices.

The transition probabilities for a three-state Markov chain can be arranged in a three-by-three matrix—a square array of nine numbers. As shown in the illustration on the left, the process for calculating multistage transitions is equivalent to matrix multiplication. The matrix itself (call it P) predicts tomorrow’s weather; the product $P \times P$, or P^2 , gives weather probabilities for the day after tomorrow; P^3 defines the probabilities for three days hence, and so on. The entire future unfolds from this one matrix.



IMPLEMENTATION

The main goal of this problem is to implement a Markovian Chain Model that can be used in turn to predict anomalies at mobile towers. The algorithm / code written must be completely Data Agnostic in nature and thus offers flexibility to the users to run it on any data set possible. The code of the algorithm is visible below.

Code:

```
#importing required libraries
import numpy as np
import pandas as pd
import statistics

#User inputs file name. Note: file should be in same directory as jupyter notbook
file_name=input("Enter file name")
#Converts CSV file to DataFrame
df=pd.read_csv(file_name)
df.head()
#Enter row name on which Markov Model will be based.
#Note: if incorrect row name is provided program will again ask for input
while(True):
    attribute_1=input("Enter ROW NAME to base markov model on: ")
    if(attribute_1 in df):
        break
    else:
        print("Please enter correct row name")

#To deal with missing values imputation is done by computing mode
#x holds the mode of the df row
x=statistics.mode(df[attribute_1])
#replacing NaN values with mode (x)
df[attribute_1].fillna(x, inplace = True)

#creating probability matrix for markov model
prob_matrix=pd.DataFrame(index=df[attribute_1].unique(),columns=df[attribute_1].uni
que())
#no_rows holds numeric value containing number of rows in that column
no_rows=df[attribute_1].count()
#replacing NaN values with 0 in probability matrix
prob_matrix.fillna(0, inplace = True)

#iterating over each entry and incrementing in probability matrix
for i in range(0,no_rows-1):
    prob_matrix[df[attribute_1][i+1]][df[attribute_1][i]]+=1
#printing probability matrix
prob_matrix
#Normalising the probability matrix i.e converting to probability
for i in prob_matrix:
    prob_matrix[i]=prob_matrix[i]/prob_matrix[i].sum(axis=0)
prob_matrix
print(df[attribute_1].unique())
sequence=input("Enter Sequence of what probability you want to calculate (space sep
erated)")
sequence=sequence.split(" ")
prob=1
for i in range(0,len(sequence)-1):
    prob=prob * prob_matrix[sequence[i+1]][sequence[i]]
prob
```

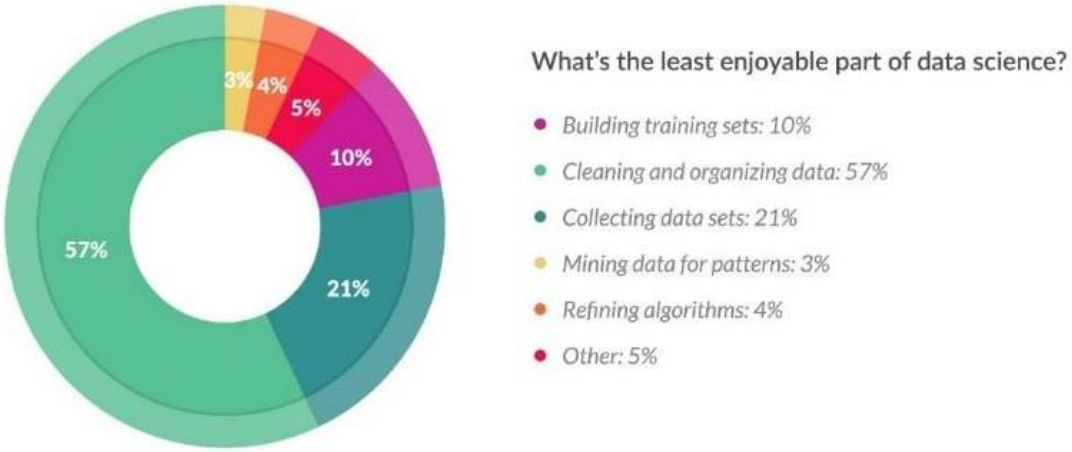
CONCLUSION

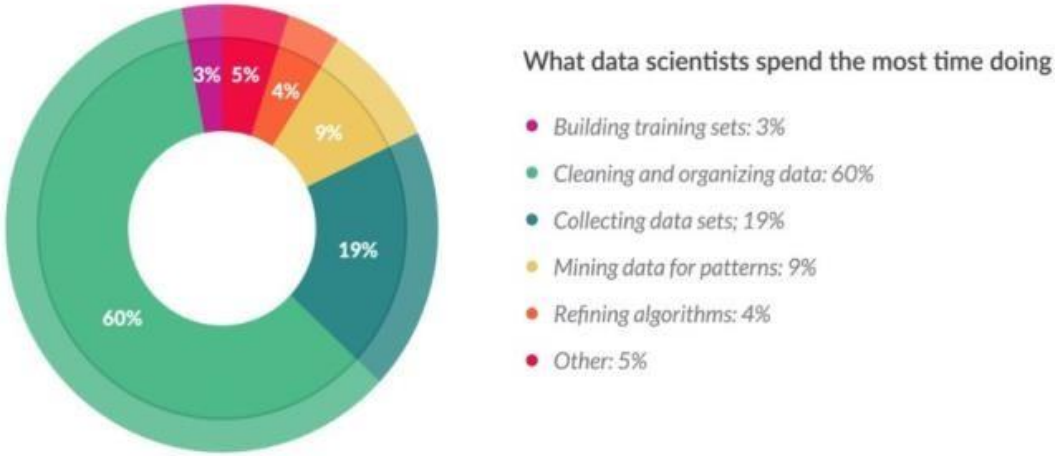
Markov chains have a versatile range of applications. For Reliance Jio it can be used to predict anomalies in telecom data. Implementing this project gave me a better idea on the various factors one has to consider while implementing a Data Agnostic Model. One the the most important lessons was the necessity to bin, scale and encode the data we are working on. These techniques can significantly optimize our code and can reduce computational needs massively.

PROJECT 2: DATA AGNOSIC HYGIENE MODULE

INTRODUCTION

The Data Agnostic Hygiene Module is a library used to clean raw datasets and converts them into clean csv files. As the statistics below suggests, the least enjoyable part of data science correlates with what data scientists spend most of their time on i.e. Cleaning and organizing data. My current project aims to automate the process of cleaning and organizing data along with the process of mining data for patterns.





My current project requires me to work with two teams, The Data Science CoE (Centre of Excellence) located in Mumbai and the AI CoE (Centre of Excellence) located in Hyderabad. The end goal of this project is to be able to add these Augmented Data Analytics Features to the JDSP. These features will be added as pipelines that can be configured as per the users' requirements and can be called by a DAG (Directed Acyclic Graph) in the mentioned order. The end goal of such a system is to reduce the need for data scientists and enable business teams to visualize and find patterns from data in an automated and unsupervised manner.

The end Goals of this project are:

A. Write a data agnostic algorithm to detect the types of data in a data set and classify them into the following:
    - ✓ Latitude and Longitude
    - ✓ Date and Time
    - ✓ ID Fields
    - ✓ Numerical categories
    - ✓ Logistic Classifications (Boolean Classifications)
B. Write functions to take advantage of these classifications that the user can call to clean, wrangle or analyse the data
C. Write a data agnostic algorithm to automate the process of calculating the feature importance of attributes
D. Write a data agnostic algorithm to help analyse and visualize the correlation between features/attributes
E. Write an algorithm to provide Perspectives to users
F. Integrate these algorithms as features to the Jio Data Science Platform

Modules:

The project is broken up into 3 modules:

1. Data Cleaning:
    - Performs the automated cleaning of data by dropping user specified columns
    - Based on the specified target feature the system drops rows with NaN or Null Values

2. Data Classification:
    - Classifies the type of data present in columns in terms of high-level classification.
    - The classification parameters are:
        - Latitude and Longitude
        - Date and Time
        - ID Fields
        - Numerical categories
        - Logistic Classifications (Boolean Classifications)

3. Class Functionality:
    - Based on the classification in the previous module the user has access to functions that manipulate, visualize, extract or convert the data
    - These functions are classification specific for eg. If system detects Latitude and Longitude Data the user has access to a function that plots the co-ordinates in the form of a heat map, density map or marker map

IMPLEMENTATION

1. **Data Cleaning:**
   - The Data Cleaning model is a Data Agnostic model which takes a file path as an input.
   - The file path is the file location at which the data is stored. The file path could point to a local data file or could point to a cloud instance or an HDFS.
   - The user may also provide a list of columns that need to be dropped. The system takes this input and drops the respective columns and returns an updated dataframe.
   - The user may also specify a list of target columns. With this input the system intelligently drops any row where the target columns value is Null or NaN (Not a Number)
   - The Data cleaning module also leverages the Data Classification module to drop those columns that have been classified as ID's. Since ID columns are not considered while modelling or analysing data they are redundant and must be dropped.
   - The data cleaning module also provides imputation functions to fill Null and NaN values. The basis of what imputation functions to be used is decided by the data classification module on the basis of what type of data is to be imputed.
   - The output of this module is a clean dataframe that is ready to be analysed by the Data Exploration Module

   The functions part of this module are:

   cleaner(csv_filepath,drop_column_list,target_list,null_value_threshold, numerical_category_threshold)

   reader(csv_filepath,drop_column_list,target_list,null_value_threshold)

   imputer(df, col_dict_final)

   convert_to_csv(df, csv_file_path)

   encoder(df, col_dict_final, type_of_encoding)

2. **Data Classification:**
    - The Data Classification module is also designed in a data agnostic method. This module takes the semi-cleaned data from the Data Cleaning Module as an input.
    - Here the role of the Data Classification module is to classify the columns by analysing the data of the columns according to some predefined classes.
    - This intelligent system uses column names along with recognizing the type of data to detect the final classification of the column, Thus, providing robustness that accounts for wrongly labelled or unlabelled data to give a correct output.
    - Since this system is data agnostic and is designed to work with big data which may contain millions of rows and columns the classification algorithm is designed in such a way that the classification id done within a few seconds.
    - The classifier detects the following classes:
        - **Latitude and Longitude:** The algorithm scans through all the columns of the dataset to check whether the data is in the form of latitude or longitude coordinates. The algorithm does so by leveraging regular expressions. In order to increase processing time, the algorithm works only on a randomly generated sample of the original data set. If the algorithm detects the data to be of longitude and latitude type, the column name is saved in a dictionary and the next column is then analysed. The user can also specify the expected region for eg. Asia, India or Mumbai in order to filter the dataset according to the coordinates that fall within the city country or continent.
        - **Date and Time:** The algorithm scans through the entire dataset to check whether a column containing a date or time object is present. The algorithm is designed to detect the time or date in any format (dd/mm/yyyy, mm/dd/yyyy, dd MM, YYYY etc.). This classification can help provide timeseries information in order to analyse data in a better manner.
        - **ID Fields:** ID fields represent redundant data in a data set. By classifying data as an ID field and sending it to the cleaning module the system can automatically drop the ID columns from the dataset.
        - **Numerical categories:** Numerical categories are data that even though may be numerical in nature represent a category. For example, building 22 even though a numerical value represents a category. This is an extremely important classifier for data scientists as these numerical category data must not be treated as normal integers.
        - **Logistic Classifications (Boolean Classifications):** Logistic classification classifies those data that are binary in nature and indicate either a presence or absence of an attribute. This also includes Boolean values like True or False along with bit values like 1 and 0. Columns classified under Logistic Classifications are generally the target variables on which models need to be based on.

The modules part of this algorithm are:

classifier(df, numerical_category_threshold)

evaluator(df, evaluator_threshold=0.7)

_is_datetime(df, col_dict_final)

_is_numerical_category(df, col_dict_final, numerical_category_threshold=100)

_is_longitude_latitude(df_sample,col_dict_final)

_is_categorical(df,col_dict_final,threshold=100)

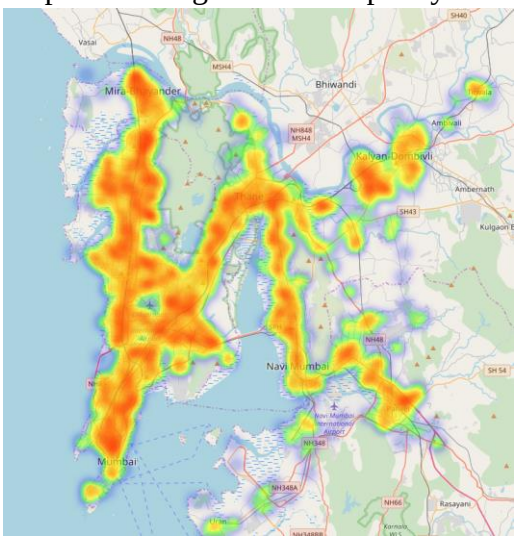
_is_binary_classification(df,col_dict_final)

_is_id(df, col_dict_final, threshold=0.99)

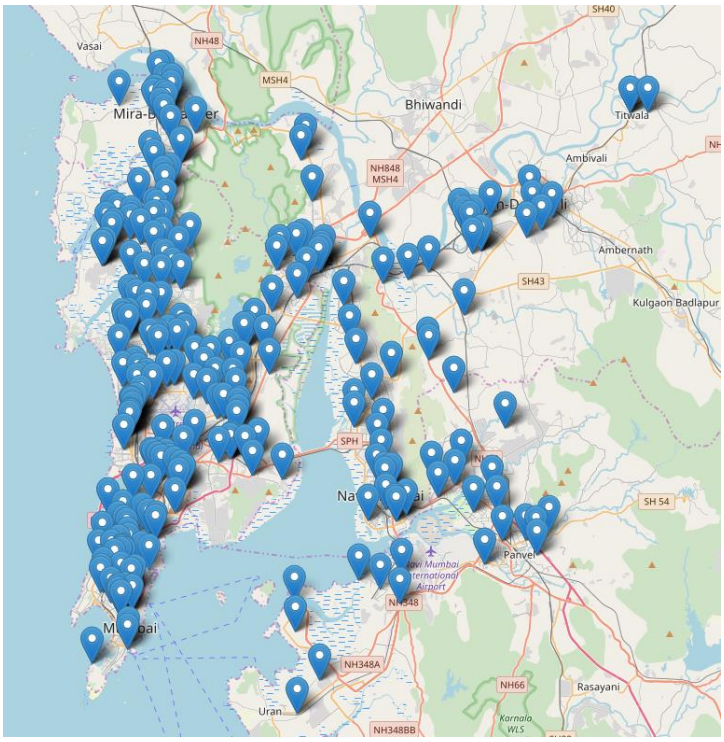
3. **Class Functionality:**
   - The class functionality module provides functions that can be used on columns that have been classified by the Data Classifier module.
   - The Class Functionality module provides certain class specific features for the user that gives them an insight to the data or helps manipulate the data easily.

   The various class specific functions are as follows:

   - **Latitude and Longitude:**
     i. **LocationFilter**: This function takes a Continent, Country or City as an input in plain text format and based on the spanning coordinates of those locations returns a dataset with only those coordinates.
     ii. **DensityMaps**: This function takes an input of latitude and longitude columns and plots a density map using google maps (license pending) or openstreetmaps with a legend. The map may look like this:



iii. **MarkerMaps**: This function takes an input of latitude and longitude co-ordinates and plots a marker map. This may be useful for anomaly detection visualization. The marker map may look like this:



- **Date and Time:**
  i. **DayofWeek**: The function extracts the date from the timestamp and converts it to the day of week format for example 22-05-19 can be extracted and converted to Wednesday.

  ii. **MonthofYear**: The function extracts the date from the timestamp and converts it to the Month of Year format. This is useful while analysing seasonal data.

- **ID Fields:**
  i. **DropID**: This function uses the Data Classifier module to detect ID columns and drop them. This helps reduce the size of the dataset by losing redundant data.

CONCLUSIONS AND RECOMMENDATIONS

The data agnostic hygiene module is an extremely powerful tool for data scientist. By providing all the pre-processing tools in a single library an performing functions in an automated manner we have saved a lot of time while cleaning our datasets. The unique feature of classifying each column can be leveraged in the future to implement more functionalities. The modularity of the code also means that each function can be called individually thus giving the code more flexibility.

By implementing the current python algorithm in pyspark this algorithm can be extremely crucial in big data applications.

New classification parameters can also be added to make the code more flexible and versatile across different industries.

In the future we could combine heuristics along with the data distribution to accurately categorise data. A new binning module can also be added to offer wider functionality. The code even though takes 0.12 milliseconds to run can also be further optimised in order to support cleaning big data.

PROJECT 3: COMMUNITY FOREST

INTRODUCTION

Community Forest Algorithm is an Ensemble Learning Technique in which logical communitites of related features are constructed on the basis of mutual affinity of features in a dataset. It employs a Gating Mechanism in the second stage where records are clustered into community based on their likelihood of being closely associated to a specific community. This algorithm outperforms the prexisting Supervised Machine Learning Algorithms and gives detailed insight of the data.

The main goal of this algorithm is to give insights to users as to what group of features lead to a certain metric or prediction. The main application of this algorithm lies in cases where the data is non-homogenous i.e. the data is a combination of multiple sources of data, and where the number of features in the data is large.

The community forest algorithm has been divided into 4 phases:

1) Feature Affinity Graph generation
2) Community Detection of features
3) Modelling each Community
4) Gating Mechanism

IMPLEMENTATION

The entire architecture of the Community Forest Algorithm has been divided into 4 phases namely:

Phase 1: Feature Affinity Graph generation:

- This phase generates an affinity graph from the cleaned data it recieves as an input.
- The affinity graph is calculated by taking all permutations and combinations of the entire cleaned feature list.
- The affinity can be calculated by using any method. This method can be given in the form of a hyperparameter.
- The Affinity Graph generated by this phase contains the features as nodes and the affinity as weights. Thus the output is finally a weighted graph.

Phase 2: Community Detection of features:

- This algorithm recieves the Affinity Graph from the previous phase as an input.
- This algorithm then performs various operations on the Affinity Graph and detects communities from it.
- The output of this phase is a whole list of unique communities

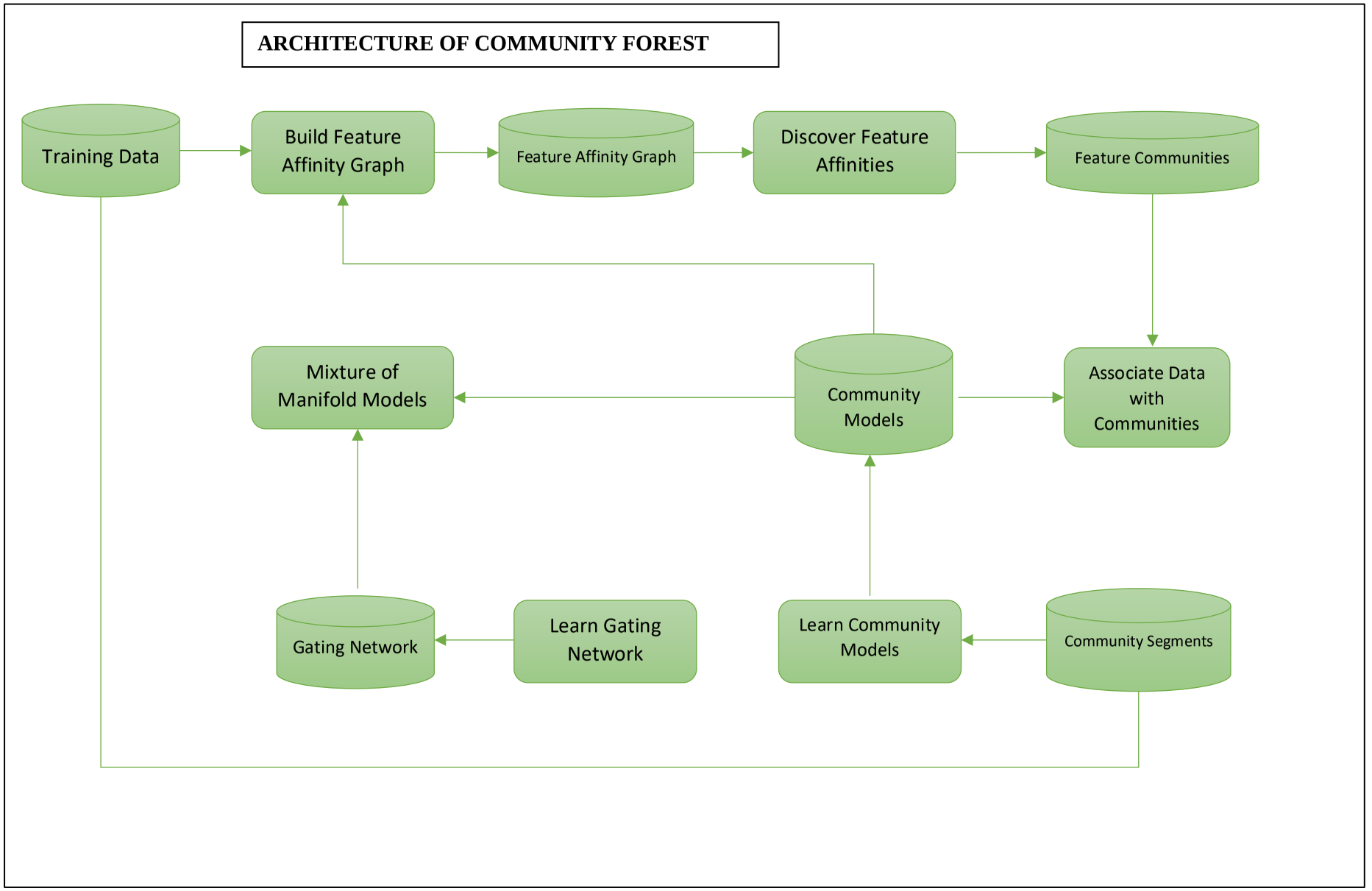
Phase 3: Modelling each Community:

- This phase is used to create a model for each and every community individually.
- The ML algorithm can be slected wither by taking it directly from the hyper paremeter or by iterating over simple ML algorithms and chosing the one with the best accuracy

Phase 4: Gating Mechanism:

- The gating mechanisms is used to allocate new data points to each community.
- This can be done by comparing confidence scores returned form all the communitity models generated in phase 3 and apllying a function such as ‘arg max’ to it.
- The gating mechanism is extremely crucial in the entire process as it reieratively trains the models.

The entire architecture of the the community forest algorithm can be seen on the next page in figure 1.


ARCHITECTURE OF COMMUNITY FOREST
Training Data
Build Feature Affinity Graph
Feature Affinity Graph
Discover Feature Affinities
Feature Communities
Mixture of Manifold Models
Community Models
Associate Data with Communities
Gating Network
Learn Gating Network
Learn Community Models
Community Segments

PHASE 1 (Feature Affinity Graph Generation):

The first phase requires a cleaned CSV file as an input. This cleaned file can either be generated by the DAM (refer project 2) or can be a manually cleaned file that the user can specify a path to.

A cleaned CSV file refers to one where:

- All unnecessary columns have been dropped
- Any row of the target column with a NaN or NULL value has been dropped
- All NaN and NULL values belonging to categorical, numerical category or Boolean classification data have been imputed. The imputing can be done using a simple mode operation or any other user defined operation.
- All NaN and NULL values belonging to numerical data or other continuous data must also be imputed. The imputation can be done by using a simple mean function or any other function defined by the user.
- All categorical data has been encoded. The encoding can be done either by using a OneHotEncoder from scikitlearns' pre-processing library (sklearn.preprocessing.OneHotEncoder()) or by using a LabelEncoder from the same library (sklearn.preprocessing.LabelEncoder())
- The final cleaned data frame must contain numerical values in all columns.

Once a cleaned csv file has been inputted in the algorithm, a machine learning model (M) must be chosen to train the data. The machine learning model is input as a hyperparameter by the user in the current implementation however in future implementations the machine learning model will be iterated over and the best model will be chosen. The machine learning models used must be simple models. The current list of ML models that are supported are:

a) Decision Trees

b) Random Forest

c) Logistic Regression

d) XGBoost

Once a model has been selected and trained against the target column a goodness score (G) is calculated. The method to calculate the goodness score is also hyper-parameterised. The methods to calculate the goodness score given a model are:

a) Accuracy

b) F1 Score

c) AUC (Area Under Curve)

d) Precision

First the goodness score of a single feature (**i**) given a model (**M**) is calculated and can be represented mathematically as **G(i | M)**.

Then The goodness score of a pair of two features (i,j) is calculated given the same Model (M). This can be mathematically represented as **G(i,j | M)**

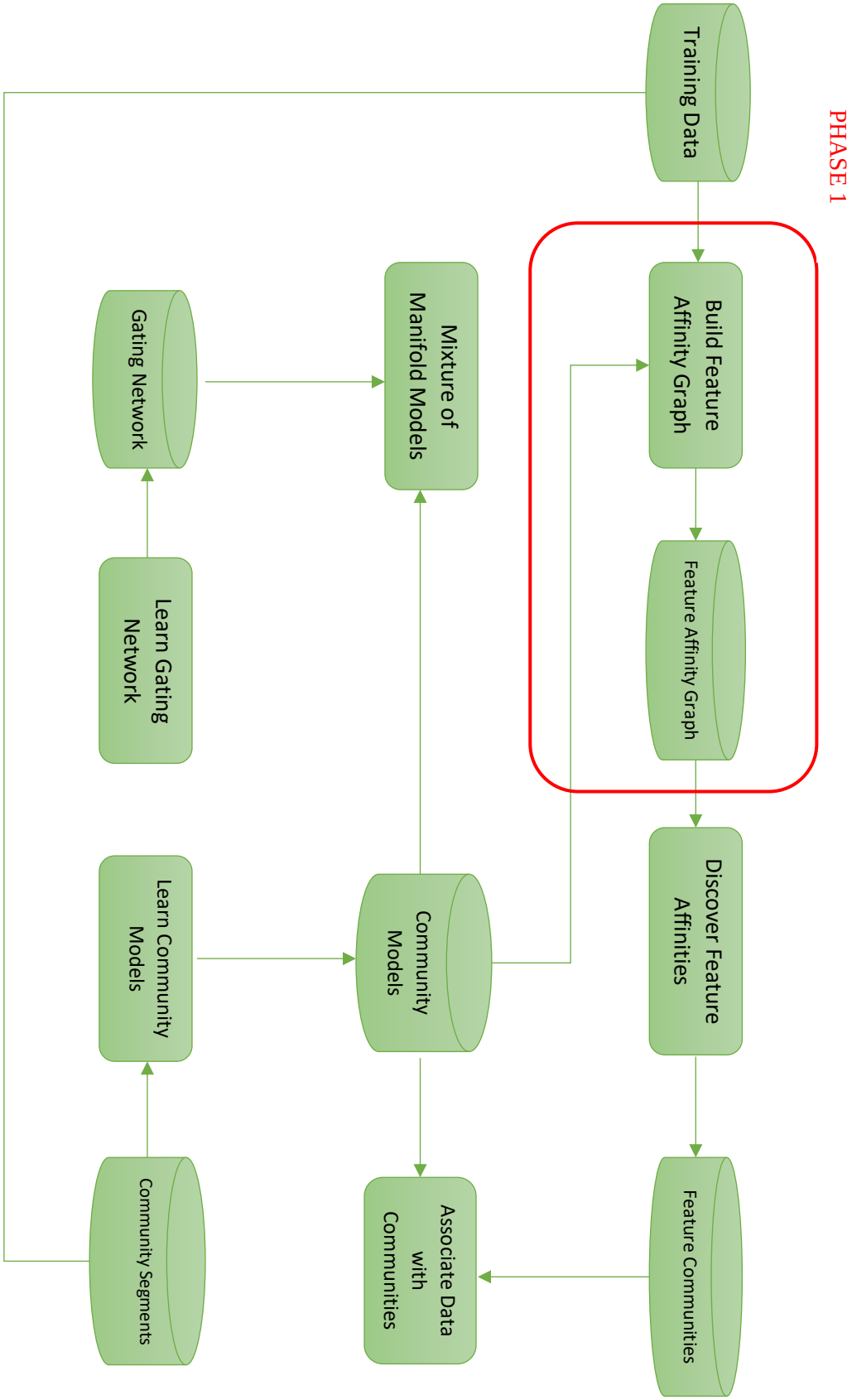
Now we calculate the strength between the two features, i.e. we check if taking the two features as a pair actually makes sense or is it better to leave the features alone. This strength of the two features is calculated using the following formula:

$$\mathbf{\Phi(i.j)} = \frac{\mathbf{G(i,j|M)}}{\sqrt{\mathbf{G(i|M)} * \mathbf{G(j|M)}}}$$

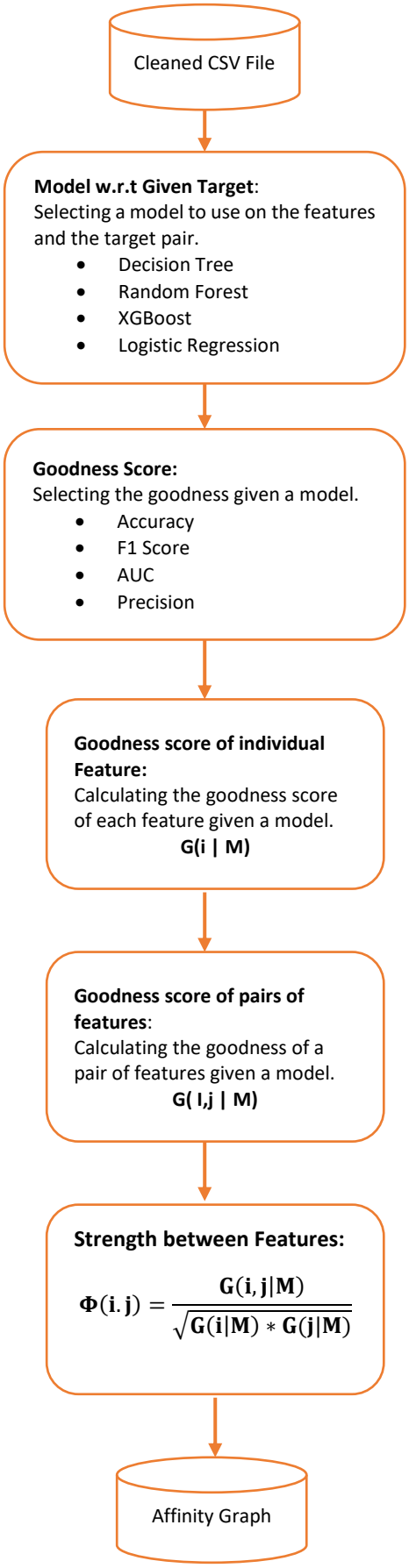
Using this formula we get the weight/affinity of the pair of features. Thus at the end of this phase we get a csv file where two columns (named a_id & b_id) contains the features and a column (named consistency) contains the affinity/weight between the corrosponding features. However before moving onto the next phase the feature columns that contain the names of the features (column names) of the data must be mapped to numbers. This mapping must then be stored for future reference. Thus the output of this phase would be a csv file that woud look like this:

a_id	b_id	consistency
Feature F1	Feature F2	1.05698
Feature F1	Feature F3	1.6598
Feature F1	Feature F4	2.5687
Feature F1	Feature F5	0.9684
Feature F1	Feature F6	0.7894
:	:	:
:	:	:
:	:	:
Feature Fn	Feature F1	$\Phi(Fn.\mathrm{F1})$
Feature Fn	Feature F2	$\Phi(Fn.\mathrm{F2})$
Feature Fn	Feature Fn-1	$\Phi(Fn.\mathrm{Fn}-1)$

Thus, the Phase 1 can be seen in the architecture below:



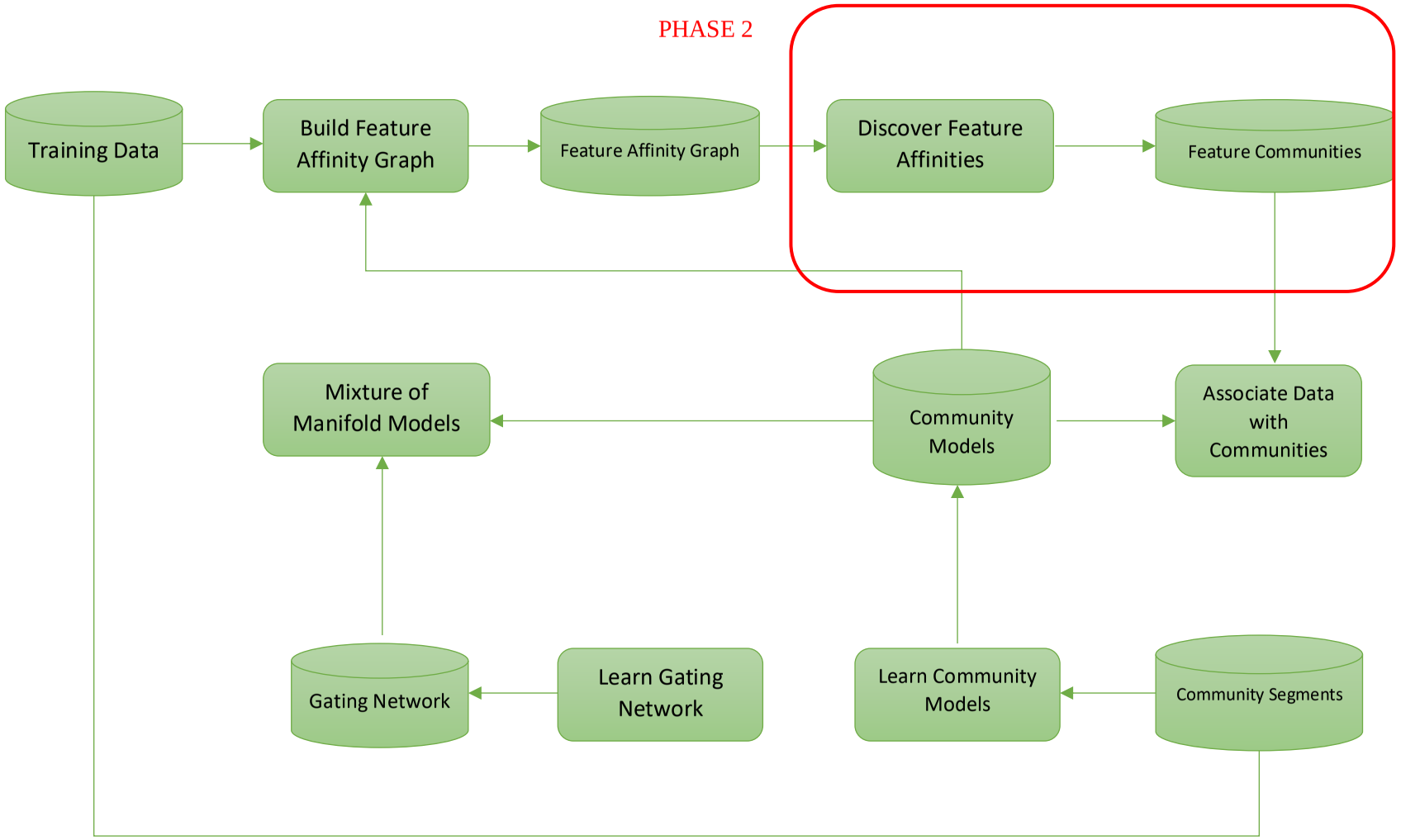
The flow of the First phase can be illustrated as follows:



PHASE 2 (Community Detection of features):

Note:

In the second phase the affinity graph is fetched into the community detection transformer which generates feature communities. The algorithm used to detect communities is the intellectual property of Reliance Jio and hence cannot be disclosed.


PHASE 2
Training Data
Build Feature Affinity Graph
Feature Affinity Graph
Discover Feature Affinities
Feature Communities
Mixture of Manifold Models
Community Models
Associate Data with Communities
Gating Network
Learn Gating Network
Learn Community Models
Community Segments

PHASE 3 (Modelling each Community):

In the third phase the parquet file containing the list of unique communities is taken as input. The main aim of this phase is to create models for each community individually. For this we first create a data frame for each community. Initially all data points are equally associated with all communities.

Based on this data each community is trained using a machine learning model. The machine learning models here are again hyper-parameterised. The ML models used to model the communities are:

a) Decision Tree

b) Random Forest

c) XGBoost

d) Logistic Regression

In order to then validate the model a model score is calculated using various methods. These methods are also hyper-parameterised. The methods for validating the accuracy are:

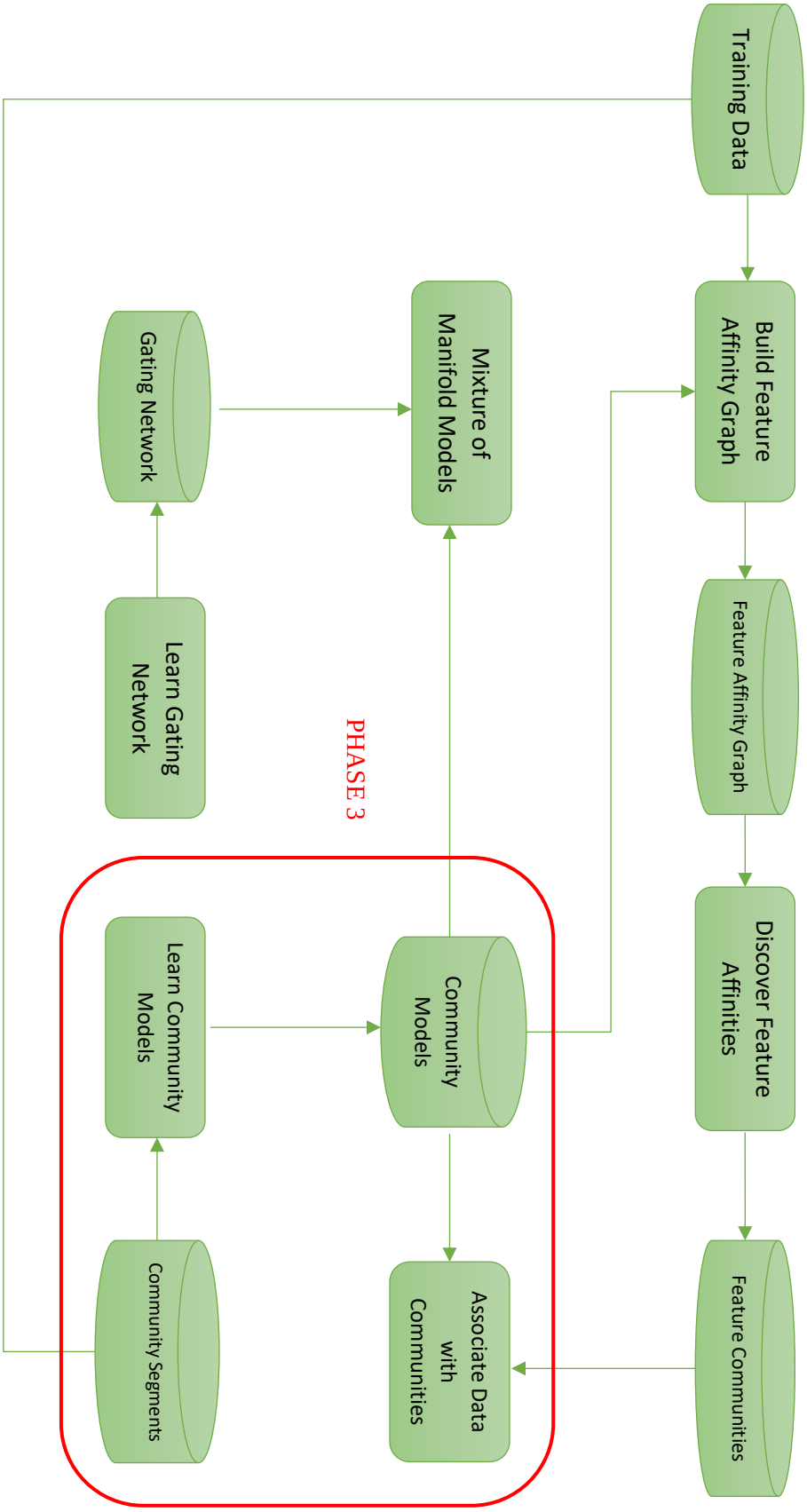
a) Accuracy

b) F1 Score

c) AUC

d) Precision

The phase 3 is represented in the architecture as follows



PHASE 4 (Gating Mechanism):

Note:

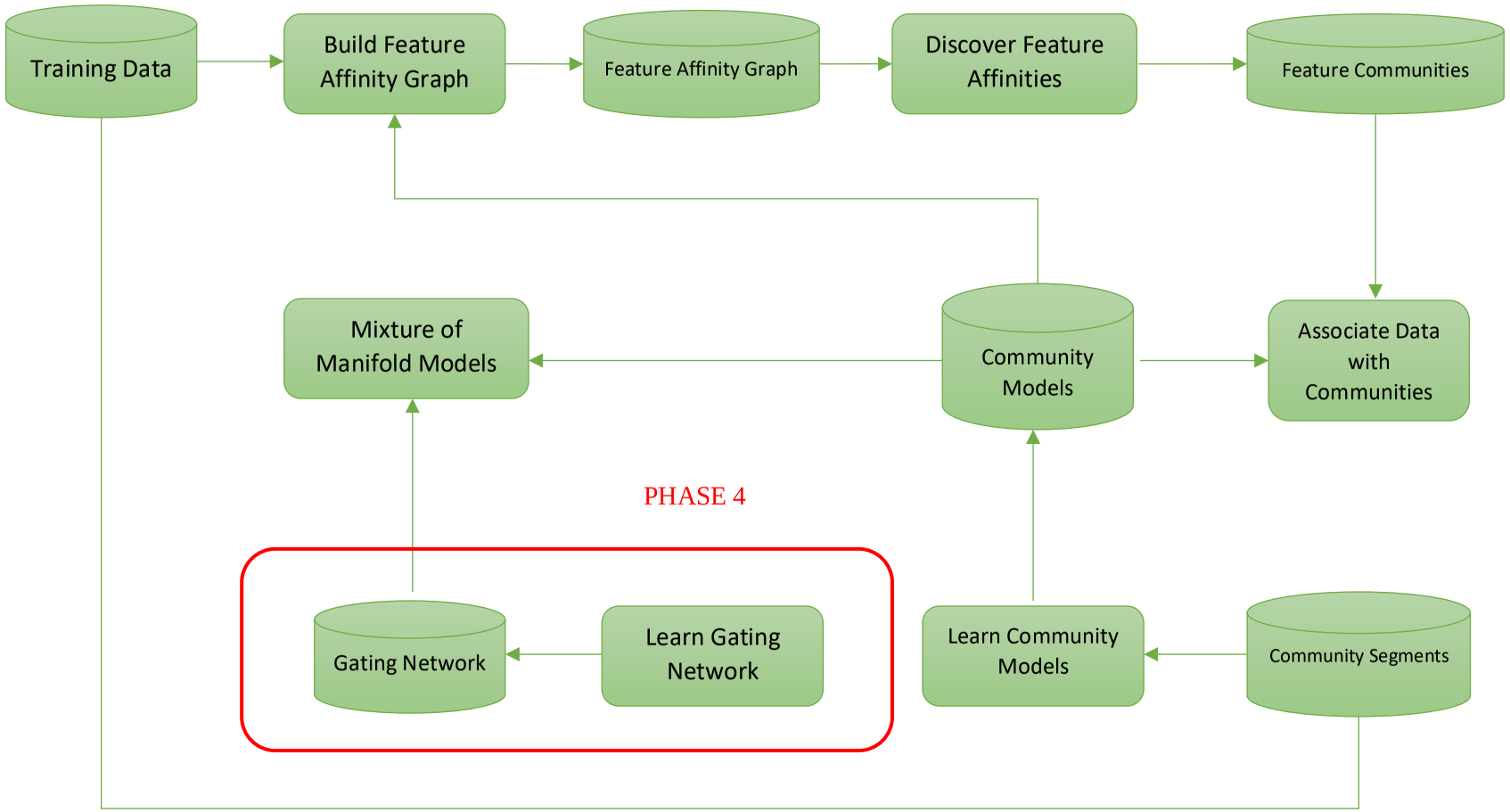
The actual implementation technique and logic of this phase is an intellectual property of Reliance Jio and hence cannot be made a part of this report. However, in order to make complete sense of this project we will be implementing a dummy method which isn't as efficient and reliable as the original algorithm but will be useful in understanding the flow of the algorithm.

The gating mechanism is the last phase of the Community Forest Algorithm and also the most important. The main aim of this phase is to take a data point as an input and assign it to a community. For this we will give the data as an input to each and every community model and based on a scoring criterion we will assign the data to a community.

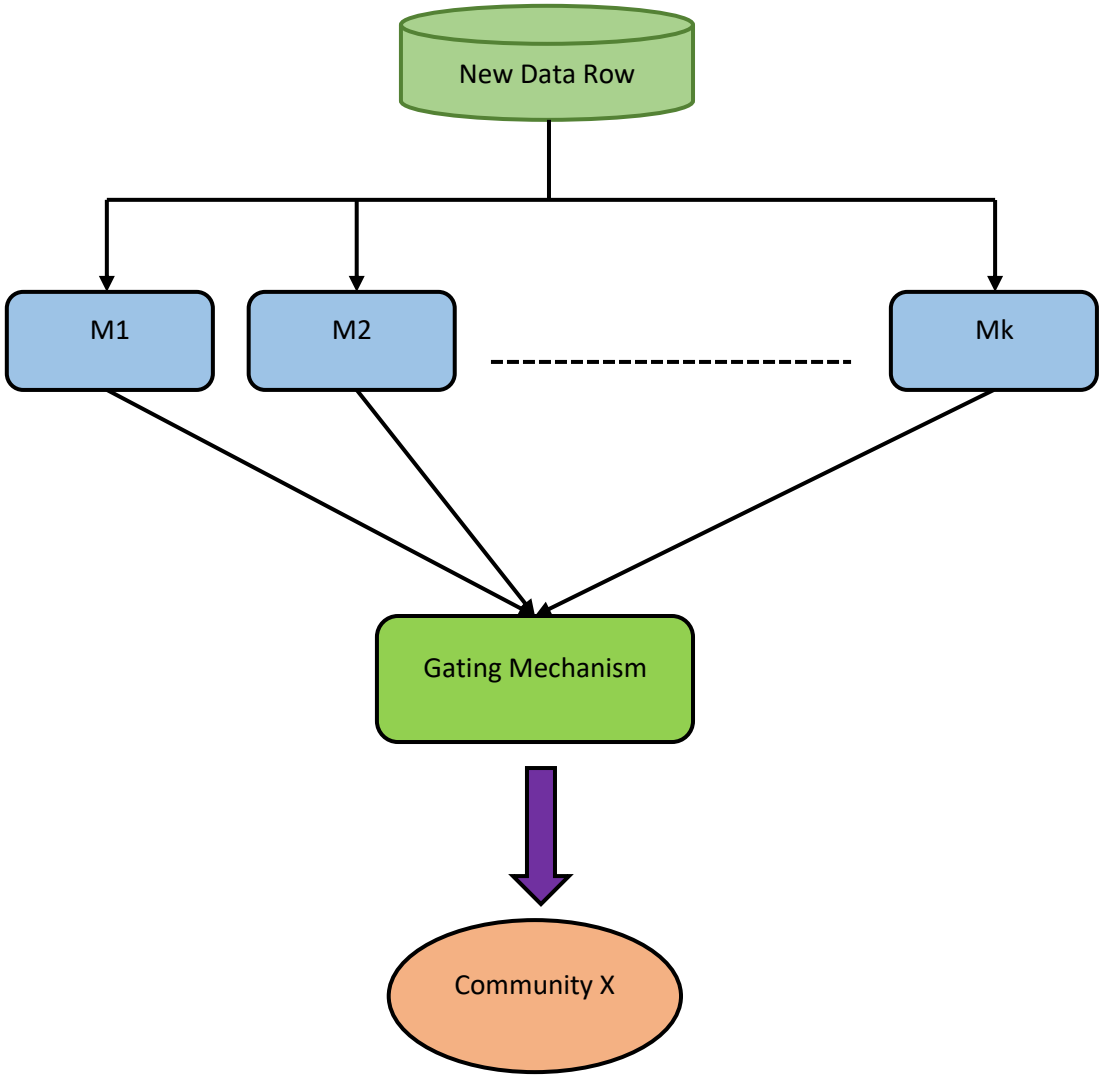
Currently for implementation purpose we will be choosing the community with the highest confidence score to be assigned the data. Thus, this Gating mechanism will help the user understand the data belongs to which community. With this module you can have real time implementation of various functions that will help transform businesses.

The steps of this phase can be summarised as below:

- The new data row is sent to each model corresponding to each community (M1, M2, M3,…….,Mk).
- Prediction for each row is calculated given the community.
  x= new data
  Mi= model for a community i
  prediction score = P(x|Mi)
- Max vote count is used to predict the community
- Max_count(P(x|M1), P(x|M2),….P(x|Mk))


Training Data
Build Feature Affinity Graph
Feature Affinity Graph
Discover Feature Affinities
Feature Communities
Mixture of Manifold Models
Community Models
Associate Data with Communities
PHASE 4
Gating Network
Learn Gating Network
Learn Community Models
Community Segments

The flow of Phase 4 can be seen below



CONCLUSIONS AND RECOMMENDATIONS

We propse a novel method to create logical cluster of features (called communities) and classifying data into them. It serves two purposes.

Firstly, we are able to interpret the communities and derive knowledge which may be helpful while making business decisions.

Secondly, in datasets with high dimensionality (>50), Community Forest has proven to classify records more accurately than XGBoost and Random Forest algorithms. Although the evaluations are done mainly for Telecom releted data, the generality of the process enables it to be applied for any business use case where the dimensionality is large

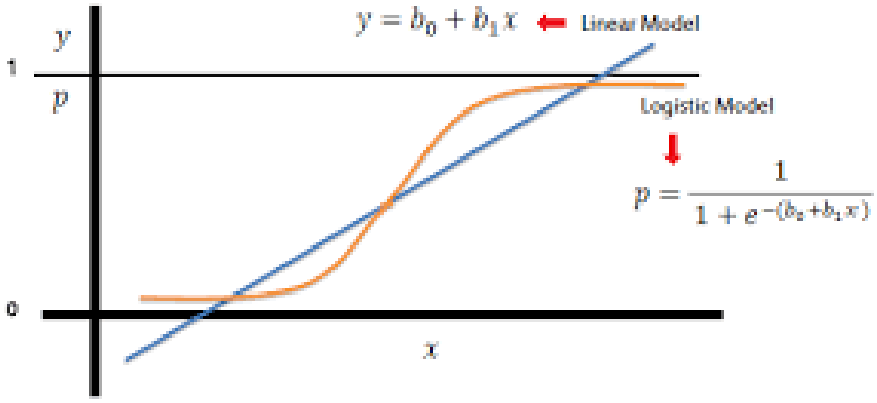
Currently due to a lack of a denoising module the graph is highly interconnected and thus the communities being formed overlap significantly. Thus, this leads to the community containing all the features of the data set. To avoid this, we currently have set a threshold of 20 features per community as a stop gap solution till the de-noising module is built.

Also due to the way the code has been written the number of communities is similar to the number of features of the dataset. Some more research needs to done on why such a limitation occurs and the code needs to be changed accordingly.

APPENDICES

Logistic Regression:

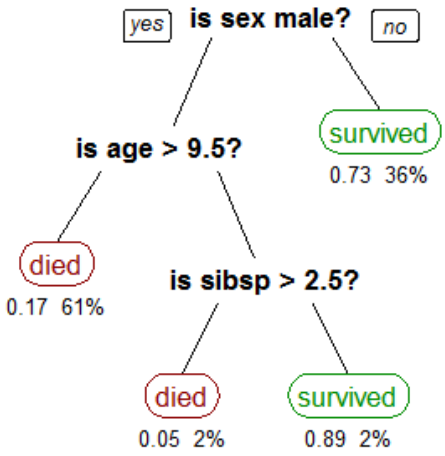
- Logistic Regression (Classification) is a predictive modelling algorithm that is used when the Y variable is binary categorical. That is, it can take only two values like 1 or 0. The goal is to determine a mathematical equation that can be used to predict the probability of event 1.
- In order to map predicted values to probabilities, we use the sigmoid function. The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.



- Logistic Regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic/sigmoid function.
- Logarithmic loss function to calculate the cost for misclassifying.
- Gradient Descent Algorithm is used for minimizing the cost function

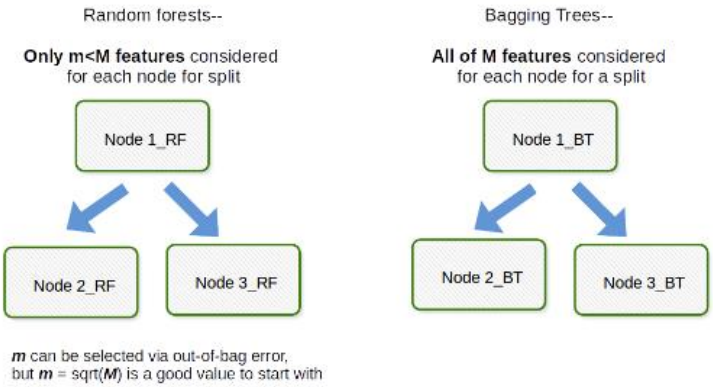
Decision Tree

- A decision tree is a largely used non-parametric effective machine learning modeling technique for regression and classification problems. To find solutions a decision tree makes sequential, hierarchical decision about the outcomes variable based on the predictor data.
- Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.



Random Forest:

- Random Forest is an improved version of Bagged Decision Trees.
- Decision Trees are considered to be algorithms with higher variance.
- In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e. a bootstrap sample) from the training set. **In addition, instead of using all the features, a random subset of features is selected, further randomizing the tree.**
- As a result, the bias of the forest increases slightly, but due to the averaging of less correlated trees, its variance decreases, resulting in an overall better model



REFERENCES

i. https://www.americanscientist.org/article/first-links-in-the-markov-chain

ii. https://stackoverflow.com/

iii. https://github.com/

iv. https://www.tutorialspoint.com/

v. https://www.getpostman.com/

vi. https://jsonlint.com/

vii. https://github.com/jupyter-widgets/ipyleaflet

i. ss

GLOSSARY

1. **DAM: Data Agnostic Model**
2. **JDSP: Jio Data Science Platform**
3. **JBDL: Jio Big Data Lake**
4. **HDFS: Hadoop Distributed File System**