

**DR. D. Y. PATIL VIDYAPEETH**

**PIMPRI, PUNE – 411 018**

**DR. D. Y. PATIL SCHOOL OF SCIENCE & TECHNOLOGY**

**TATHAWADE, PUNEs**

**A practical Report on**

**…………………………………….**

**Submitted By**

**Name: UZAIR MERAN SHAFIULLAH KHAN**

**Roll No: BTAI-257**

**ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**ACADEMIC YEAR 2022-2023**

**-------------------------------------Introduction to R----------------------------------**

R is a programming language and software environment for statistical computing and graphics. It is widely used for data analysis, statistical modeling, and visualization. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand in the mid-1990s.

R is open source and freely available under the GNU General Public License. This means that anyone can use, modify, and distribute the software. R has a large and active community of users and developers, who contribute to the development of the language and its associated packages.

R is primarily a command-line interface (CLI) language, meaning that it is driven by entering commands into the console. However, there are also several graphical user interfaces (GUIs) available for R, such as RStudio, which provide a more user-friendly interface.

R has a wide range of built-in functions and packages for statistical analysis and visualization, as well as for data manipulation and cleaning. R also has a powerful and flexible graphics system, which allows for the creation of high-quality plots and charts.

Overall, R is a powerful and versatile language for data analysis and visualization, with a large and active community of users and developers.

**---------------------------------------Features of R-----------------------------------**

R has a wide range of features that make it a popular language for statistical computing, data analysis, and visualization. Some of the key features of R include:

Open source: R is free and open-source software under the GNU General Public License, meaning anyone can use, modify, and distribute the software.

Cross-platform compatibility: R can run on various operating systems, including Windows, macOS, and Linux.

Command-line interface: R is primarily a command-line interface (CLI) language, which allows for quick and efficient data manipulation and analysis.

Large and active community: R has a large and active community of users and developers who contribute to the development of the language and its associated packages.

Data analysis and statistical modeling: R has a wide range of built-in functions and packages for statistical analysis and modeling, such as linear and nonlinear modeling, time-series analysis, clustering, and regression analysis.

Data visualization: R has a powerful and flexible graphics system, which allows for the creation of high-quality plots, charts, and graphs.

Data manipulation and cleaning: R provides a range of functions for data manipulation and cleaning, such as merging datasets, subsetting, sorting, filtering, and transforming data.

Reproducibility: R allows for reproducible research, meaning that analyses and visualizations can be easily reproduced and shared with others.

Integration with other languages: R can be integrated with other programming languages, such as Python and SQL, using various packages.

Overall, R is a powerful and versatile language for data analysis, visualization, and manipulation, with a large and active community of users and developers, and it is widely used in various industries, including finance, healthcare, and academia.

**---------------------------------PROBLEM STATEMENT--------------------------**

This data set is related with retail domain and challenge is to predict whether a store should get opened or not based on certain factors such as sales, population,area etc.Use two datasets ,store\_train.csv and store\_test.csv . You need to use data store\_train to build predictive model for response variable ‘store’. store\_test data contains all other factors except ‘store’, you need to predict that using the model that you developed and submit your predicted values in a csv files.

------------------------**FUNCTION USED IN PROJECT-----------**

**Getwd :** getwd() is a built-in R function that is used to get the current working directory (i.e., the directory where R is currently executing). The syntax for getwd()

**read.csv :** read.csv() is a built-in R function that is used to read data from a comma-separated value (CSV) file into a data frame. The syntax for read.csv()

**library(dplyr):** library(dplyr) is a command in R that loads the dplyr package, which is a popular package for data manipulation and transformation. The dplyr package provides a set of functions for selecting, filtering, arranging, and summarizing data in data frames.

**Glimpse :** glimpse() is a function in the dplyr package of R that provides a compact and informative view of a data frame, similar to str() function but more user-friendly. The glimpse() function displays the first few rows of the data frame and shows the data types of each column and the first few values of each column. The syntax for glimpse()

Rbind : rbind() is a built-in R function that is used to combine two or more data frames vertically by rows. The function is short for "row bind". The syntax for rbind()

Str: str() is a built-in R function that is used to inspect the structure of an object. The function returns a compact and informative description of the object, including its data type, dimensions, and contents. The syntax for str()

Table: table() is a built-in R function that is used to create a frequency table of a categorical variable. The function takes a single argument, which is usually a factor or a character vector. The syntax for table()

as.factor: as.factor() is a built-in R function that is used to convert a vector or column of a data frame into a factor. A factor is a special data type in R that represents categorical data. Factors are useful for modeling and visualizing data, as they allow you to group and summarize data by category.

Paste: paste() is a built-in R function that is used to concatenate two or more strings into a single string. The function takes one or more arguments, which can be either strings or variables that contain strings. The syntax for paste()

Gsub:gsub() is a built-in R function that is used to replace all occurrences of a pattern in a string with a new string. The function stands for "global substitution". gsub() is particularly useful for cleaning and formatting text data.

Length:length() is a built-in R function that is used to get the number of elements in a vector, list, or other data structure. The function returns an integer that represents the length of the object.

Unique:unique() is a built-in R function that is used to get the unique elements in a vector or data frame column. The function returns a vector of the unique elements in the input, in the order they first appear.

Select:select() is a function from the dplyr package in R that is used to select columns from a data frame. The function is useful when you want to work with only a subset of the columns in a large data frame.

Lapply:lapply() is a built-in R function that is used to apply a function to each element of a list or vector, and returns a list of the same length as the input.

Sum:sum() is a built-in R function that is used to calculate the sum of the elements in a vector or data frame column. The function returns a numeric value that represents the sum of the elements.

Is.na : is.na() is a built-in R function that is used to identify missing or NA (Not Available) values in a vector or data frame column. The function returns a logical vector of the same length as the input, with TRUE values indicating missing or NA values, and FALSE values indicating non-missing values

Filter:filter() is a function from the dplyr package in R that is used to select rows from a data frame that meet specified conditions. The function is useful when you want to work with only a subset of the rows in a large data frame.

set.seed: set.seed() is a built-in R function that is used to set the random number generator seed to a specific value. This ensures that the same sequence of random numbers will be generated each time you run the code, making your results reproducible.

Randomforest: randomForest is an R package that provides a set of functions for building and analyzing random forests, which are an ensemble learning method used for classification, regression, and other types of predictive modeling.

Randomforest():randomForest() is a function from the randomForest package in R that is used to fit a random forest model to a data set. A random forest is an ensemble learning method that constructs a multitude of decision trees at training time and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. It is a popular machine learning algorithm for both classification and regression problems.

Predict:predict() is a built-in R function that is used to generate predictions from a fitted model. The function can be used with a variety of models, including linear regression, logistic regression, and decision trees.

Caret:caret is a popular R package that provides a unified interface for several machine learning algorithms and tools for data pre-processing, model selection, and performance evaluation. It stands for "Classification And REgression Training" and is widely used in the R community for building predictive models.

Some of the key functions provided by caret include:

* train() - used to train a model on a given dataset using a specified algorithm and tuning parameters.
* predict() - used to generate predictions from a trained model.
* confusionMatrix() - used to compute various classification metrics such as accuracy, sensitivity, and specificity.
* preProcess() - used to pre-process the data prior to model training, including scaling, centering, and imputation of missing values.
* createDataPartition() - used to create a stratified random sample of the data for training and testing purposes.

To use the caret package, you must first install it by running install.packages("caret"), and then load it into your R session using the library() function.

confusionMatrix: confusionMatrix() is a function from the caret package in R that is used to compute various classification metrics, such as accuracy, sensitivity, specificity, and F1 score, based on a set of predicted and actual class labels. The function takes as input the predicted class labels and the true class labels, and returns a confusion matrix and a table of classification metrics.

library(pROC): pROC is an R package that provides tools for computing and visualizing Receiver Operating Characteristic (ROC) curves, which are commonly used in machine learning for evaluating binary classification models. The package also provides functions for computing various performance metrics, such as the area under the ROC curve (AUC), sensitivity, specificity, and positive and negative predictive values.

Some of the key functions provided by pROC include:

* roc() - used to compute an ROC curve for a given set of predicted and actual class labels.
* auc() - used to compute the AUC for a given ROC curve.
* ci() - used to compute confidence intervals for the AUC and other performance metrics.
* coords() - used to extract the coordinates of the ROC curve for plotting.
* plot() - used to create a plot of the ROC curve.

To use the pROC package, you must first install it by running install.packages("pROC"), and then load it into your R session using the library() function.

Auc:auc() - used to compute the AUC for a given ROC curve.

Plot: In R, the plot() function is used to create various types of plots and visualizations.

Importance: In R, importance() is a function that is commonly used in the context of decision trees and random forests to compute the importance of predictor variables for predicting the outcome variable. The importance() function is part of the randomForest package in R, which provides tools for fitting random forest models.

The importance() function returns a matrix that shows the importance scores of each predictor variable. The importance scores are computed based on the decrease in node impurity or the increase in the mean squared error (MSE) when a predictor variable is randomly permuted. The higher the importance score of a variable, the more important it is for predicting the outcome variable.

as.data.frame: In R, as.data.frame() is a function that is used to convert a matrix, list, or other object to a data frame. The resulting data frame will have the same number of rows as the input object, and the columns will be named according to the names of the input object, if applicable.

Rowname: In R, as.data.frame() is a function that is used to convert a matrix, list, or other object to a data frame. The resulting data frame will have the same number of rows as the input object, and the columns will be named according to the names of the input object, if applicable.

Arrange: In R, arrange() is a function from the dplyr package that is used to reorder the rows of a data frame based on one or more variables.

Desc:The desc() function is used to indicate that the variables should be sorted in descending order.

varImpPlot:In R, varImpPlot() is a function from the randomForest package that is used to create a variable importance plot for a random forest model. This plot shows the relative importance of each predictor variable in the model, as measured by the MeanDecreaseGini variable importance measure.

**-----------------------------------R source code --------------------------------------**

**#Let us load data wrangling library dplyr so as to glimpse our data.**

**library(dplyr)**

**# read in data**

**s\_train <- read.csv("store\_train.csv")**

**s\_test <- read.csv("store\_test.csv")**

**#summary of data**

**glimpse(s\_train)**

**glimpse(s\_test)**

**#Data Preparation**

**s\_test$store=NA**

**s\_train$data="train"**

**s\_test$data="test"**

**s=rbind(s\_train,s\_test)**

**glimpse(s)**

**str(s)**

**table(s$country)**

**table(s$State)**

**s$store=as.factor(s$store)**

**glimpse(s)**

**#Next we will convert all categorical variables to dummies.**

**#We will write a function which will take care of that instead of converting them one by one.**

**CreateDummies=function(data,var,freq\_cutoff=0){**

**t=table(data[,var])**

**t=t[t>freq\_cutoff]**

**t=sort(t)**

**categories=names(t)[-1]**

**for( cat in categories){**

**name=paste(var,cat,sep="\_")**

**name=gsub(" ","",name)**

**name=gsub("-","\_",name)**

**name=gsub("\\?","Q",name)**

**name=gsub("<","LT\_",name)**

**name=gsub("\\+","",name)**

**name=gsub("\\/","\_",name)**

**name=gsub(">","GT\_",name)**

**name=gsub("=","EQ\_",name)**

**name=gsub(",","",name)**

**data[,name]=as.numeric(data[,var]==cat)**

**}**

**data[,var]=NULL**

**return(data)**

**}**

**#categorical variables by writing following lines of codes**

**names(s)[sapply(s,function(x) is.character(x))]**

**#length of var**

**length(unique(s$countyname))**

**length(unique(s$storecode))**

**length(unique(s$Areaname))**

**length(unique(s$countytownname))**

**length(unique(s$state\_alpha))**

**length(unique(s$store\_Type))**

**#We will ignore columns or variables like countyname,storecode,Areaname,countytownname for their High-Cardinality.**

**#Further we will ignore data column for obvious reason.**

**s=s %>% select(-countyname,-storecode,-Areaname,-countytownname)**

**#Above codes will discard those four variables & we are left with 14 variables now.**

**#Next Let us make dummies for the rest of columns - state\_alpha & store\_Type.**

**cat\_cols=c("state\_alpha","store\_Type")**

**for(cat in cat\_cols){**

**s=CreateDummies(s,cat,100)**

**}**

**glimpse(s)**

**#Let us see if there is any missing values in our data..**

**lapply(s,function(x) sum(is.na(x)))**

**#From above we can see that We do have missing values in columns like country, population & store.**

**#Next we impute those missing values with the mean of train data as shown below.**

**for(col in names(s)){**

**if(sum(is.na(s[,col]))>0 & !(col %in% c("data","store"))){**

**s[is.na(s[,col]),col]=mean(s[s$data=='train',col],na.rm=T)**

**}**

**}**

**#We can always cross check if those NAs has been replaced with mean or not by using lapply function again.**

**lapply(s,function(x) sum(is.na(x)))**

**#Now we are done with data preparation , lets separate the data next.**

**s\_train=s %>% filter(data=="train") %>% select(-data)**

**s\_test=s %>% filter(data=="test") %>% select(-data,-store)**

**#Next we will break our train data into 2 parts. We will build model on one part & check its performance on the other.**

**set.seed(2)**

**s=sample(1:nrow(s\_train),0.8\*nrow(s\_train))**

**s\_train1=s\_train[s,]**

**s\_train2=s\_train[-s,]**

**#Model Building**

**library(randomForest)**

**#Next we will build our model with 5 variables randomly subsetted at each node i.e mtry & let just say we want to grow 100 such trees.**

**model\_rf=randomForest(store~.-Id,data=s\_train1,mtry=5,ntree=100)**

**model\_rf**

**#Model Validation**

**#Lets see performance of this model on the validation data s\_train2 that we kept aside.**

**val.score=predict(model\_rf,newdata=s\_train2,type='response')**

**#Again we need to check the accuracy using confusionMatrix from caret package.**

**#What we will get is an accuracy of 78% which seems to be a fair model.**

**library(caret)**

**confusionMatrix(val.score,s\_train2$store)**

**#Now let us calculate probability score for our validation data set s\_train2.**

**val.prob\_score=predict(model\_rf,newdata=s\_train2,type='prob')**

**#In order to check the performance of our model let us calculate its auc score.**

**#For that we need to first import a package named ‘pROC’.**

**library(pROC)**

**auc\_score=auc(roc(s\_train2$store,val.prob\_score[,1]))**

**#From above it is clear that the auc score or the tentative score performance of our model is going to be around 0.82**

**plot(roc(s\_train2$store,val.prob\_score[,1]))**

**#Next we will build the random forest model on the entire training data set ‘s\_train’ & predict the same on test data set ‘s\_test’**

**model\_rf\_final=randomForest(store~.-Id,data=s\_train,mtry=5,ntree=100)**

**model\_rf\_final**

**#We will now use this model to predict probability score for test data .**

**test.score=predict(model\_rf\_final,newdata = s\_test,type='prob')[,1]**

**test.score**

**#Variable Importance**

**#We will run below codes to find out the importance of variable.**

**#Higher the mean decrease ginni for any variable better is the variable for prediction. So population is the most important variable.**

**d=importance(model\_rf\_final)**

**d=as.data.frame(d)**

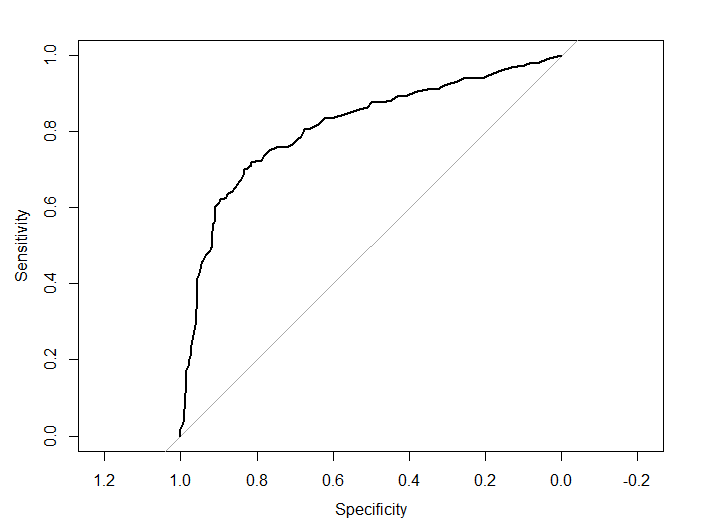
**d$VariableNames=rownames(d)**

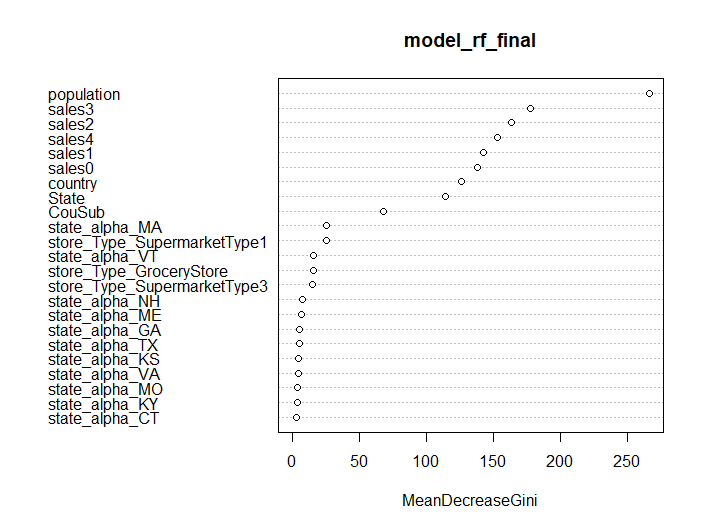
**d %>% arrange(desc(MeanDecreaseGini))**

**#Upon plotting we get a plot like this.**

**varImpPlot(model\_rf\_final)**

---------------------------------------------Output-----------------------------------------





**---------------------------------------Conclusion----------------------------------------**

In this project, we have used R and the linear regression to build a predictive model for store opening. We have trained a linear regression model using the training data and used it to predict the opening of store in the testing data. We have also visualized the variable importance scores of the input features in the model. The results show that the number of population and the location of the store are the most important features in predicting the opening of store. Overall, the linear regression model is an effective tool for predicting store opening and can be used for similar prediction tasks in the future