International Conference on Emerging Trends in Engineering, Science and Management
Sphoorthy Engineering College, Hyderabad, India (ESM-17)
17th and 18th March 2017 , www.conferenceworld.in ISBN: 978-93-86171-32-0

BUILDING PREDICTIVE MODEL USING PYTHON & R FOR ENHANCING CONSUMER SATISFACTION AND COMPETITIVE ADVANTAGE

Dr.V.V.Narendra Kumar[#1], Dr.K.Kondaiah[*2], Regula Thirupathi

Department of CSE/IT, [1&2]St.Mary's Group of Institutions, [3]Highercollege of Tech,Muscat

ABSTRACT

In the deregulated markets empowered consumers expect innovative and personalised services. Hence there is a great need for the enterprises to develop innovative models to enhance the consumer satisfaction and thereby gain a competitive advantage over others. Applying Data mining and statistical techniques will aid us in developing several novel models. Predictive Analytics and predictive modelling can play a key role in optimizing consumer relation management.. In this article we have enlightened the use of data mining techniques of business intelligence and the use data mining languages namely Python and R in developing predictive models.

Keywords: Predictive Modelling, Predictive Analytics, Consumer satisfaction, Data mining,

I. INTRODUCTION

Customer behaviour is used to help make key business decisions through market segmentation and predictive analytics. This can be termed as consumer analytics. This can be used by businesses for customer relationship management. Customer analytics is very important in the prediction of customer behaviour today. Consumer analytics is widely used in retail, finance, and community and consumer relationship management.Forecasting buying habits and lifestyle preferences is a process of data mining and analysis. Through customer analytics, companies can make decisions with confidence because every decision is based on facts and objective Data.

There are two types of categories of data mining. Predictive models use previous customer interactions to predict future events while segmentation techniques are used to place customers with similar behaviours and attributes into distinct groups. This grouping can help marketers to optimize their campaign management and targeting processes.

II. PREDICTIVE MODELLING

Predictive modelling[1] is the practice of forecasting future customer behaviours and tendencies and assigning a score or ranking to each customer that depicts their probable actions.

One important question in predictive modelling is how many different models will be required. Each model is often devoted to predicting a single behaviour. For example, which customers are most likely to buy a specific

product or which customers will spend the most money across your entire portfolio of products over the next twelve months? Two separate predictive models would be required to effectively address each business question. The answer to the number of models a company needs is linked to the number of different profit-driving behaviours a company believes they can influence with customer data-driven campaigns.

Few common applications for predictive modelling are:

1. Identifying targets for marketing campaigns.
2. Forecasting customer behaviours.
3. Optimizing effectiveness of marketing levers and supporting 'what-if' analysis.
4. Measuring impact of specific marketing elements and treatments on subsequent customer behaviour.

Predictive models play an important role as companies attempt to optimize the usage of some of their primary marketing levers, such as value proposition, price, channel and media mix. Many companies are turning to predictive modelling (marketing mix models) to better understand the impact of advertising across different channels so that media mix investments are informed by quantitative measures of expected yield.

III. BUILDING A SIMPLE PREDICTIVE MODEL

Predictive modelling is the process of creating, testing and validating a model to best predict the probability of an outcome. Several modelling methods from machine learning, artificial intelligence, and statistics are present in predictive analytics software solutions for doing this task.

Predictive modelling[3] is a process that uses data mining and probability to forecast outcomes. Each model is made up of a number of predictors, which are variables that are likely to influence future results. Once data has been collected for relevant predictors, a statistical model is formulated. The model may employ a simple linear equation or it may be a complex neural network, mapped out by sophisticated software. As additional data becomes available, the statistical analysis model is validated or revised.

Let us build a small predictive model using Iris data set of WEKA. This example in machine learning can provide you a good clean dataset and easy to understand. We shall use this example for classifying plant species based on flower measurements

There are three steps in building a predictive model. They are

1. **Collect Sample Data**: the data we collect must describe our problem with known relationships between inputs and outputs.
2. **Create a Model**: the algorithm that we use on the sample data to create a model that we can later use over and over again.
3. **Make Predictions**: the use of our learned model on new data for which we don't know the output.

Let us analyse how these steps should be carried out for predictive modelling

COLLECT SAMPLE DATA

Let us assume that we want to identify the species of flower from the dimensions of a flower provided by Iris Data set. The data is comprised of four flower measurements in centimetres; these are the columns of the data like sepal length & width, petal length & width and species. Each row of data is one example of a flower that has been measured and its known species. The problem we are solving is to create a model from the sample data

ISBN: 978-93-86171-32-0

that can tell us which species a flower belongs to from its measurements alone. Fig. 1 provides the sample data taken in a CSV file build through excel.

	A	B	C	D	E
1	Sepal length	Sepal width	Petal length	Petal width	Species
2	5.1	3.5	1.4	0.2	I. setosa
3	4.9	3	1.4	0.2	I. setosa
4	4.7	3.2	1.3	0.2	I. setosa
5	4.6	3.1	1.5	0.2	I. setosa
6	5	3.6	1.4	0.2	I. setosa
7	5.4	3.9	1.7	0.4	I. setosa
8	4.6	3.4	1.4	0.3	I. setosa
9	5	3.4	1.5	0.2	I. setosa

Fig. 1. Sample of Iris flower data

CREATE A MODEL

Here we apply the concept of the supervised learning in data mining.The objective of a supervised learning algorithm is to take some data with a known relationship (actual flower measurements and the species of the flower) and to create a model of those relationships. The output here is a category (flower species) and this is called a classification problem. If the output is a numerical value, we call it a regression problem.The algorithm does the learning. The model (Fig. 2.) contains the learned relationships.The model itself may be a handful of numbers and way of using those numbers to relate input (flower measurements in centimetres) to an output (the species of flower).We shall retain this model after we have learned it from our sample data.

Used by output is

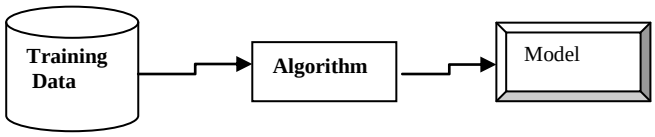

Fig. 2. Create a predictive model from training data and an algorithm.

Make Predictions

There is no need to keen the training data as the model has summarized the relationships contained within it. We retain this model learned from data to make predictions. In this example, we used the model by taking measurements of specific flowers of which don't know the species. The model (Fig. 3) will read the input (new measurements), perform a calculation of some kind with its internal numbers and make a prediction about which species of flower it happens to be. The prediction may not be perfect, but if you have good sample data and a robust model learned from that data, it will be quite accurate.

Reads

makes

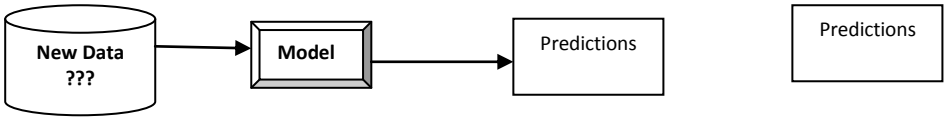

Fig. 3.Use the model to make predictions on new data.

IV. PREDICTIVE MODELLING USING LARGE DATASETS

The above model can work well for small datasets and shall consume less time. Models can be created for large datasets where data has more than 100,000 observations using Python or R. This can be solution to create a benchmark on which we need to improve.

To identify with the strategic areas, initially split down the process of predictive analysis into its essential components. Roughly, it can be divided into 4 parts. Every component demands x amount of time to execute. Let's evaluate these aspects n (with time taken):

1. Descriptive analysis on the Data – 50% time
2. Data treatment (Missing value and outlier fixing) – 40% time
3. Data Modelling – 4% time
4. Estimation of performance – 6% time

These percentages are based on a sample of 40 observations out of 100. At this instant we know where we need to reduce down time. Let's go step by step into the process (with time estimate):

1. **Descriptive Analysis**: In analytics we build models based on Logistic Regression and Decision Trees. Most of the algorithmsuse greedy algorithms, which can subset the number of features we need to focus on. By means of advanced machine learning tools, time taken to perform this task can be significantly reduced. For initial analysis, there is no need to do any kind of feature engineering. Hence, the time we need to do descriptive analysis is restricted to know missing values and big features which are directly visible. In this methodology, we may need 2 minutes to complete this step (assuming a data with 100,000 observations).
2. **Data Treatment:** Since, this is considered to be the most time consuming step, we need to find smart techniques to fill in this phase. Here are two simple tricks which we can implement :

- **Create dummy flags for missing value(s):** Generally missing values in a variable also sometimes carry a good amount of information. For example, if we are analyzing the click stream data, we may not have a lot of values in specific variables corresponding to mobile usage.
- **Assign the missing value with mean or any other easiest method**: It is found that 'mean' works just fine for the first iteration. Just in cases where there is an obvious trend coming from Descriptive analysis, we probably need a more intelligent method.

With such simple methods of data treatment, we can reduce the time to treat data to 3-4 minutes.

3. **Data Modelling**: Gradient Boosting Algorithm[5] will be extremely effective for 100,000 observation cases. In case of larger data, running a Random Forest may be more useful. This will take maximum amount of time (approximately 4-5 minutes).

While working with boosting algorithms we come across two frequently occurring buzzwords: Bagging and Boosting.

Bagging: It is an approach where you take random samples of data, build learning algorithms and take simple means to find bagging probabilities.

Boosting: Boosting is similar, however the selection of sample is made more intelligently. We subsequently give more and more weight to hard to classify observations.

There are multiple boosting algorithms like Gradient Boosting, XGBoost, AdaBoost, Gentle Boost etc. Every algorithm has its own underlying mathematics and a slight variation is observed while applying them. The accuracy of a predictive model can be boosted in two ways: Either by embracing feature engineering or by applying boosting algorithms straight away.

4. **Estimation of Performance:** A k-fold with k=7 will be highly effective which takes 1-2 minutes to execute and document. The reason to build this model is to establish a benchmark for our self. A few snippets of the code in R are given below :

   Step 1: Append both train and test data set together

   Step 2: Read data-set to your memory

   setwd("C:\\Users\\vvnk\\Desktop\\smec\\AV")

   complete<- read.csv("complete_data.csv", stringsAsFactors = TRUE)

   Step 3: View the column names/summary of the datasetcolnames(complete)[1]"ID" "Gender" "City" "Monthly_Income" "payment type" "consumer type"

   Step 4: Identify the a) Numeric variable b) ID variables c) Factor Variables d) Target variables

   Step 5: Create flags for missing values

   missing_val_var<- function(data, variable,new_var_name) {

   data$new_var_name<- ifelse(is.na(variable),1,0))

   return(data$new_var_name)}

   Step 6: Impute Numeric Missing values

   numeric_impute <- function(data, variable) {

   mean1 <- mean(data$variable)

   data$variable<- ifelse(is.na(data$variable),mean1,data$variable)

   return(new_var_name)}

   Similarly impute categorical variable so that all missing value is coded as a single value say “Null”

   Step 7: Pass the imputed variable into the modelling process

   #Challenge: Try to Integrate a K-fold methodology in this step

   create_model<- function(trainData,target){set.seed(120)myglm <- glm(target ~ ., data=trainData, family = "binomial")

   return(myglm) }

   Step 8: Make predictions

   score <- predict(myglm, newdata = testData, type = "response")

   score_train <- predict(myglm, newdata = complete, type = "response")

   Step 9: Check performance

   auc(complete$Disbursed,score_train)

V. PREDICTIVE MODELLING USING LASSO & RANDOM FOREST

Inpredictive modelling generally two situations [4]arise:

1. Fit a well-defined parameterised model to your data where we require a learning algorithm which can find those parameters on a large data set without over-fitting.

In this case, lasso and elastic-net regularized generalized linear models are a set of modern algorithms which meet all these needs. They are fast, work on huge data sets, and avoid over-fitting automatically. They are available in the "glmnet" package in R.

2. A "black box" which can predict the dependent variable as accurately as possible, where we need a learning algorithm which can automatically identify the structure, interactions, and relationships in the data

In this case, ensembles of decision trees (often known as "Random Forests") have been the most successful general-purpose algorithm in modern times. For instance, most Kaggle competitions have at least one top entry that heavily uses this approach. This algorithm is very simple to understand, and is fast and easy to apply. It is available in the "randomForest" package in R.

Lasso Regression

LASSO stands for *Least Absolute Shrinkage and Selection Operator*. There are two important things here – '*absolute*' and '*selection*'.

Lasso regression performs **L1 regularization**, i.e. it adds a factor of sum of absolute value of coefficients in the optimization objective. Thus, lasso regression optimizes the following:

Objective = RSS + α * (sum of absolute value of coefficients)

Here, α (alpha) works similar to that of ridge and provides a trade-off between balancing RSS (residual sum of squares) and magnitude of coefficients. Like that of ridge, α can take various values. Let's iterate it here briefly:

1. α = 0: Same coefficients as simple linear regression
2. α = ∞: All coefficients zero (same logic as before)
3. 0 < α < ∞: coefficients between 0 and that of simple linear regression

Now, let's run lasso regression. First we'll define a generic function:

```
from sklearn.linear_model import Lasso
def lasso_regression(data, predictors, alpha, models_to_plot={}):
    #fit the model
    lassoreg = lasso(alpha=alpha,normalize=True, max_iter=1e5)
    lassoreg.fit(data[predictors],data['y'])
    y_pred = lassoreg.predict(data[predictors])
        #Check if a plot is to be made for the entered alpha
    if alpha in models_to_plot:
        plt.subplot(models_to_plot[alpha])
        plt.tight_layout()
        plt.plot(data['x'],y_pred)
        plt.plot(data['x'],data['y'],'.')
        plt.title('Plot for alpha: %.3g'%alpha)
        #Return the result in pre-defined format
    rss = sum((y_pred-data['y'])**2)
```

```
    ret = [rss]
    ret.extend([lassoreg.intercept_])
    ret.extend(lassoreg.coef_)
    return ret
```

Here, '*max_iter*' is the maximum number of iterations for which we want the model to run if it doesn't converge before. Set this to a higher than default value. Next, let's check the output for 10 different values of alpha using the following code:

```
#Initialize predictors to all 15 powers of x
predictors= ['x']
predictors.extend(['x_%d'%i for i in range(2,16)])
#Define the alpha values to test
alpha_lasso = [1e-15, 1e-10, 1e-8, 1e-5,1e-4, 1e-3,1e-2, 1, 5, 10]
#Initialize the dataframe to store coefficients
col = ['rss','intercept'] + ['coef_x_%d'%i for i in range(1,16)]
ind = ['alpha_%.2g'%alpha_lasso[i] for i in range(0,10)]
coef_matrix_lasso = pd.DataFrame(index=ind, columns=col)
#Define the models to plot
models_to_plot = {1e-10:231, 1e-5:232,1e-4:233, 1e-3:234, 1e-2:235, 1:236}
#Iterate over the 10 alpha values:
for i in range(10):
    coef_matrix_lasso.iloc[i,] = lasso_regression(data, predictors, alpha_lasso[i], models_to_plot)
```

This gives us the following plots (Fig.4.):

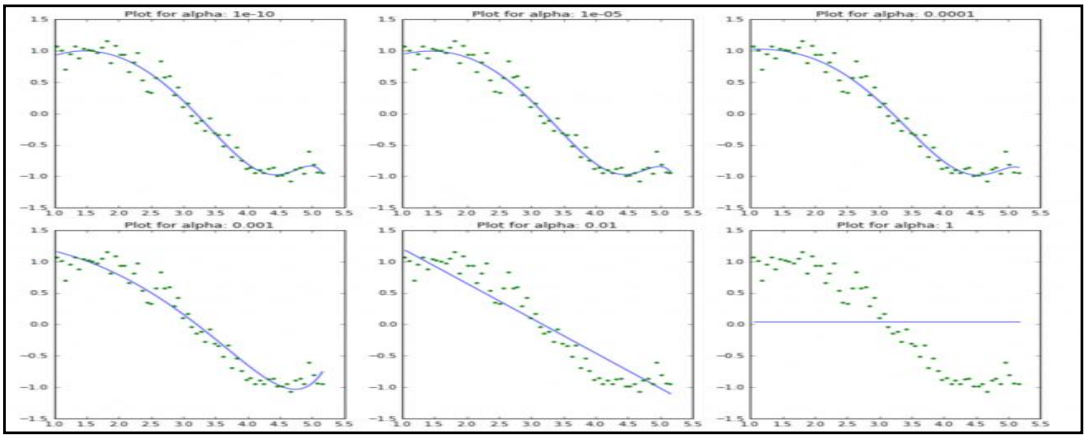

Fig.4. Alpha plots for Lasso Regression

This again tells us that the model complexity decreases with increase in the values of alpha. But notice the straight line at alpha=1. Let's explore this further by looking at the coefficients (Fig.5.):

ISBN: 978-93-86171-32-0

	rss	intercept	coef_x_1	coef_x_2	coef_x_3	coef_x_4	coef_x_5	coef_x_6	coef_x_7	coef_x_8	coef_x_9	coef_x_10	coef_x_11	coe
alpha_1e-15	0.96	0.22	1.1	-0.37	0.00089	0.0016	-0.00012	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.4
alpha_1e-10	0.96	0.22	1.1	-0.37	0.00088	0.0016	-0.00012	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.4
alpha_1e-08	0.96	0.22	1.1	-0.37	0.00077	0.0016	-0.00011	-6.4e-05	-6.3e-06	1.4e-06	7.8e-07	2.1e-07	4e-08	5.3
alpha_1e-05	0.96	0.5	0.6	-0.13	-0.038	-0	0	0	0	7.7e-06	1e-06	7.7e-08	0	0
alpha_0.0001	1	0.9	0.17	-0	-0.048	-0	-0	0	0	9.5e-06	5.1e-07	0	0	0
alpha_0.001	1.7	1.3	-0	-0.13	-0	-0	-0	0	0	0	0	0	1.5e-08	7.5
alpha_0.01	3.6	1.8	-0.55	-0.00056	-0	-0	-0	-0	-0	-0	-0	0	0	0
alpha_1	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
alpha_5	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0
alpha_10	37	0.038	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0	-0

HIGH SPARSITY

Fig.5. Coefficient matrix for Lasso Regression

Output:

```
alpha_1e-15      0
alpha_1e-10      0
alpha_1e-08      0
alpha_1e-05      8
alpha_0.0001    10
alpha_0.001     12
alpha_0.01      13
alpha_1         15
alpha_5         15
alpha_10        15
dtype: int64
```

Fig.6. Output for coefficient which are zero

Apart from the expected inference of higher RSS for higher alphas, we notice the following:

1. For the same values of alpha, the coefficients of lasso regression are much smaller as compared to that of ridge regression (compare row 1 of the 2 tables).
2. For the same alpha, lasso has higher RSS (poorer fit) as compared to ridge regression
3. Many of the coefficients are zero even for very small values of alpha

Inferences #1, 2 might not generalize always but will hold for many cases. The real difference from ridge is coming out in the last inference. Let's check the number of coefficients which are zero in each model using following code:

coef_matrix_lasso.apply(lambda x: sum(x.values==0),axis=1)

It is observed that **even for a small value of alpha, a significant number of coefficients are zero**(Fig.6.). This also explains the horizontal line fit for alpha=1 in the lasso plots;it's just a baseline model! This phenomenon of most of the coefficients being zero is called '**sparsity**'. Although lasso performs feature selection, this level of sparsity is achieved in special cases only.

Random Forest

Random Forest is a versatile machine learning method capable of performing both regression and classification tasks. It also undertakes dimensional reduction methods, treats missing values, outlier values and other essential steps of data exploration, and does a fairly good job. It is a type of ensemble learning method, where a group of weak models combine to form a powerful model.

In Random Forest, we grow multiple trees (Fig.7.) as opposed to a single tree in CART model. To classify a new object based on attributes, each tree gives a classification and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest) and in case of regression, it takes the average of outputs by different trees.

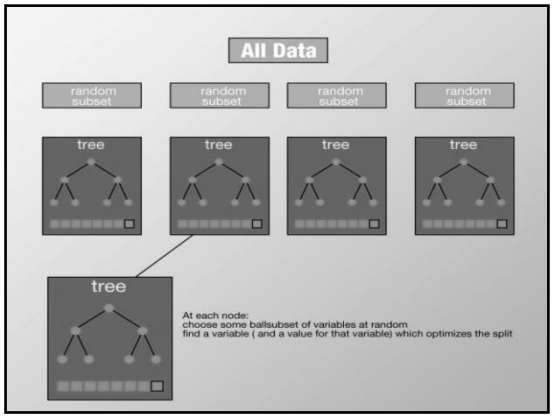

Fig.7. Random Forest

It works in the following manner. Each tree is planted & grown as follows:

1. Assume number of cases in the training set is N. Then, sample of these N cases is taken at random but with replacement. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m<M is specified such that at each node, m variables are selected at random out of the M. The best split on these m is used to split the node. The value of m is held constant while we grow the forest.
3. Each tree is grown to the largest extent possible and there is no pruning.
4. Predict new data by aggregating the predictions of the ntree trees (i.e., majority votes for classification, average for regression).

Python & R implementation

Random forests have commonly known implementations in R packages [6] and Python scikit-learn. Here we have given sample code of loading random forest model in R and Python.

Python

```
#Import Library
from sklearn.ensemble import RandomForestClassifier #use RandomForestRegressor for regression problem
#Assumed you have, X (predictor) and Y (target) for training data set and x_test(predictor) of test_dataset
# Create Random Forest object
model= RandomForestClassifier(n_estimators=1000)
# Train the model using the training sets and check scoremodel.fit(X, y)
#Predict Output
```

```
predicted= model.predict(x_test)
```

R Code

```
> library(randomForest)
> x <- cbind(x_train,y_train)
# Fitting model
> fit <- randomForest(Species ~ ., x,ntree=500)
> summary(fit)
#Predict Output
> predicted= predict(fit,x_test)
```

VI. PREDICTIVE APPS FOR ENHANCING CUSTOMER SATISFACTION

Predictive analytics [7]and customerengagement applications can be combined together to form a new predictive model. Forrester Research coined the term **'Predictive App'**for this combination. These apps use Big Data and predictive analytics. These apps provide the right functionalityand the right content on the right device, at just the right moment for the right person. Using analytical models, predictive apps can analyze the data from multiple sources (from enterprise applications to social networks to third parties) to discover appropriate data, to develop insights and generate actionable information.

A predictive modelcan be based on a Big Data platform and use tools such asR, KNIME, SAS, Predixion, SPSS, Rapid Miner, or Alteryx for advanced analytics. This model treats predictive analyticsfor customer engagement as a cycle with the following phases:

1. **Source:** The model starts by discovering and sourcing data from within the enterprise applications and fromexternal sources.

   The sourcing data includes:

   i) **Behavioural data-**Payment history,Consumption and Savings

   ii) Descriptive data-Characteristics, Geography, Household

   iii) **Interaction data-**Email and chat, Call centre notes and logs, Web clicks & Self Service use, App usage

   iv) **Attitudinal data-**Opinions & Preferences, Interactions with peers

   v) **Social data-**Social Sentiments

   vi) **Third party data-**Weather, Partner Data

2. **Analyze:** It then analyzes the data to develop insights. It analyses Behaviour, Social sentiments, Influence, Price sensitivity, Digital affinity, Product affinity, Customer lifetime value, Interests and preferences, Compliance.
3. **Improve:** The next phase of the model involves improving these insights to generate useful information usingvarious standard modelling techniques. Such information allows the utility to take informed decisions, establishthe context before interacting with customers, create targeted campaigns, and improve its relationship withcustomers. Models include Acquisition model, Campaign response model, Churn model, Basket analysis, Product affinity model, Segmentation model, Sentiments model, Up-selling/cross-selling model, Pricing model

4. **Deliver:** Information is delivered to customers through appropriate communication channels and devices. Thisinformation is also again fed back to the sourcing phase of the model for subsequent analysis. Communication channels and devices include Campaigns and offers, Gamification, Web-based communication such as Social media, Mobile apps, SMS, Chat, Voice and Email.

According to this model, this cycle should continue through the lifetime of the customer, enablingthe utility to generate valuable insights about the customer from both new and old information. Utilities can alsoembrace gamification as a mechanism of delivery to enhance customer engagement and loyalty.

Using such a model, utilities can improve their customer satisfaction scores (such as JD Power scores), therebyhelping them to retain existing customers and acquire new customers. They can also identify ways to improve theirpricing scheme.

Utilities can choose to host their predictive analytics models on the cloud or opt for Business Process as a Service (BPaaS). This will further reduce the total expenditure of the utility, and also provide flexibility and scalability.

VII. COMPETITIVE ADVANTAGE WITH PREDICTIVE ANALYTICS MODEL

A predictive analytics model can improve [8] customer satisfaction by offering contextual andpersonalized customer services, reducing costs, and improving efficiency. This will help improve customerloyalty, which is a distinct competitive advantage in de-regulated markets.

Predictive analytics is not a plug-and-play solution. Even in the case of BPaaS, it is necessary to identify appropriatesources, analyze the correct set of data, use the right model to improve the data, and finally, establish appropriatedelivery channels. It is also important to bring in changes into IT and business processes to support the objectivesof using predictive analytics.

Various factors contribute towards the success of predictive analytics models. These factors can be the availability of customerdata from internal or external sources, as well as integration with digital technologies such as mobility, social media,Big Data, and cloud. Various othersignificant factors are identifying the correct business goals, choosing the righttechnology, and adopting the appropriate roadmap to implement it based on the current landscape and futuredigital strategy.

Utilities usually tend to be risk-averse, watching and learning from other industry players that are subject to newtechnological dynamics. However, with increasing digitization and competition, all utilities need to embracetechnology and business processes that are focused on proactive and contextual customer interaction.

VIII. CONCLUSION

Analytics plays a very important role in consumer relation management. Several statistical techniques and data mining algorithms can be used to develop versatile predictive models. The use of Python and R helps us in coding. Consumer satisfaction can be enhanced by deriving innovative predictive models using data mining thereby gaining a distinct and better competitive advantage in de-regulated markets. Predictive Apps can help in proactive and contextual customer interaction.

ISBN: 978-93-86171-32-0

REFERENCES

[1]. Mike McGuirk , "Customer Segmentation and Predictive Modelling - It's not an either / or decision", white paper, iknowntion, analytic vision, business impact,September 2007,

[2]. Tavish Srivastava , "Perfect way to build a Predictive Model in less than 10 minutes", September 2015, www.analyticsvidhya.com

[3]. Eric Siegel, "Predictive Analytics with Data Mining: How It Works", DM Review's DM Direct, February 2005.

[4]. Jeremy Howard (Enlitic), Mike Bowles (Biomatica), "The Two Most Important Algorithms in Predictive Modelling Today", ,February 2012, www.analyticsvidhya.com

[5]. Buhlmann P, Hothorn T (2007). "Boosting Algorithms: Regularization, Prediction and Model ¨Fitting." Statistical Science, 22(4), 477–505.

[6]. Max Kuhn, "Building Predictive Models in R Using thecaret Package", Journal of Statistical Software, November 2008, Volume 28, Issue 5.

[7]. Galit Shmueli, Otto R. Koppius, "Predictive Analytics in Information Systems Research", MIS Quarterly Vol. 35 No. 3 pp. 553-572/September 2011,P 553-572

[8]. Anirban Banerjee, "Using Predictive Analytics to Enhance Customer Acquisition, Satisfaction, and Retention" , May 2016, white paper, TCS.