TOPIC: IMPLEMENTATION OF NAÏVE BAYES ALGORITHM FOR BUILDING CHURN PREDICTION MODEL FOR TELECOMMUNICATION COMPANY.

TABLE CONTENT:

Abstract

In telecommunication companies, ‘churn’ means customers’ decision to move from one service provider to another. The competition environment in telecom companies' makes their aim is to maintains their customers who are likelihood to leave and earns their satisfaction, so to avoid the problem of churn they need churn predictive models.

Data mining techniques can be used to build churn prediction model for telecommunication companies to identify churner and non-churner customers because it can extract the predictive information from large databases.

Introduction:

Churn is a term used in many companies which is mean loss of customers of the company for many rezones one of them is the dissatisfaction of customer.

In telecommunication companies “churn” term refers to customer's decision to leave the current service provider and move to other service provider, it can be easily happen especially for prepaid customers because they have not any contract same as to postpaid customers.

Churn occurs easily because of the strong and breeding competition environment in services which are providing especially in telecommunication sector, also churn can be happen for another rezones for examples customer's dissatisfaction with services and high cost of these services which can be in another service provider with best quality and lower cost. So churn become a concern issue in telecom sector because retaining of existing customer is costly than acquiring new one.

To identify churner and non-churner customers and understand the rezones of this churn to reduce it, these companies can build churn prediction model which can help them in churn issue to build this model they can use Data mining (DT) this can be useful because DM can extracting a predictive information from large databases, it's have many techniques which can for example Decision Tree (DT), Support vector machine (SVM), Naïve Bayes etc.

Problem Statement: Churn is big issue for telecommunication companies so they need a churn prediction model to help them to identify churner and non-churner customers.

Problem Objective: The objective of this problem is to build a churn prediction model which can identify churner customers and non-churner customers and implementing this model by using Naïve Bayes algorithm.

Methodology: DM (Data Mining) is the process of extracting useful patterns from large databases. The problem in this research is to build a churn prediction model to identify subscriber as churner or non churner, so the DM classification is used to build this model.

The model contains four steps these steps in ordering are identification of the problem, acquisition the data, preparation of data which was acquisitioned, and finally implementing this model by classification technique and Naïve bayes algorithm.

Using data mining concept
Selecting classification technique for building the model
Building the model
Implementing model by using Naïve bayes algorithm

Fig: Methodology Model

Bayesian networks

Bayesian models are probability models that can be used in classification problems to estimate the likelihood of occurrences. They are graphical models that provide a visual representation of the attribute relationships, ensuring transparency, and an explanation of the model's rationale.

Naïve Bayes algorithm

Naïve Bayes is a simple probabilistic classifier which is simple and easy to understand, its deal with any number of features or classes, and it's strong so if there are a few noises in data it doesn't affect the results, depending on the nature of the probability of Naive Bayes classifier it can be trained efficiently in supervised learning.

What are the other Techniques? (Data Mining)

Data mining is one of advanced type of analytical tools at this time available; these tools can include statistical models, mathematical algorithms, and machine learning methods (algorithms that increase their performance automatically during experience, such as neural networks or decision trees) but data analysis system that does not deal with large amounts of data.

They can be grouped into two main models based on their goals:

1. **Supervised/Predictive Models:** In supervised, or predictive, directed, or targeted modeling, the goal is to predicting event or estimate the values of a continuous numeric attribute.

2. **Unsupervised Models:** In unsupervised or undirected models there is no output field, just inputs. The Pattern recognition is undirected; it is not

guided by a specific target attribute. The goal of such models is to uncover data patterns in the set of input fields.

Classification of Techniques

a. **Logistic Regression:** It uses the generalized linear model and calculates regression coefficients that represent the effect of predictors on the probabilities of the categories of the target field.

b. **Neural Networks:** Neural Networks is a data mining technique that has the capability of learning from errors. Based on the research it is calculated that neural network based approach can predict the customer churn accuracy of more than 92% but they need large amount of data sets and a lot of time to calculate a considerable load for the predictor attributes.

c. **Decision Trees:** Decision trees are the most common methods used in predicting and evaluating the classification of customer churn problems. To evaluate a customer churn analysis, the tree is altered or split until the leaf node is obtained. A simple decision tree for customer churn in telecom industry is as follows.

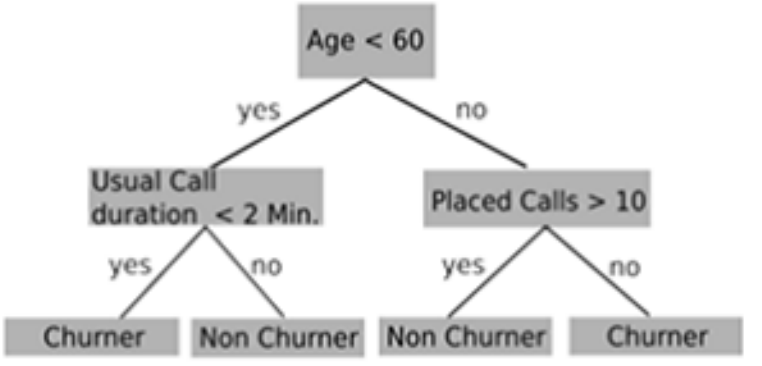
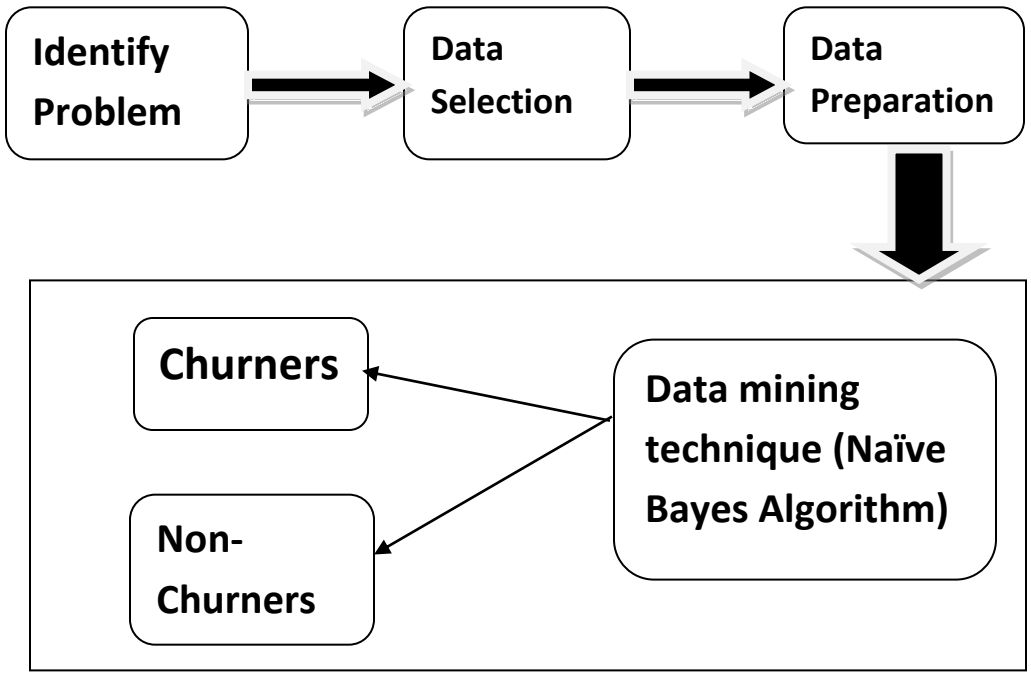


Fig: A simple decision tree for Customer Churn

The Proposed Model

The proposed model composed of four steps:



1. **Identify domain problem:** Churn prediction is a phenomenon which is used to identify the possible churners in advance before they leave the network. This helps the CRM department to prevent subscribers who are likely to churn in future by taking the required retention policies to attract the likely churners and to retain them. Thereby, the potential loss of the company could be avoided. This research utilizes data mining techniques to identify the churners.
2. **Data Selection:** Churn prediction models require the past history or the usage behavior of customers during a specific period of time to predict their behavior in the near future. Data Acquisition

is a difficult problem for the researchers to acquire the actual dataset from the telecom industries. This is because the customer's private details may be misused.

3. **Data Preparation:** After collection of required datasets, we require to prepare the dataset so that they can be processed by appropriate tools. These datasets are generally large volumes of unstructured data which cannot be handled by traditional data processing techniques. So for processing them we require special computing frameworks which process the data and makes the data ready for analysis to be applied on.
4. **Data mining Technique (Naïve Bayes algorithm):** In this step the model was implemented by using Naïve bayes algorithm in classification. Naïve bayes algorithm was implemented to predicts wither the customer will churn or not.

Description of the project (Implementation using Hadoop)

Hadoop : Hadoop is an open source, Java-based programming framework that supports the processing and storage of extremely large data sets in a distributed computing environment. It provides a MapReduce model to tackle large distributed data processing, by mapping data and reducing it to a result. Our main challenge is to write the appropriate logic for mapping of data and getting the required probabilities from there.

1. **Implementation of mapper, custom key, reducer class for calculation of the conditional probabilities.**

   Before calculating the probabilities, we need to find the no. of customers with specific attributes who have either churned or not. This requires mapping the attributes together with the state of churning (‘Yes’/ ‘No’) which is then reduced accordingly. For this,

   - At first, a composite data type named ‘CStype.java’ has been defined which stores a particular attribute of the record, along with state of churning for the same record.
   - Then, the mapper’s task is programmed to mark an entry if a combination of a specific combination of a state of attribute and state of churning appears or not. Here all attributes have different states w.r.t. each other. The mapper here is ‘PAtrriMapper.java’
   - After that the reducer groups such combinations to get the count of each combination in the set. The reducer here is ‘PAtrriReducer.java’

   Also, an additional ‘Yes,yes’, ‘No,no’ entry has been added to track the total no. of churnings (and vice versa in the set).

2. **Calculation of the probabilities of the training set’s data.**

   After getting the appropriate counts, we can easily get the probabilities of the obtained combinations by dividing the counts

with the total no. of ‘yes’ (or ‘no’ accordingly). From the files, we store the entries in a java data structure called ‘HashMap’ which stores the value for a combination occurring in the file. Then the division is performed, and the probabilities are written in a separate file. For processing, we stored them in form of records containing attribute and corresponding probability of ‘yes’ or ‘no’. Here this is performed by a program ‘ChurnProb.java’.

3. **Final mapper class creation for the calculation of the likelihood and Bayes theorem implementation for finding out the churning probability on the testing data set.**

   This is the final step where we require to process the probabilities and apply Naïve Bayes algorithm to find out the probability of a customer’s churning out, given his/her attributes. For each attribute, the conditional probability is extracted out from the probabilities already obtained from the previous step. Those probabilities are multiplied as they are independent events. This gives out the probability of a customer’s having that set of probabilities provided the churning state of customer. Then, finally, we determine the likelihood of the customer’s churning out which is used for getting the final probability of the customer’s churning out. This is implemented a mapper program called ‘TestMapper.java’. No reducer is required here.

   Then the probabilities are appended to the set’s records along the predicted decision. Additionally, we have also counted the no of records where the predicted decision meets the actual decision

and this is marked as the accuracy of the system. All these have been done in program ‘AppendPr.java’.

Program Codes

Code for CStype.java

```
package churn;

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.WritableComparable;

public class CStype implements WritableComparable<CStype>
{
	private Text attribute=new Text();
	private Text churn=new Text();

	public void readFields(DataInput in) throws IOException {
		// TODO Auto-generated method stub
		attribute.readFields(in);
		churn.readFields(in);

	}

	public void write(DataOutput out) throws IOException {
		// TODO Auto-generated method stub
		attribute.write(out);
		churn.write(out);
	}

	public int compareTo(CStype o) {
		// TODO Auto-generated method stub
		int c=attribute.compareTo(o.attribute);
		if(c==0)
		{
			c=churn.compareTo(o.churn);
		}

		return c;
	}

	@Override
	public int hashCode()
	{
```

```
		int a=attribute.hashCode();
		int c=churn.hashCode();
		int hc=a*31+c; //a+c , a*c
		return hc;
	}

	@Override
	public String toString()
	{
		// TODO Auto-generated method stub
		StringBuilder sb = new StringBuilder();
		//sb.append("[");
		sb.append(attribute.toString()+","+churn.toString());
		//sb.append("]");
		String r=sb.toString();
		return r;
	}


	public void set(String attri, String chur)
	{
		attribute.set(attri);
		churn.set(chur);

	}

	public void set(Text attri, Text chur)
	{
		attribute.set(attri);
		churn.set(chur);

	}

}
```

Code for PAtrriMapper.java

```
package churn;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class PAtrriMapper extends
Mapper<LongWritable,Text,CStype,IntWritable>
```

```
{
	private CStype keyout=new CStype();
	private IntWritable valueout=new IntWritable(1);
	@Override
	protected void map(LongWritable key, Text value,Context context)
			throws IOException, InterruptedException
	{
		// TODO Auto-generated method stub
		String rec = value.toString();//hadoop doesnt undrstand text
so string
		String  f[] = rec.split("  ");
		for(int i=0;i<f.length;i++)
		{
			//if(f[i].trim().length()!=0)
				keyout.set(f[0],f[i]);
				context.write(keyout,valueout);
		}

	}
}
```

Code for PAtrriReducer.java

```
package churn;

import java.io.IOException;

import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Reducer;

public class PAtrriReducer extends
Reducer<CStype,IntWritable,CStype,IntWritable>
{

	@Override
	protected void reduce(CStype key, Iterable<IntWritable>
values,Context context)
			throws IOException, InterruptedException {
		// TODO Auto-generated method stub
		int sum = 0;
		for(IntWritable v : values)
			sum = sum + v.get();
		context.write(key,new IntWritable(sum));
	}

}
```

Code for ChurnProb.java

```
package churn;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.HashMap;
import java.util.Map;

public class ChurnProb
{
	public static void main(String[] args) throws IOException
	{
		HashMap<String,Double> hm1 = new HashMap<String,Double>();
		HashMap<String,Double> hm2 = new HashMap<String,Double>();
		BufferedReader br=new BufferedReader(new FileReader("part-r-
00000"));
		double yes=0.0,no=0.0;
		while(true)
		{
			String ln = br.readLine();
			if(ln==null)	break;
			String rc[]=ln.split(",");
			String det[]=rc[1].split("\t");
			if(rc[0].equals(det[0]))
			{
				if(rc[0].equals("yes"))
					yes=Double.parseDouble(det[1]);
				else if(rc[0].equals("no"))
					no=Double.parseDouble(det[1]);
				continue;
			}
			else if(rc[0].equals("yes"))
				hm1.put(det[0], new Double(det[1]));
			else if(rc[0].equals("no"))
				hm2.put(det[0], new Double(det[1]));
		}
		for(Map.Entry<String, Double> etr:hm1.entrySet())
			etr.setValue((etr.getValue())/yes);
		for(Map.Entry<String, Double> etr:hm2.entrySet())
			etr.setValue((etr.getValue())/no);

		hm1.put("yes", (yes/(yes+no)));
		hm2.put("no", (no/(yes+no)));

		PrintWriter pw=new PrintWriter(new FileWriter("Probability"));

		for(Map.Entry<String, Double> etr:hm1.entrySet())
		{
			if(!(etr.getKey()).equals("yes"))
```

```
	pw.println(etr.getKey()+",Yes,"+etr.getValue()+",No,"+hm2.get(etr.ge
tKey()));
			}
			pw.println("total,Yes,"+hm1.get("yes")+",No,"+hm2.get("no"));

			pw.close();
			br.close();
	}
}
```

Code for TestMapper.java

```
package churn;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Mapper;

public class TestMapper extends
Mapper<LongWritable,Text,Text,DoubleWritable>
{
	public HashMap<String,double[]> hm= new HashMap<String,double[]>();
	double t_y=1,t_n=1, l_y=0, l_n=0, ev=0, f_y=0, f_n=0;

	public void readFile(Path p)
	{
		try{
			BufferedReader br = new BufferedReader(new
FileReader(p.toString()));
			String rec = null;
			while(true)
			{
				rec=br.readLine();
				if(rec==null)
					break;
				if(rec.trim().length()==0)
					continue;
				String ent[] = rec.split(",");
				double ar[] = new double[2];
				ar[0]=Double.parseDouble(ent[2]);
				ar[1]=Double.parseDouble(ent[4]);
				if(!ent[0].equals("total"))
					hm.put(ent[0].trim(),ar);
				else
				{
					t_y=Double.parseDouble(ent[2]);
```

```
                        t_n=Double.parseDouble(ent[4]);
                  }
            }
      }
      catch(Exception e){}
}

@Override
protected void setup(Context context) throws IOException,
            InterruptedException {
      // TODO Auto-generated method stub
      @SuppressWarnings("deprecation")
      Path [] paths=context.getLocalCacheFiles();
      if(paths!=null && paths.length!=0)
      {
            for(Path p:paths)
            {
                  readFile(p);
            }
      }
}
@Override
protected void map(LongWritable key, Text value,
            Context context)
            throws IOException, InterruptedException
{
      // TODO Auto-generated method stub

      String line = value.toString();
      String atrs[] = line.split("  ");
      double pr_y=1f,pr_n=1f;

      for(int i = 1; i <= 6; i++)
      {
            if(hm.containsKey(atrs[i]))
            {
                  pr_y *= hm.get(atrs[i])[0]; //0 - yes
                  pr_n *= hm.get(atrs[i])[1]; //1 - no
            }
      }

      l_y = pr_y/t_y;
      l_n = pr_n/t_n;

      ev = l_y+l_n;  //estimated values p[t]

      f_y = l_y/ev; //prob of final yes
      f_n = l_n/ev; //prob of final no

      if(f_y>f_n)
            context.write(new Text("yes"), new DoubleWritable(f_y));
      else
            context.write(new Text("no"), new DoubleWritable(f_n));
```

```
	}
}
```

Code for AppendPr.java

```
package churn;

import java.io.*;

public class AppendPr {

	/**
	 * @param args
	 * @throws IOException
	 */
	public static void main(String[] args) throws IOException {
		// TODO Auto-generated method stub
		BufferedReader br1 = new BufferedReader(new
FileReader("finalProb"));
		BufferedReader br2 = new BufferedReader(new
FileReader("trainingSet"));
		PrintWriter pw = new PrintWriter(new FileWriter("fop_tr"));
		int cnt=0;
		while(true)
		{
			String l1 = br1.readLine();
			String l2 = br2.readLine();
			if(l1==null || l2==null) break;
			if(l2 != null)
			{
				StringBuilder sb = new StringBuilder(l2);
				sb.append("  "+l1);
				pw.write(sb.toString()+"\n");
				String ar[]=sb.toString().split("  ");
				if(ar[7].split("\t")[0].endsWith(ar[0]))
					cnt++;
			}
		}
		pw.write("\n"+cnt+" out of 85 records agree with the
formulation.\nAccuracy in % : "+((cnt/85.0)*100));
		pw.close();
		br2.close();
		br1.close();
	}

}
```

Code for PAtrriDriver. Java

```
package churn;

import java.io.IOException;
```

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class PAtrriDriver
{
      public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException
      {
            // TODO Auto-generated method stub
            Configuration conf = new Configuration();
            Job j = Job.getInstance(conf,"composite");

            j.setJarByClass(PAtrriDriver.class);

            j.setMapperClass(PAtrriMapper.class);
            j.setReducerClass(PAtrriReducer.class);

            j.setOutputKeyClass(CStype.class);
            j.setOutputValueClass(IntWritable.class);

            //j.setOutputKeyClass(CStype.class);
            //j.setOutputValueClass(IntWritable.class);

            //file location : /user/hadoop/TrainingFile/trainingSet

            FileInputFormat.addInputPath(j, new
Path("smp_data/trainingSet"));//folder name and data set name
            FileOutputFormat.setOutputPath(j, new
Path("ATrriProbeFile1"));

            j.waitForCompletion(true);
      }
}
```

Code for TestDriver.java

```
package churn;

import java.io.IOException;
import java.net.URI;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
```

```
public class TestDriver
{
      public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException
      {
            // TODO Auto-generated method stub
            Configuration conf = new Configuration();
            Job j = Job.getInstance(conf,"composite");

            Path p = new Path("Probability");
            URI uri = p.toUri();
            j.addCacheFile(uri);

            j.setJarByClass(TestDriver.class);

            j.setMapperClass(TestMapper.class);

            j.setOutputKeyClass(Text.class);
            j.setOutputValueClass(DoubleWritable.class);

            j.setNumReduceTasks(0);

            //j.setOutputKeyClass(CStype.class);
            //j.setOutputValueClass(IntWritable.class);

            //file location : /user/hadoop/TrainingFile/trainingSet

            FileInputFormat.addInputPath(j, new
Path("smp_data/trainingSet"));//folder name and data set name
            FileOutputFormat.setOutputPath(j, new
Path("ATrriProbeFile3"));

            j.waitForCompletion(true);
      }
```

Outputs

1. Getting the counts of (Attribute, Churn state pairs)

Counts with 'No'	Counts with 'Yes'
no,AboveNormal 16	yes,AboveNormal 21
no,Android 14	yes,Android 14
no,Basic 14	yes,Basic 13
no,Frequent 15	yes,Frequent 12
no,High 21	yes,High 11
no,IOS 19	yes,IOS 16
no,Less 27	yes,Less 15
no,Low 11	yes,Low 10
no,Medium 14	yes,Medium 18
no,Middle 12	yes,Middle 10
no,More 19	yes,More 24
no,Normal 30	yes,Normal 18
no,Old 20	yes,Old 16
no,Regular 17	yes,Regular 14
no,Windows 13	yes,Windows 9
no,Young 14	yes,Young 13
no,no 46	yes,yes 39

2. Getting probabilities for the attributes

```
Medium,Yes,0.46153846153846156,No,0.30434782608695654
Low,Yes,0.2564102564102564,No,0.2391304347826087
Regular,Yes,0.358974358974359,No,0.3695652173913043
Middle,Yes,0.2564102564102564,No,0.2608695652173913
Young,Yes,0.3333333333333333,No,0.30434782608695654
AboveNormal,Yes,0.5384615384615384,No,0.34782608695652173
Less,Yes,0.38461538461538464,No,0.5869565217391305
Windows,Yes,0.23076923076923078,No,0.2826086956521739
Frequent,Yes,0.3076923076923077,No,0.32608695652173914
Basic,Yes,0.3333333333333333,No,0.30434782608695654
Android,Yes,0.358974358974359,No,0.30434782608695654
IOS,Yes,0.41025641025641024,No,0.41304347826086957
More,Yes,0.6153846153846154,No,0.41304347826086957
Normal,Yes,0.46153846153846156,No,0.6521739130434783
Old,Yes,0.41025641025641024,No,0.43478260869565216
High,Yes,0.28205128205128205,No,0.45652173913043476
total,Yes,0.4588235294117647,No,0.5411764705882353
```

3. Final results : Probabilities of customer's churning out in the given data set, and the accuracy of the work.

```
yes Medium Middle More Normal IOS Frequent yes 0.5956245625249935
yes High Young Less AboveNormal Android Frequent no 0.5669095416778691
no Medium Old More Normal IOS Regular yes 0.5927709670591954
yes High Old More Normal Windows Regular no 0.6722510154399826
yes Medium Middle More Normal IOS Basic yes 0.6309518367577823
no Medium Old Less Normal Android Basic no 0.5384417069130486
no High Young Less Normal Android Basic no 0.7115608712398052
no Medium Young Less AboveNormal Android Frequent yes 0.652192014478589
no Low Old Less AboveNormal Windows Regular no 0.5512466569900888
no Medium Young Less Normal Android Regular no 0.5312301283461133
yes High Old More AboveNormal Android Basic yes 0.6346338879028417
yes Medium Middle Less Normal Windows Frequent no 0.652491558797426
no Medium Middle More Normal Android Regular yes 0.642930131772295
yes Low Old Less Normal IOS Regular no 0.6883879958608914
no High Young More Normal Windows Frequent no 0.6452754572909263
yes High Old Less Normal Windows Basic no 0.8052972229817486
no Medium Middle Less Normal IOS Basic no 0.5707959011750056
yes High Middle Less AboveNormal Android Basic no 0.5568607468415684
no Medium Young Less Normal Windows Regular no 0.6207677313461104
no High Young More Normal IOS Frequent no 0.5992788497499199
no High Young More AboveNormal IOS Frequent yes 0.5939450046850939
no High Old More Normal Android Regular no 0.586777375842176
no High Old Less AboveNormal Windows Frequent no 0.6869747103485899
yes Low Old More Normal Android Regular yes 0.5499968539142648
no High Middle More Normal Android Frequent no 0.5839049622616459
yes Medium Old Less Normal Android Regular no 0.568103814004461
yes Low Young More Normal Android Regular yes 0.5865432668132214
yes Medium Old Less Normal Windows Basic no 0.6275675561475796
yes Low Young More Normal IOS Frequent yes 0.5371453785943276
yes High Middle Less AboveNormal IOS Basic no 0.5987552671425517
no Low Middle Less AboveNormal IOS Frequent yes 0.5005000018080978
yes Medium Old More Normal Windows Basic yes 0.5743461617835401
no High Middle Less Normal IOS Regular no 0.7863543606292673
no High Old Less AboveNormal Windows Basic no 0.6540699952173937
yes Medium Young More AboveNormal Android Frequent yes 0.8100122806282269
yes High Young More Normal IOS Regular no 0.5922980742904906
no High Young Less AboveNormal IOS Regular no 0.6015951126751834
yes Medium Old More AboveNormal IOS Regular yes 0.761003977396526
no High Middle More Normal IOS Regular no 0.6181462182441512
yes High Middle More AboveNormal IOS Regular yes 0.5747045736965547
yes Low Old More Normal Windows Basic no 0.511752523365591
yes Medium Young Less AboveNormal Android Regular yes 0.6587381397751985
no Low Young More AboveNormal Windows Basic yes 0.7078109811468873
no Low Young More Normal Android Basic yes 0.6153225960459318
no High Middle Less Normal Windows Frequent no 0.8217062997462603
yes High Middle Less AboveNormal IOS Regular no 0.6272247972576196
no High Middle Less Normal Android Basic no 0.7332530635274568
yes Medium Young More Normal Android Frequent yes 0.660905186146784
no Low Middle Less Normal Windows Frequent no 0.7264397891386577
no Low Old More Normal IOS Basic yes 0.5371453785943275
```

```
yes  Medium  Old  Less  Normal  IOS  Regular  no    0.6096804443052227
yes  Medium  Young  More  AboveNormal  IOS  Frequent  yes   0.7821499882879484
no  High  Old  More  AboveNormal  IOS  Frequent  yes    0.557559330371923
no  High  Old  Less  AboveNormal  Windows  Regular  no  0.6807076891263336
no  Medium  Young  Less  Normal  Android  Regular  no   0.5312301283461133
yes  Medium  Old  Less  AboveNormal  IOS  Basic  yes    0.612263841951686
no  Low  Old  Less  AboveNormal  Android  Basic  yes    0.5700521313379797
no  Medium  Young  More  Normal  Android  Basic  yes    0.6934644441373568
no  Low  Old  Less  Normal  IOS  Frequent  no   0.6945719085593169
no  Low  Old  More  AboveNormal  IOS  Frequent  yes 0.686235795135218
yes  Medium  Old  Less  AboveNormal  IOS  Frequent  yes 0.5763488155426196
no  High  Middle  More  Normal  Windows  Basic  no  0.6358770737819066
yes  Low  Middle  More  AboveNormal  IOS  Basic  yes    0.7256049975367872
yes  Medium  Young  Less  AboveNormal  Android  Frequent  yes   0.652192014478589
no  High  Old  Less  Normal  IOS  Frequent  no  0.7978480157345176
no  Medium  Middle  Less  Normal  IOS  Frequent  no 0.6068605687390454
yes  Medium  Old  More  AboveNormal  IOS  Frequent  yes 0.7556919955947486
yes  Medium  Young  More  AboveNormal  Windows  Basic  yes  0.7740636482084727
no  High  Middle  Less  Normal  Windows  Frequent  no   0.8217062997462603
yes  Medium  Middle  More  AboveNormal  Windows  Basic  yes 0.7545794284149848
no  Medium  Old  More  Normal  Android  Regular  yes    0.6335049021090581
no  Low  Old  Less  Normal  IOS  Basic  no  0.6620729599015504
no  High  Old  Less  Normal  Windows  Regular  no   0.8234335116015116
yes  Low  Middle  More  AboveNormal  Windows  Regular  yes  0.6584769936177709
yes  High  Young  More  AboveNormal  Android  Frequent  yes 0.6346338879028417
```

```
yes  Medium  Young  Less  AboveNormal  Android  Frequent  yes   0.652192014478589
no  High  Old  Less  Normal  IOS  Frequent  no  0.7978480157345176
no  Medium  Middle  Less  Normal  IOS  Frequent  no 0.6068605687390454
yes  Medium  Old  More  AboveNormal  IOS  Frequent  yes 0.7556919955947486
yes  Medium  Young  More  AboveNormal  Windows  Basic  yes  0.7740636482084727
no  High  Middle  Less  Normal  Windows  Frequent  no   0.8217062997462603
yes  Medium  Middle  More  AboveNormal  Windows  Basic  yes 0.7545794284149848
no  Medium  Old  More  Normal  Android  Regular  yes    0.6335049021090581
no  Low  Old  Less  Normal  IOS  Basic  no  0.6620729599015504
no  High  Old  Less  Normal  Windows  Regular  no   0.8234335116015116
yes  Low  Middle  More  AboveNormal  Windows  Regular  yes  0.6584769936177709
yes  High  Young  More  AboveNormal  Android  Frequent  yes 0.6346338879028417
no  High  Old  More  AboveNormal  IOS  Regular  yes 0.5646977663108965
no  Medium  Young  More  Normal  IOS  Basic  yes    0.6557735841136714
no  Medium  Old  Less  AboveNormal  Android  Basic  yes 0.6521920144785889
no  Low  Young  Less  Normal  Windows  Regular  no  0.6983457295744653
yes  High  Young  Less  AboveNormal  Android  Basic  no 0.5300182452027686
yes  Low  Old  More  AboveNormal  Android  Regular  yes 0.7277857614688436
no  Medium  Old  More  AboveNormal  IOS  Regular  yes   0.761003977396526
yes  Low  Old  More  Normal  IOS  Frequent  no  0.5000449524466783
yes  Low  Young  More  Normal  Android  Regular  yes    0.5865432668132214
no  High  Old  Less  AboveNormal  IOS  Regular  no  0.636718918210655

53 out of 85 records agree with the formulation.
Accuracy in % : 62.35294117647059
```

Future Enhancement:

The efficiency that we have got after the calculation is almost 62.3% which can be enhanced using some developed formula and further calculations, so that it will be easier for us to calculate the churn probability of a customer, which will provide better stability and better flexibility.

Conclusion:

Customer churn is a big issue in telecom companies especially for prepaid subscribers because it happen easily under light of strong competition in this business area, so these companies need to build a churn prediction model t identify churner and non-churner customer and avoid this churn.

In this study the churn prediction model was built, this model contains of four steps which in ordering are: identify problem which is the churn problem in telecom companies, data selection in this step the data which is given to us by the faculty, data preparation, implementation of Naïve Bayes algorithm.

The model was built by treading its four steps and using the data which selected and Naïve bayes algorithm to implement it the result was there is from 100 customers correctly predicted with a certain level of accuracy.

Certificate

This is to certify that Mr. *DHRITIMAN SOME* of *B.P PODDAR INSTITUTE OF MANAGEMENT & TECHNOLOGY*, registration number: ***151150120005 OF 2015-2016***, has successfully completed a project on *CUSTOMER CHURN ANALYSIS* using *BIG DATA TECHNOLOGY* under the guidance of Mr. *TITAS ROYCHOUDHURY*.

--- ----------

-

Titas Roychowdhury

Globsyn Finishing School

Certificate

This is to certify that Mr. *ARPAN GUIN* of B.P. PODDAR INSTITUTE OF MANAGEMENT & TECHNOLOGY, registration number: ***141150110028 OF 2014-2015***, has successfully completed a project on *CUSTOMER CHURN ANALYSIS* using *BIG DATA TECHNOLOGY* under the guidance of Mr. *TITAS ROYCHOUDHURY*.

--- ----------

-

Titas Roychowdhury

Globsyn Finishing School

Certificate

This is to certify that Mr. *ARUNAVA BANERJEE* of *B.P PODDAR INSTITUTE OF MANAGEMENT & TECHNOLOGY*, registration number: ***141150110030 OF 2014-2015***, has successfully completed a project on *CUSTOMER CHURN ANALYSIS* using *BIG DATA TECHNOLOGY* under the guidance of Mr. *TITAS ROYCHOUDHURY*.

--- ----------

-

Titas Roychowdhury

Globsyn Finishing School

Certificate

This is to certify that Mr. *DEBANJAN DEY* of B.P. PODDAR INSTITUTE OF MANAGEMENT & TECHNOLOGY, registration number: ***141150110137 OF 2014-2015***, has successfully completed a project on *CUSTOMER CHURN ANALYSIS* using *BIG DATA TECHNOLOGY* under the guidance of Mr. *TITAS ROYCHOUDHURY*.

--- ----------

-

Titas Roychowdhury

Globsyn Finishing School

Certificate

This is to certify that Mr. *SANKAR PRASAD BISWAS* of BRAINWARE GROUP OF INSTITUTIONS-SDET, registration number: ***152700120006 OF 2015-2016***, has successfully completed a project on *CUSTOMER CHURN ANALYSIS* using *BIG DATA TECHNOLOGY* under the guidance of Mr. *TITAS ROYCHOUDHURY*.

--- ----------

-

Titas Roychowdhury

Globsyn Finishing School