**Date: 12/07/2019**

Santander Customer

Prediction

Project

Report

**Submitted by Anushka Mishra**

**Contents**

1. **Introduction** 
   1. **Problem Statement**
   2. **Data**
   3. **Exploratory Data Analysis**

1. **Methodology** 
   1. **Pre-Processing** 
      1. **Missing Value Analysis**
      2. **Outlier Analysis**
      3. **Data Understanding**
      4. **Feature Selection**
      5. **Feature Scaling**

* 1. **Model Development** 
     1. **Choosing Model**
     2. **Logistic Regression**
     3. **Decision Tree**
     4. **Random Forest**

1. **Conclusion** 
   1. **Model Evaluation** 
      1. **Accuracy, Precision, FNR & Recall**

* 1. **Model Selection**

**Appendix A – Errors**

**Appendix B – R Script**

**References**

# Chapter 1

## 1. Introduction

### 1.1 Problem Statement

At Santander, mission is to help people and businesses prosper. We are always looking for ways to help our customers understand their financial health and identify which products and services might help them achieve their monetary goals. Our data science team is continually challenging our machine learning algorithms, working with the global data science community to make sure we can more accurately identify new ways to solve our most common challenge, binary classification problems such as: is a customer satisfied? Will a customer buy this product? Can a customer pay this loan?

##### Our objective of this we need to identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted..

### 1.2 Dataset

Image 1

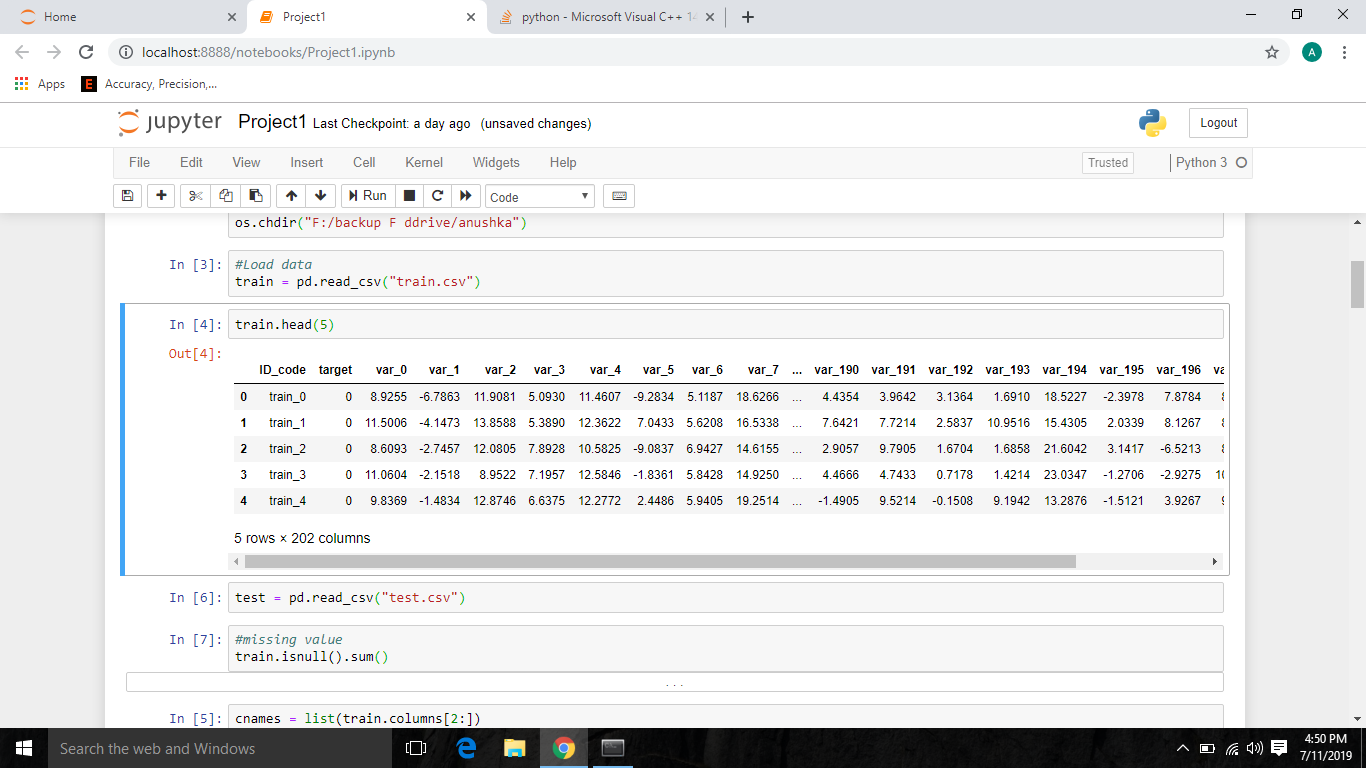
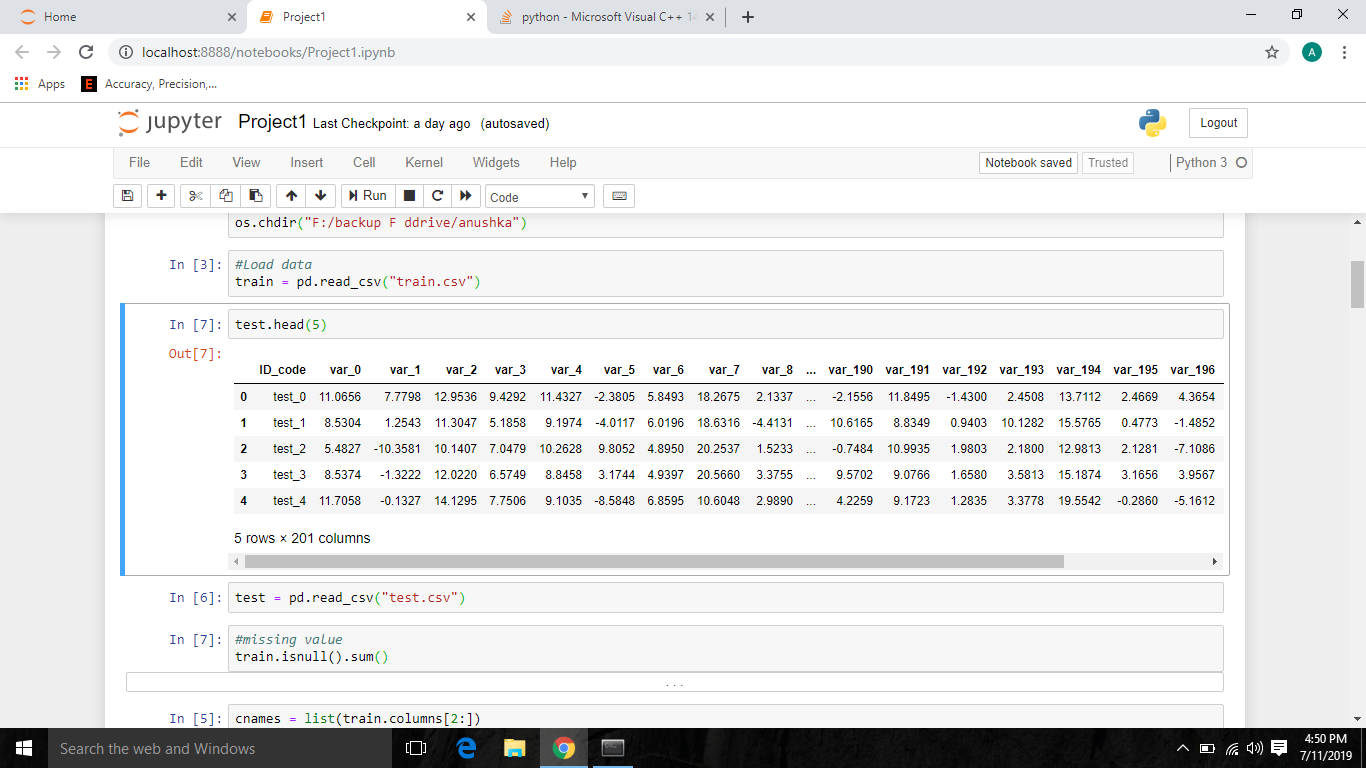


Image 2



In image 1, we can see our train data. It has 200000 observations and 202 features, including target which is our target variable.

In image 2, we can see test data, whose target variable i.e. “target” values to be predicted. It has 200000 observations and 201 features, where, “target” is not available, as we predict those values later in the journey.

As said above, our primary focus is to determine the values for our target variable “target” for future test cases using the above shown dataset in image 1.

### 1.3 Exploratory Data Analysis

In Exploratory Data Analysis, we go through different things, like:

* **Brainstorming** – Here, I have actually draw a rough sketch which talks about – the steps you are going to follow to achieve my given objective.

# Chapter 2

## 2. Methodology

Now, we have the dataset and also, we discussed about Exploratory Data Analysis, let’s talk about the **methodology** we are going to follow to achieve our goal.

We will be going through:

* Pre-processing which includes missing value analysis, outlier analysis, feature selection and feature scaling.
* Model development, where we will choose what machine learning algorithms to apply.

### 2.1 Pre-processing

In pre-processing, we actually apply few techniques like missing value analysis, outlier analysis, feature selection, feature scaling.

Why we do that? Well, actually we never get a structured data to work with. Always messy data is handed to us, and we need to clean that data.

The data may have many observations (rows in dataset), where values in few fields will be absent. We can also say, there may be some inconsistent values in a variable (column in dataset), when compared with other values.

When we go for model development, we should have a structured dataset. We can’t go forward for model development, if we don’t apply pre-processing techniques on data and convert it into structured format.

#### 2.1.1 Missing Value Analysis

Missing value analysis, as the name suggests, we face with situations, where we are given with dataset, and we have missing values in the observations.

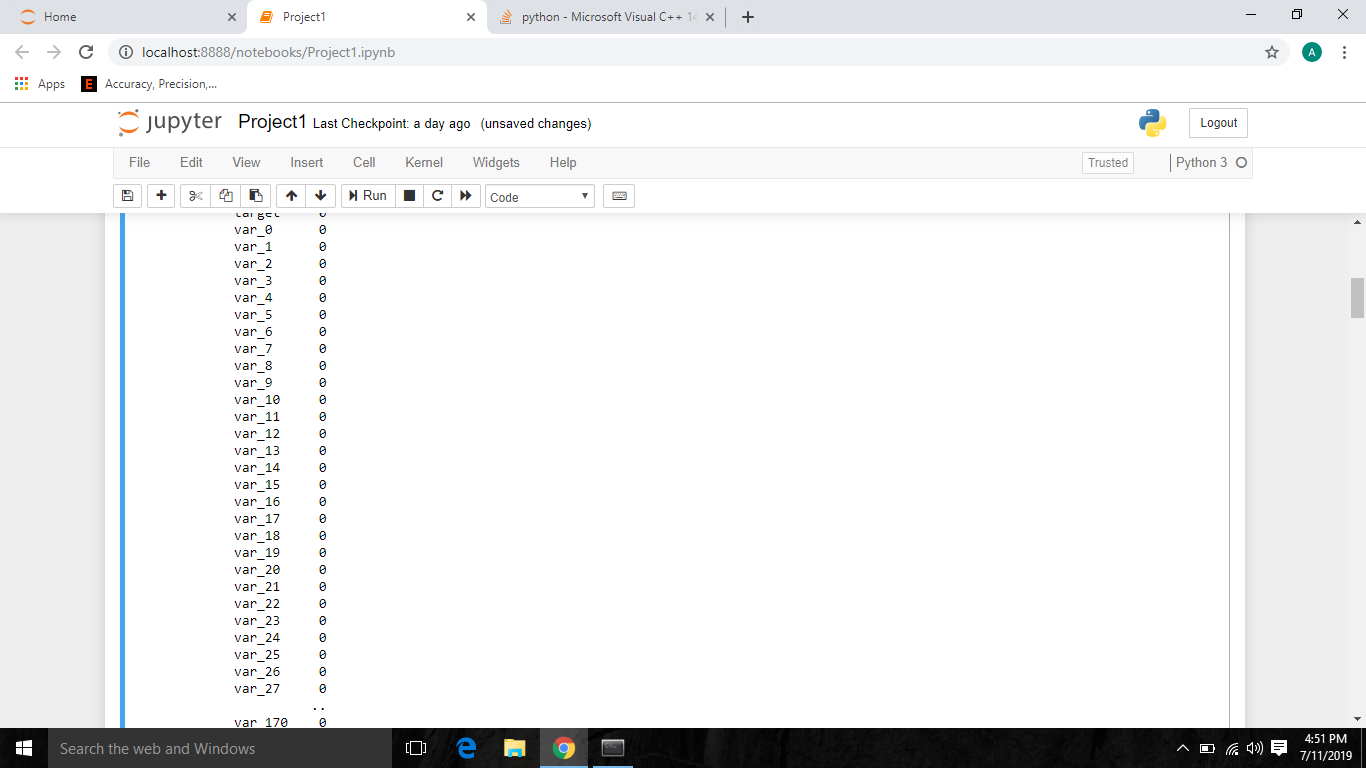
The reason why we have missing values may be plenty, like human error, the user didn’t want to share his complete information, the one who was supposed to fill the data may didn’t work properly.

But, as discussed earlier, we have to give structured data to the model. In order to achieve that, we perform missing value analysis on top of the data to clean the data, to transform the data from unstructured format to structured format.

We give a line of code and it gives us the total number of missing values in each column. Later, we impute those missing values using mean, median or KNN method for numerical variables or using mode for categorical variables. In some cases, we may delete the observations with missing values, only when we have a case where we got few observations with missing values.

After imputing the missing values, we proceed further with outlier analysis. In our project, we had 0 missing values.

Image 3



Another important thing is, we apply missing value techniques only on numerical variables.

#### 2.1.2 Outlier Analysis

Outliers may be defined as the inconsistent values in a variable. For example, a = 1,2,3,4, 20. In object a, 20 is inconsistent, in terms of mean. Another important thing is, we apply outlier analysis only on numerical variables.

Outliers are used for fraud detection. Let’s say, in one bank account, consistently, we observe an amount which ranges from 50,000 to 1,00,000, but in one case 10 lakhs gets deposited. In that case, simply using outlier technique, we can get the inconsistent values.

In our project, we found outliers and we deleted the outliers in these variables as the number of outliers were minimum. We used to describe function to know minimum and maximum.

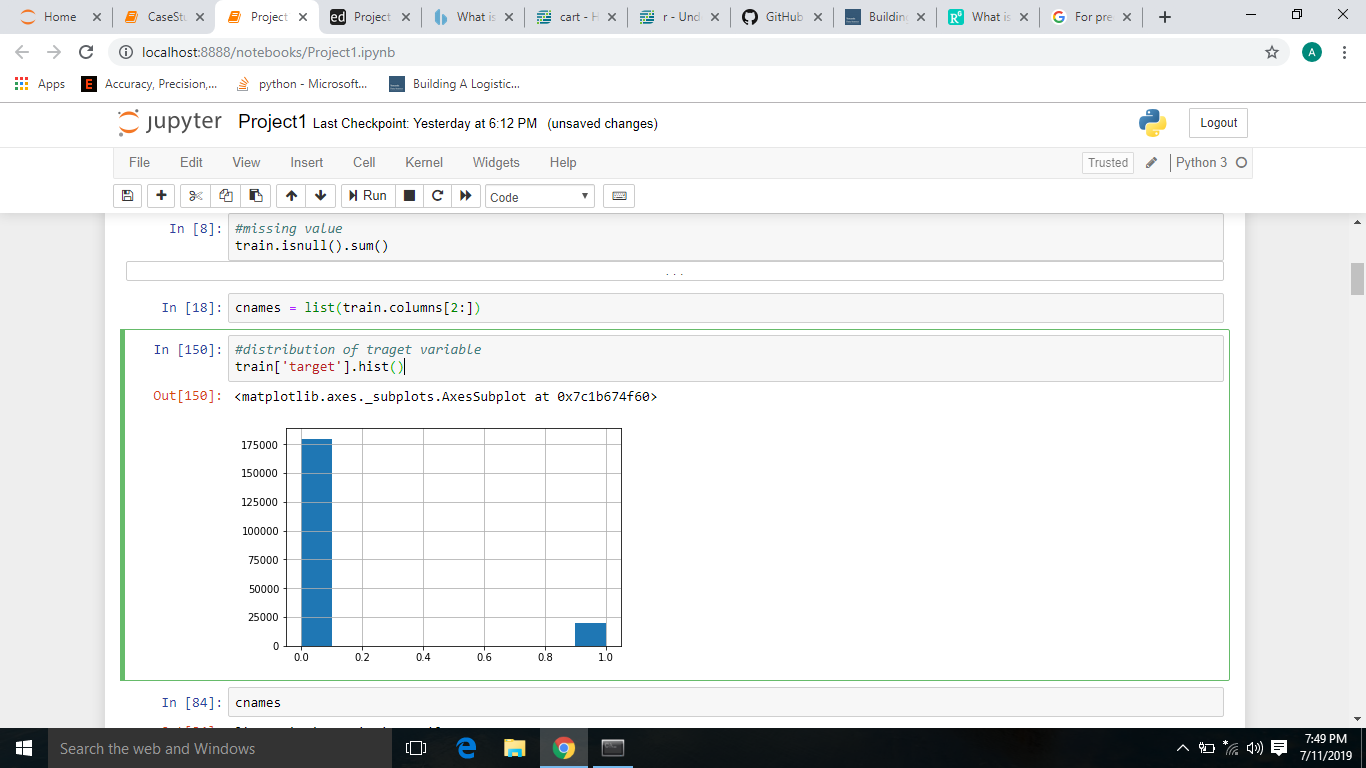
#### 2.1.3 Data Understanding

Understanding visually is easier sometimes and for that purpose we have few libraries in python which allows us to plot awesome visualizations.

We get data, and in order to talk about how few attributes relationship with each other, we can use these visualizations.

You can give two variables, to know how they are related with each other. You can also give three variables to understand the relationship between three variables. You got infinite possibilities to plot. Let’s have a view on few of the plots we used in our project.

Image 4



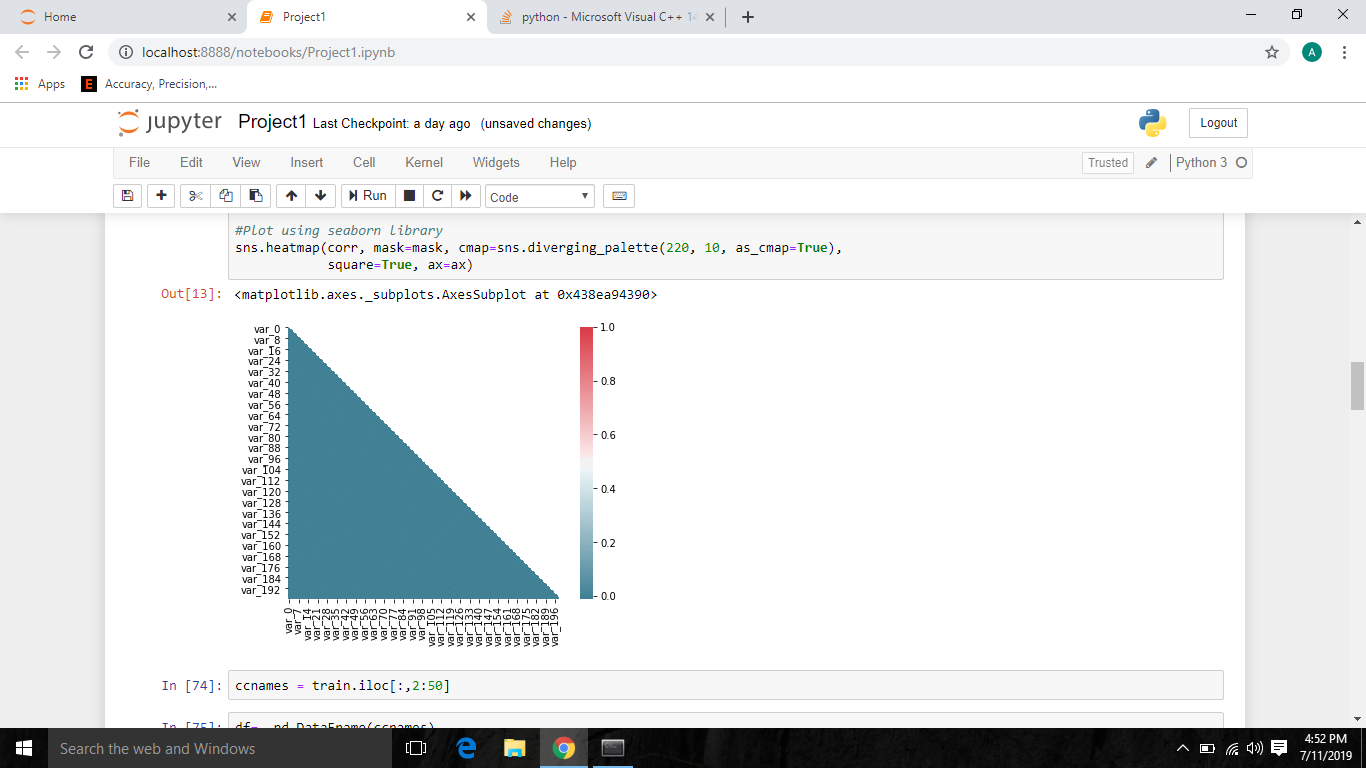
#### 2.1.4 Feature Selection

In Feature Selection, we use few techniques to know which variables are important to us, it’s all about selecting subset of variables from all variables. Actually, when we are given with a dataset, we perform exploratory data analysis, later data pre-processing to clean the data and transform it from unstructured to structured data to feed into model.

However, after this, we may face a situation where we may have variables which have same information with them about the target variable. Let’s talk about an example, in a situation where we are sending five people on a mission. Later, you came to know that, two individuals have the same exact information with them about the mission. Definitely you would drop one, in order to reduce infrastructure and complexity.

The same way, we also drop few variables if they have same information. We always aim that, there should be no independent variable which talks the same as other independent variables but, we appreciate those variables which talks more about the target variable.

In our project, we did correlation analysis between target variable (continuous variable) and other continuous variable. Let’s check it:



Now, we can see, data is not correlated.

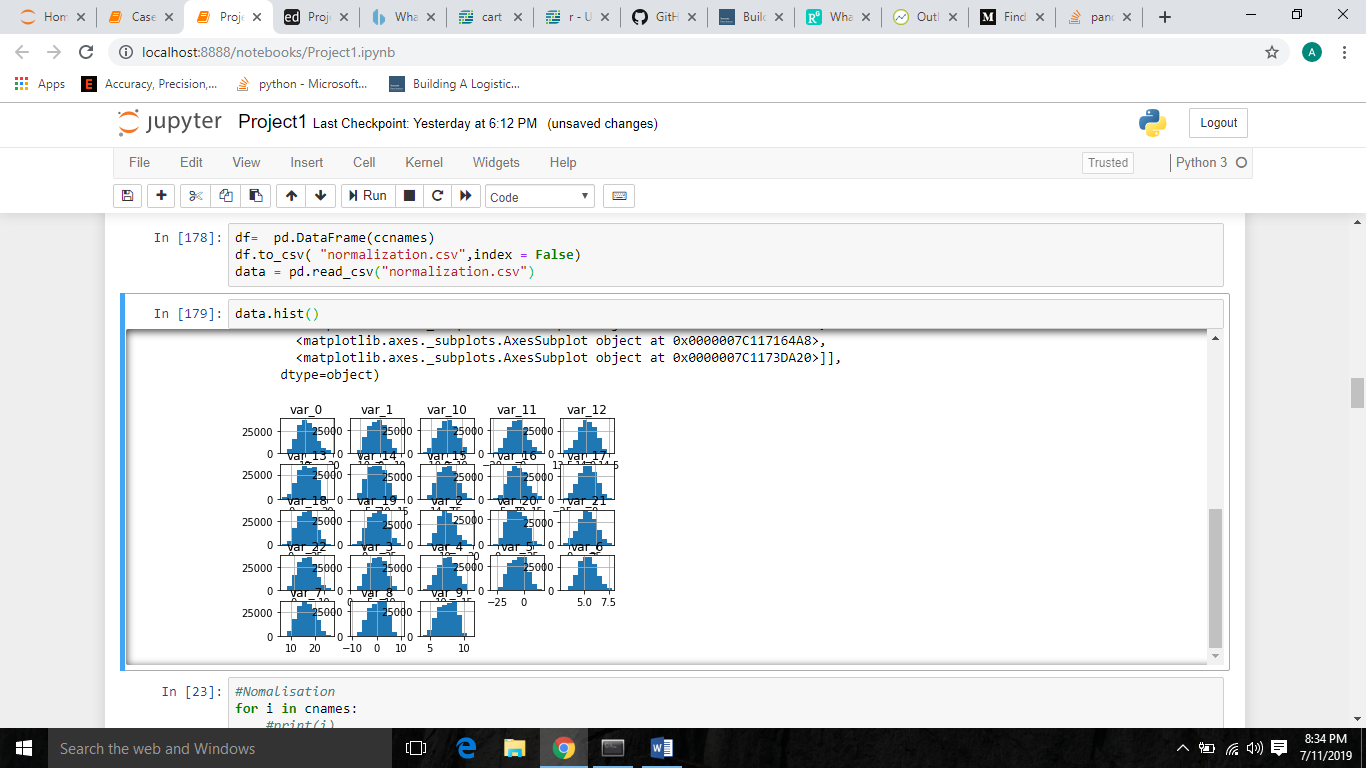
#### 2.1.5 Feature Scaling

In Feature Scaling, we try to limit the ranges of variables, so that they can be compared on the same ground.

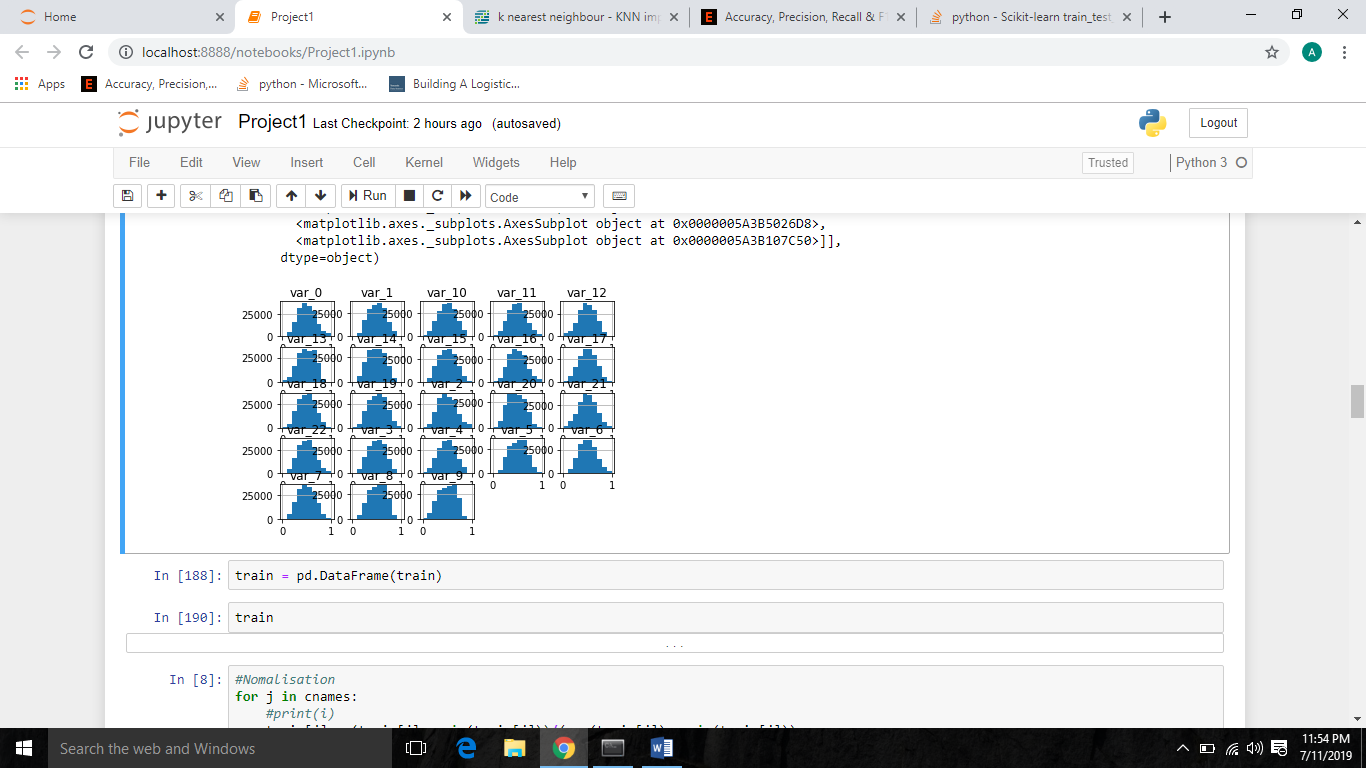
Let’s talk about one example, consider, two variables, age and income. Age is varied from 1 to 100 but coming to income, it ranges in a large scale.

At this situation, the higher values bias the result towards themselves, in order to overcome a situation like this, we use Feature Scaling, where we limit the ranges of variables.

In our project, we did go for feature scaling for distance variable as the given data was left skewed. We also went for distribution check, so let’s have look on that.



Now, as observed, the data points of variables are almost normalized. Now, we did Normalization to bring data in same range.



As we can see now, the distance variable got a pretty desired distribution. Finally, we are done with Feature Scaling. Now, let’s step into Model Development.

### 2.2 Model Development

Model Development, is the phase which comes after we are done with applying the exploratory data analysis, data pre-processing techniques, on the top of data.

The data, will be in structured format, which was our goal, is now ready to develop model.

After we defined our objective and received the data, we transformed it into our required form, we enter into model development, but before that, let’s discuss about model selection.

#### 2.2.1 Model Selection

Model Selection particularly depends on the objective, the problem statement. We have to know at first hand, that, under which category, the problem statement falls.

We have four categories:

* Prediction
* Classification
* Optimization
* Unsupervised Learning

##### Our problem statement is to design a system that predicts the which customers will make a specific transaction in the future, irrespective of the amount of money transacted..

##### 

Our problem statement is a Classification problem (target variable is a classified variable) and it falls under Predication category.

In our project we decided to go with, Logestic Regression, Decision Tree, Naive Bayes .

#### 2.2.2 Logestic Regression

Logestic Regression is a supervised machine learning algorithm where the predicted output is categorical.

In our project, we get #Accuracy = 91.4

#Precision = 67.7

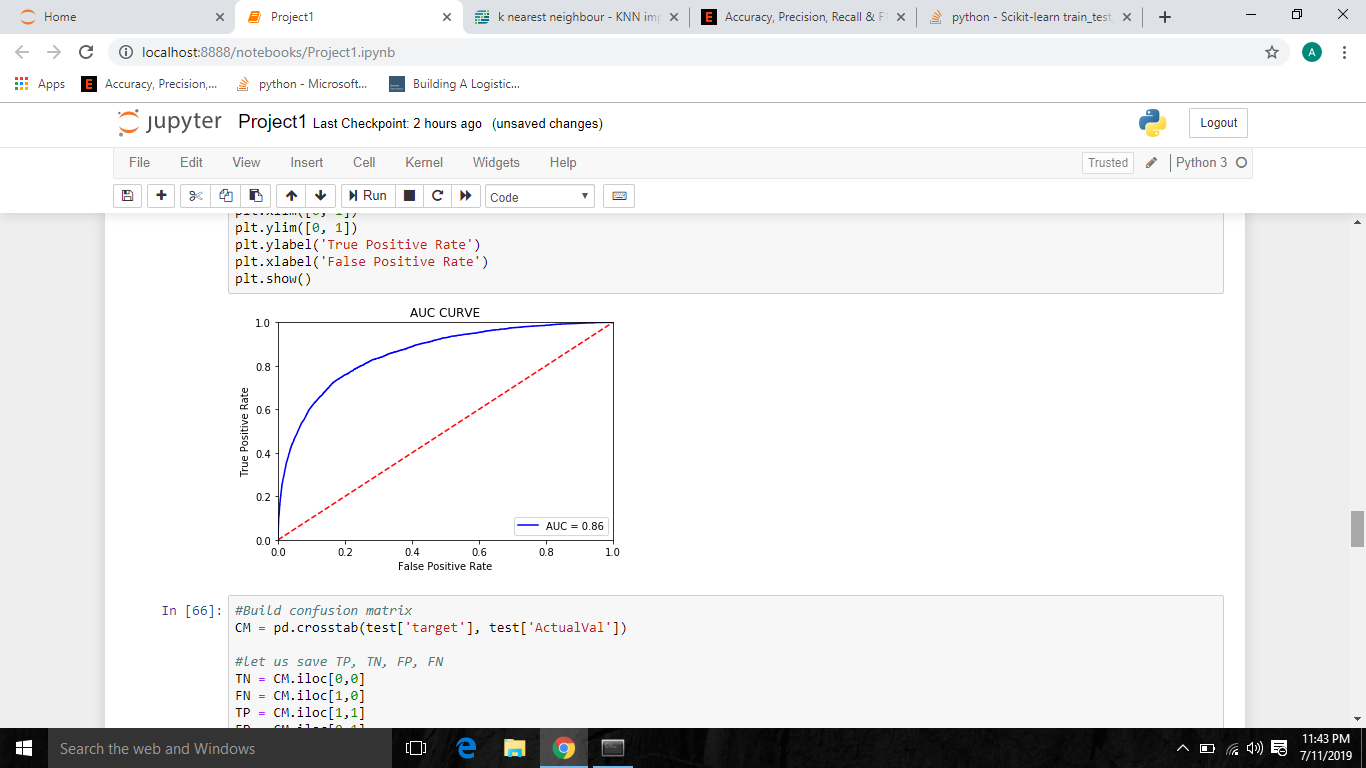
#recall = 26.7

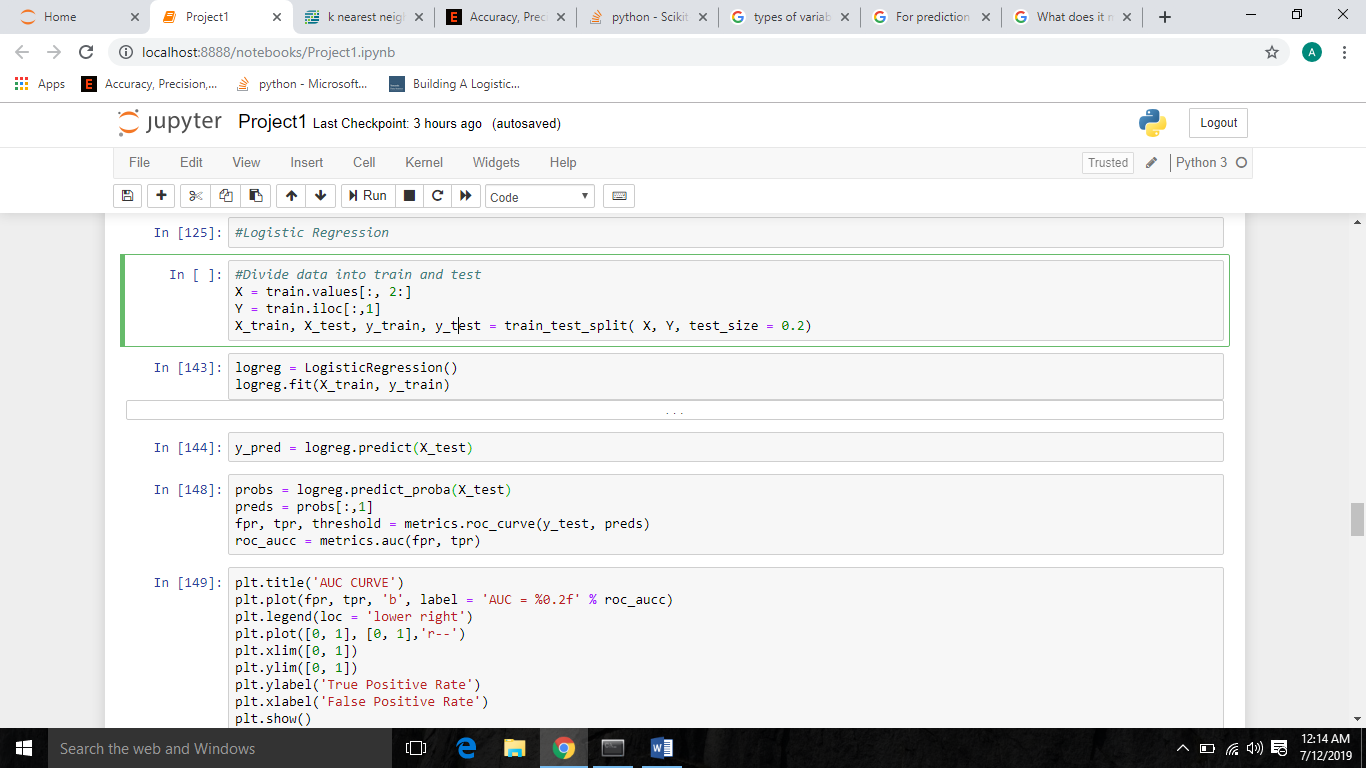
#FNR = 73.9

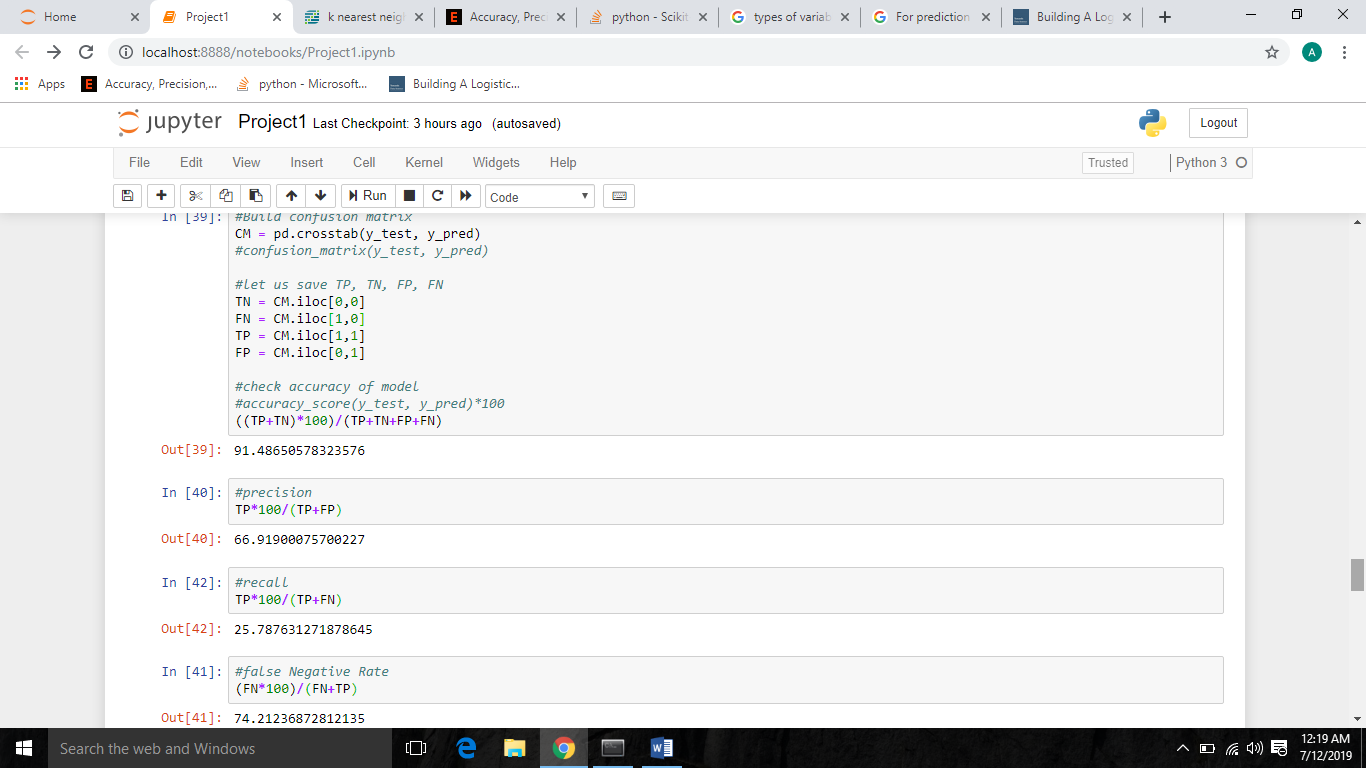
#AUC = 0.86.

We are rejecting this model as FNR is high and recall is low when compared with all other models. Our aim is – we always want a model with low FNR value. AUC of this model is good but not better than others. FNR is Important as client want us to predict the transaction done by his user in future. Hence, having more FNR means that Actual people who can take loan is marked as people who will not take loan.

Image 15







#### 2.2.3 Decision Tree

In our project, we get

#decision tree

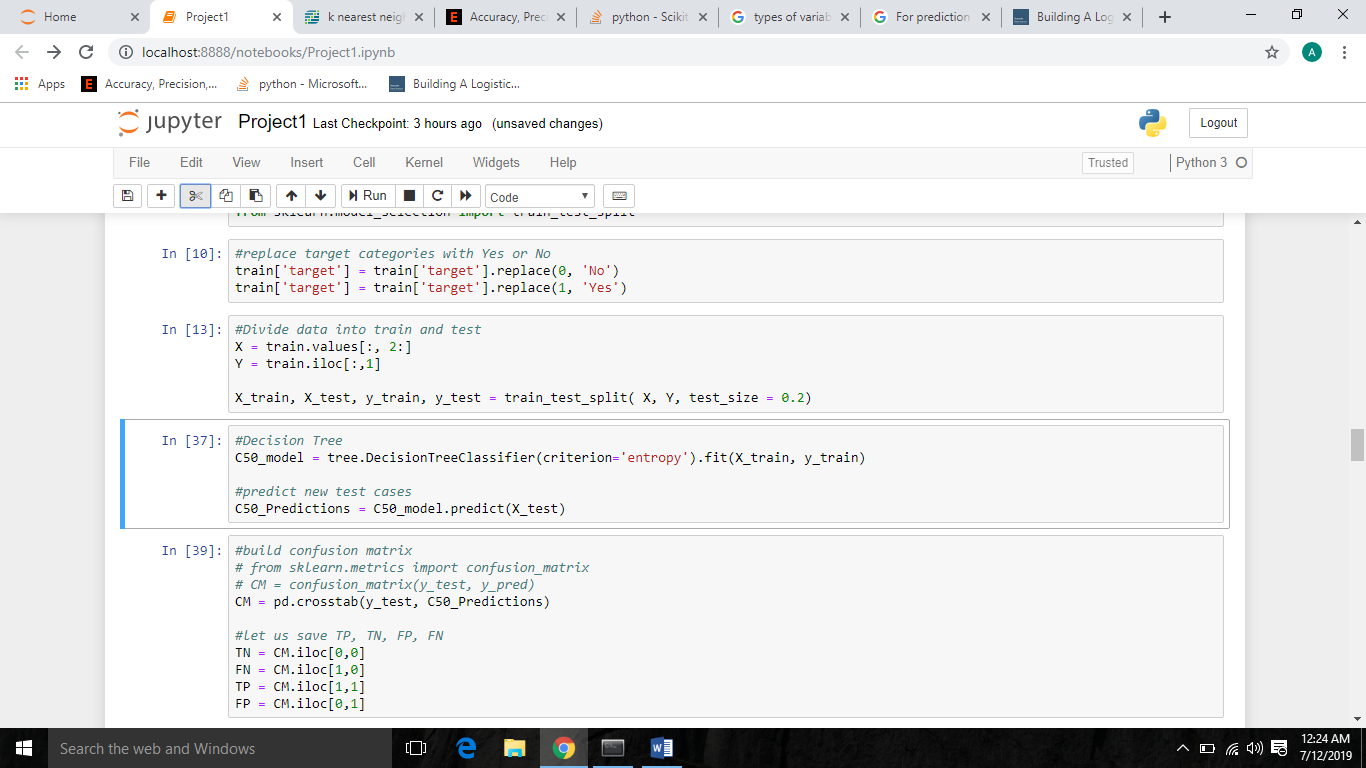
#accuracy = 83.3

#precision = 18.99

#recall = 19.0

#fnr = 80.9.

Decision Tree performed poor than Logistic Regression, we are rejecting this model as FNR is high and Recall is low when compared with Logistic.



#### 2.2.4 Naive\_Bayes

In our project, we get

#Accuracy = 92.2

#Precision = 71.6

#recall = 35.8

#FNR = 63.1

#AUC = 0.89.

Naïve Bayes performed best than other two models

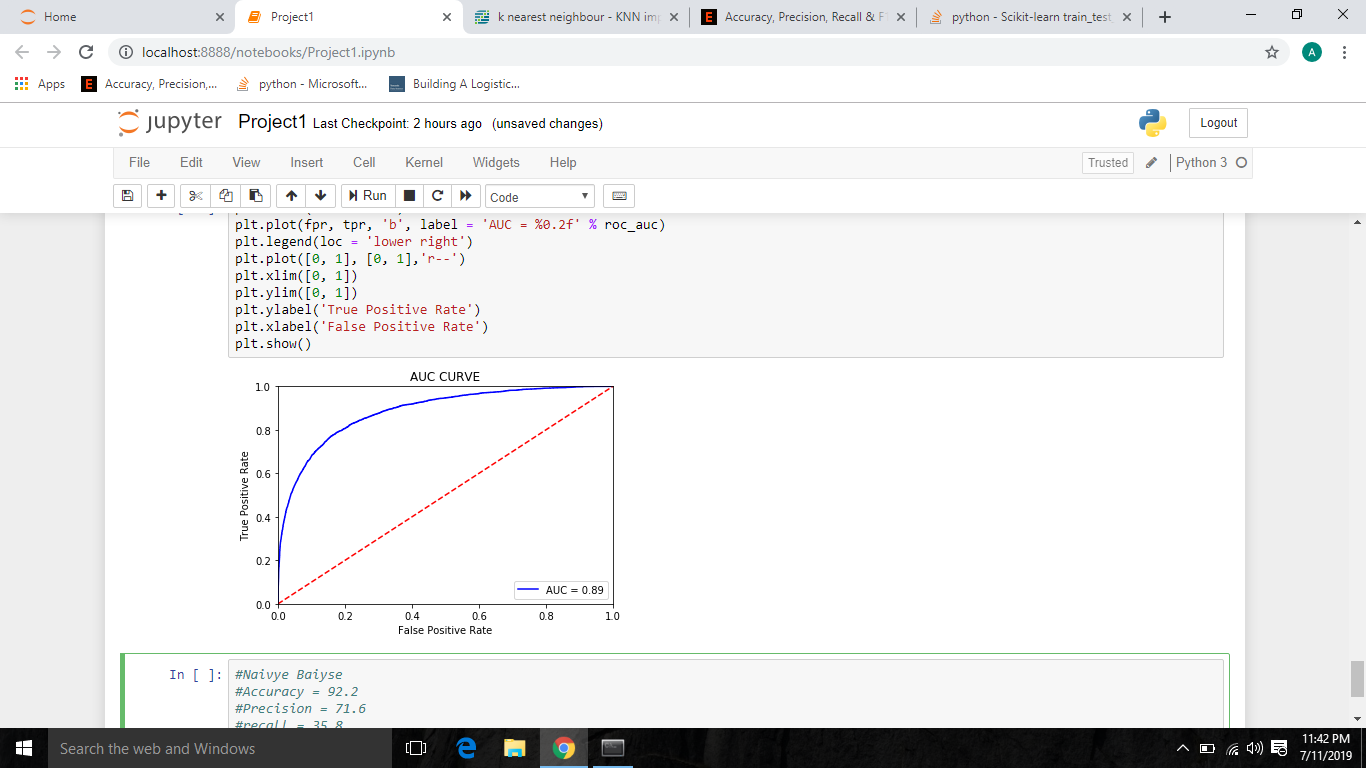
#Accuracy = 92.2

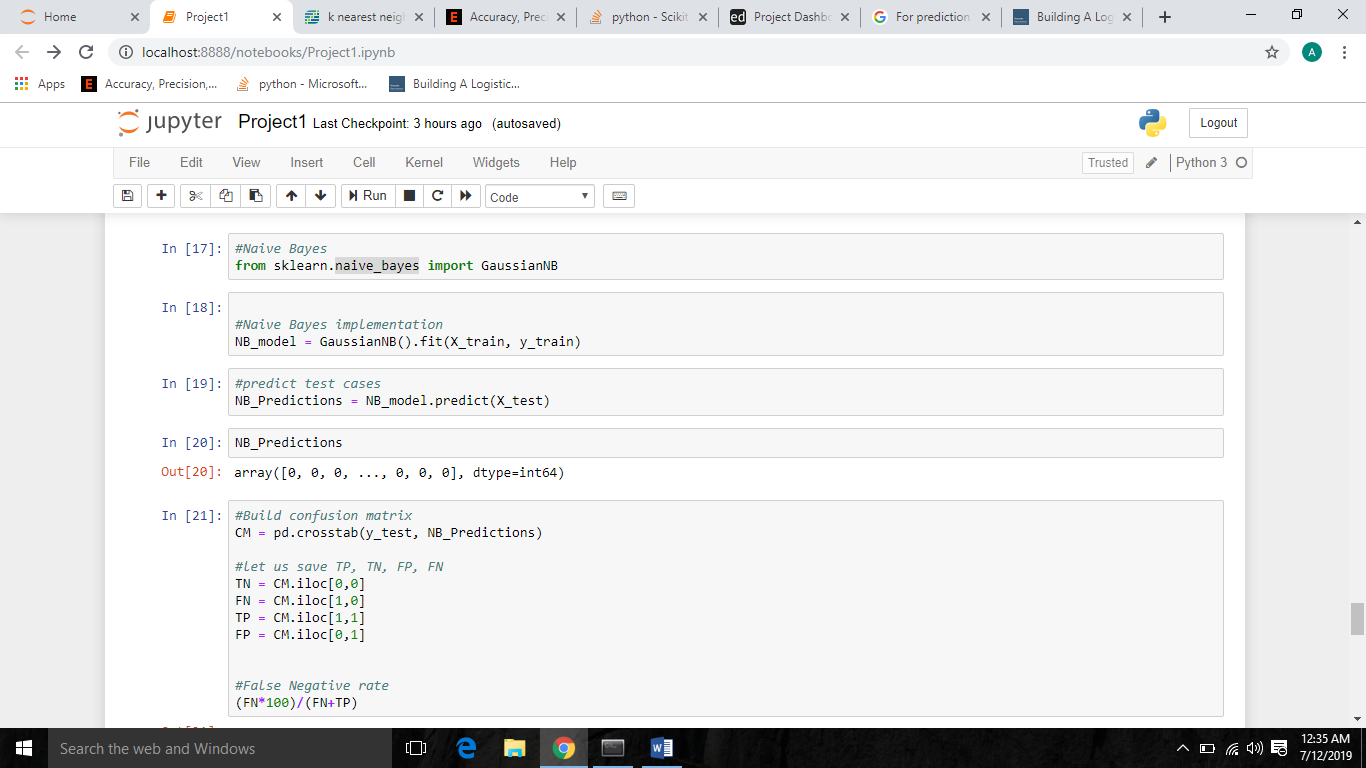
#Precision = 71.6

#recall = 35.8

#FNR = 63.1

#AUC = 0.89





# 

# Chapter 3

## Conclusion

### 3.1 Model Evaluation

We always need a metric to evaluate the work we did. So, the same way, after we developed our models, we need a metric to validate the model we developed.

There are many metrics to evaluate, even, we have different metrics for classification problem and different metrics for regression problems.

For classification problems, we have metrics like:

* Confusion Matrix.
* Accuracy.
* Recall.
* Specificity.

For regression problems, we have metrics like:

* MSE.
* RMSE.
* MAPE.
* Rsquare.

We are choosing Recall and FNR for our project. **Why Recall over Precision?**

Because, **Precision**-**Recall is** a useful measure of success of prediction when the classes **are** very imbalanced. ... A **high** area under the curve represents both **high recall** and **high precision**, where **high precision** relates to a **low** false positive rate, and **high recall** relates to a **low** false negative rate.

#### 3.1.1 False Negative Rate (FNR) & Recall

We are going to use FNR and Recall as our error metrics to evaluate our models.

### 3.2 Model Selection

Finally, it’s our Model selection time. We developed three models. Logestic Regression, Decision Tree and Naïve Bayes.

We are going to freeze Naïve Bayes with FNR as 65 and Recall as 35.

# Chapter 4

## Deployment

## For Python

* In anaconda pip install this

pip install pyinstaller

* After this type

pyinstaller --noconsole script.py

script is project name

* Then you can find your .exe(Window) under dist folder of your working directory.

Run Python script

Copy python installed directory path

Right Click on MY Computer

Click on Advance System settings

Click on Environment Variables

Path-> edit and paste the python code

Run CMD

Write python and load

If it shows you the version than run the Script.py

## For R

# If on R Server 9.0, load mrsdeploy package now

library(mrsdeploy)

# Create glm model with `mtcars` dataset

carsModel <- glm(formula = am ~ hp + wt, data = mtcars, family = binomial)

# Produce a prediction function that can use the model

manualTransmission <- function(hp, wt) {

newdata <- data.frame(hp = hp, wt = wt)

predict(carsModel, newdata, type = "response")

}

# test function locally by printing results

print(manualTransmission(120, 2.8)) # 0.6418125

##########################################################

# Log into Server #

##########################################################

# Use `remoteLogin` to authenticate with Server using

# the local admin account. Use session = false so no

# remote R session started

remoteLogin("http://localhost:12800",

username = "admin",

password = "{{YOUR\_PASSWORD}}",

session = FALSE)

##########################################################

# Publish Model as a Service #

##########################################################

# Generate a unique serviceName for demos

# and assign to variable serviceName

serviceName <- paste0("mtService", round(as.numeric(Sys.time()), 0))

# Publish as service using publishService() function from

# mrsdeploy package. Name service "mtService" and provide

# unique version number. Assign service to the variable `api`

api <- publishService(

serviceName,

code = manualTransmission,

model = carsModel,

inputs = list(hp = "numeric", wt = "numeric"),

outputs = list(answer = "numeric"),

v = "v1.0.0"

)

##########################################################

# Consume Service in R #

##########################################################

# Print capabilities that define the service holdings: service

# name, version, descriptions, inputs, outputs, and the

# name of the function to be consumed

print(api$capabilities())

# Consume service by calling function, `manualTransmission`

# contained in this service

result <- api$manualTransmission(120, 2.8)

# Print response output named `answer`

print(result$output("answer")) # 0.6418125

##########################################################

# Get Service-specific Swagger File in R #

##########################################################

# During this authenticated session, download the

# Swagger-based JSON file that defines this service

swagger <- api$swagger()

cat(swagger, file = "swagger.json", append = FALSE)

* # Now share this Swagger-based JSON so others can consume it

# Chapter 5

## **Summarize**

This project can help the business in achieving the strategic goals by predicting the percentage of customers will make a specific transaction in the future, irrespective of the amount of money transacted.

Will help the client to predict about his customers and increase his business.

## Appendix A – R Script

# Setting the working directory

setwd("F:/Users/anushka /python ")

# checking the working directory getwd()

# Loading the data into our R environment

train = read.csv("train.csv")

#~~~~~~~~~~~~~~~~Let's interact with our data and perform Exploratory Data

Analysis~~~~~~~~~~~~~~~~~~~~~~~~~~

class(train) # Its DataFrame

head(train\_cab) # Let's have a look on first 6 observations dim(train)

# 200000 observations & 202 variables

str(train) # Have a look on the structure

summary(train\_cab) # let us see all the summary

# As observed, we have to change target from numeric to factor train$target = as.factor(train $target)

#~~~~~~~~~~~~~~~~Here comes Missing Values~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

# NO MISSING VALUE IN DATA

# Line of code to know the sum of missing values in dataset sum(is.na(train)) # Total number of missing values are 0

#~~~~~~~~~~~~~~~~~~Its time for OUTLIER analysis~~~~~~~~~~~~~~~~~~~~~

# We are going to delete outlier as they are very few as compare to data size

# Usingbox plt

cnames = colnames(train[,3:202])

for(i in cnames){

print(i)

val = train[,i][train[,i] %in% boxplot.stats(train[,i])$out]

print(length(val))

marketing\_train = train[which(!train[,i] %in% val),]

}

#~~~~~~~~~~~~~~~~~~Time for Feature Selection~~~~~~~~~~~~~~~~~

# In Feature Selection, we perform Correlation Analysis

# Correlation Analysis is performed between num\_var (continuous independent variables) & target (continuous target variable)

library(corrgram)

corrgram(train[1:30,3:202], order = F,

upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

#Data is correlated

#~~~~~~~~~~~~~~~~Let's jump to FEATURE SCALING~~~~~~~~~~~~~~~~~~

# Normalization – all values in observation will be in same range

#qqnorm(train$var\_1) – Visualization of single variable

#code for normalization

for(i in cnames){

print(i)

train[,i] = (train[,i] - min(train[,i]))/

(max(train[,i] - min(train[,i])))

}

#~~~~~~~~~~~~~~Now, We can go for MODEL DEVELOPMENT~~~~~~~~~~~~~~~~~

# Let's clean R Environment, as it uses RAM which is limited

library(DataCombine)

rmExcept("train\_cab")

#~~~~~~~~~~~~~~It's time for DECISION TREE~~~~~~~~~~~~~~~~~~~~~~~~~~~

# Code to build Decision Tree

#Divide data into train and test using stratified sampling method

set.seed(12234)

install.packages('caret')

library(caret)

train.index = createDataPartition(train$target, p = .80, list = FALSE)

train = train[ train.index,]

test = train[-train.index,]

##Decision tree for classification

install.packages('C50')

library(C50)

#Develop Model on training data

str(train$target)

train$target = as.factor(train$target)

C50\_model = C5.0(target ~., trn, trials = 1, rules = TRUE)

#Summary of DT model

summary(C50\_model)

#write rules into disk

write(capture.output(summary(C50\_model)), "c50Rules.txt")

#Lets predict for test cases

C50\_Predictions = predict(C50\_model, test[,-2], type = "class")

##Evaluate the performance of classification model

ConfMatrix\_C50 = table(test$target, C50\_Predictions)

confusionMatrix(ConfMatrix\_C50)

#False Negative rate

FNR = FN/FN+TP

#~~~~~~~~~~~~~~Here we go for Logistic Regression~~~~~~~~~~~~~~~~~~~~~~~~

#Logistic Regression

logit\_model = glm(target ~ ., data = trn, family = "binomial")

#summary of the model

summary(logit\_model)

#predict using logistic regression

logit\_Predictions = predict(logit\_model, newdata = test, type = "response")

#convert prob

logit\_Predictions = ifelse(logit\_Predictions > 0.5, 1, 0)

##Evaluate the performance of classification model

ConfMatrix\_RF = table(test$target, logit\_Predictions)

#False Negative rate

FNR = FN/FN+TP

#~~~~~~~~~~~~~~~Naïve Bayes~~~~~~~~~~~~~~~~~~~~~

#naive Bayes

library(e1071)

#Develop model

NB\_model = naiveBayes(target ~ ., data = trn)

#predict on test cases #raw

NB\_Predictions = predict(NB\_model, test[,3:202], type = 'class')

#Look at confusion matrix

Conf\_matrix = table(observed = test[,2], predicted = NB\_Predictions)

confusionMatrix(Conf\_matrix)

#Accuracy: 92.16

#Recall: 0.93

#precision: 0.63

#specivity:0.70

**References**

* Edwisor.com
* Edwisor Community.
* Google