Customer Transaction

Pesala Ravi Kumar

13/10/2019

Contents

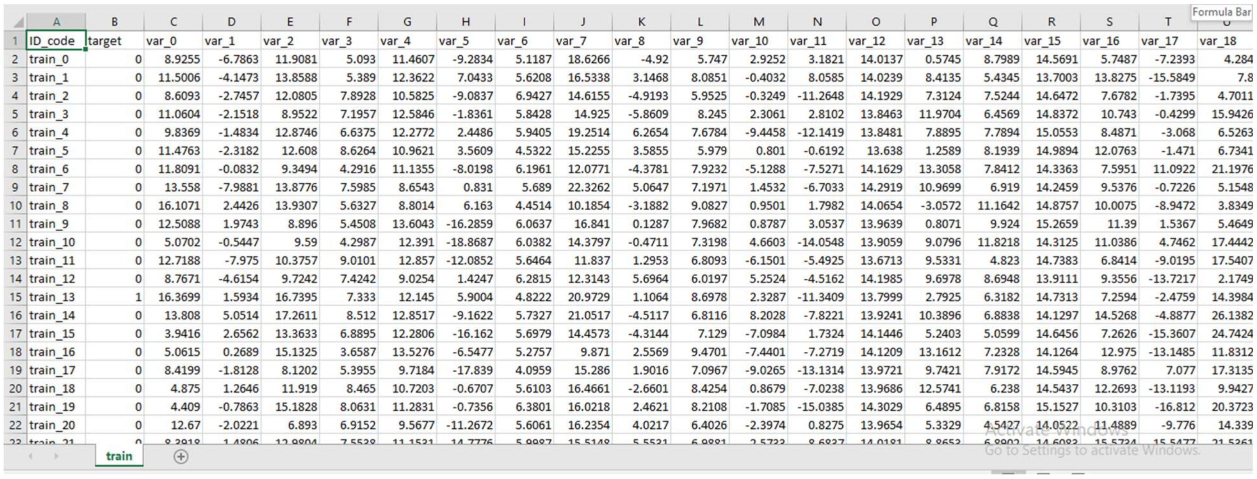
Chapter 1

Introduction

1.1 Problem Statement

We need to identify which customers will make a specific transaction in the future, irrespective of the amount of money transacted. Objective of this project is to predict the customer's future transaction without prior knowledge of customer transaction amount. So this project analyze the past transactions of customer in order to predict the future transaction.

1.2 Data



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	var_8	var_9	var_10	var_11	var_12	var_13	var_14	var_15	var_16	var_17	var_18
2	train_0	0	8.9255	-6.7863	11.9081	5.093	11.4607	-9.2834	5.1187	18.6266	-4.92	5.747	2.9252	3.1821	14.0137	0.5745	8.7989	14.5691	5.7487	-7.2393	4.284
3	train_1	0	11.5006	-4.1473	13.8588	5.389	12.3622	7.0433	5.6208	16.5338	3.1468	8.0851	-0.4032	8.0585	14.0239	8.4135	5.4345	13.7003	13.8275	-15.5849	7.8
4	train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	-4.9193	5.9525	-0.3249	-11.2648	14.1929	7.3124	7.5244	14.6472	7.6782	-1.7395	4.7011
5	train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.925	-5.8609	8.245	2.3061	2.8102	13.8463	11.9704	6.4569	14.8372	10.743	-0.4299	15.9426
6	train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	6.2654	7.6784	-9.4458	-12.1419	13.8481	7.8895	7.7894	15.0553	8.4871	-3.068	6.5263
7	train_5	0	11.4763	-2.3182	12.608	8.6264	10.9621	3.5609	4.5322	15.2255	3.5855	5.979	0.801	-0.6192	13.638	1.2589	8.1939	14.9894	12.0763	-1.471	6.7341
8	train_6	0	11.8091	-0.0832	9.3494	4.2916	11.1355	-8.0198	6.1961	12.0771	-4.3781	7.9232	-5.1288	-7.5271	14.1629	13.3058	7.8412	14.3363	7.5951	11.0922	21.1976
9	train_7	0	13.558	-7.9881	13.8776	7.5985	8.6543	0.831	5.689	22.3262	5.0647	7.1971	1.4532	-6.7033	14.2919	10.9699	6.919	14.2459	9.5376	-0.7226	5.1548
10	train_8	0	16.1071	2.4426	13.9307	5.6327	8.8014	6.163	4.4514	10.1854	-3.1882	9.0827	0.9501	1.7982	14.0654	-3.0572	11.1642	14.8757	10.0075	-8.9472	3.8349
11	train_9	0	12.5088	1.9743	8.896	5.4508	13.6043	-16.2859	6.0637	16.841	0.1287	7.9682	0.8787	3.0537	13.9639	0.8071	9.924	15.2659	11.39	1.5367	5.4649
12	train_10	0	5.0702	-0.5447	9.59	4.2987	12.391	-18.8687	6.0382	14.3797	-0.4711	7.3198	4.6603	-14.0548	13.9059	9.0796	11.8218	14.3125	11.0386	4.7462	17.4442
13	train_11	0	12.7188	-7.975	10.3757	9.0101	12.857	-12.0852	5.6464	11.837	1.2953	6.8093	-6.1501	-5.4925	13.6713	9.5331	4.823	14.7383	6.8414	-9.0195	17.5407
14	train_12	0	8.7671	-4.6154	9.7242	7.4242	9.0254	1.4247	6.2815	12.3143	5.6964	6.0197	5.2524	-4.5162	14.1985	9.6978	8.6948	13.9111	9.3556	-13.7217	2.1749
15	train_13	1	16.3699	1.5934	16.7395	7.333	12.145	5.9004	4.8222	20.9729	1.1064	8.6978	2.3287	-11.3409	13.7999	2.7925	6.3182	14.7313	7.2594	-2.4759	14.3984
16	train_14	0	13.808	5.0514	17.2611	8.512	12.8517	-9.1622	5.7327	21.0517	-4.5117	6.8116	8.2028	-7.8221	13.9241	10.3896	6.8838	14.1297	14.5268	-4.8877	26.1382
17	train_15	0	3.9416	2.6562	13.3633	6.8895	12.2806	-16.162	5.6979	14.4573	-4.3144	7.129	-7.0984	1.7324	14.1446	5.2403	5.0599	14.6456	7.2626	-15.3607	24.7424
18	train_16	0	5.0615	0.2689	15.1325	3.6587	13.5276	-6.5477	5.2757	9.871	2.5569	9.4701	-7.4401	-7.2719	14.1209	13.1612	7.2328	14.1264	12.975	-13.1485	11.8312
19	train_17	0	8.4199	-1.8128	8.1202	5.3955	9.7184	-17.839	4.0959	15.286	1.9016	7.0967	-9.0265	-13.1314	13.9721	9.7421	7.9172	14.5945	8.9762	7.077	17.3135
20	train_18	0	4.875	1.2646	11.919	8.465	10.7203	-0.6707	5.6103	16.4661	-2.6601	8.4254	0.8679	-7.0238	13.9686	12.5741	6.238	14.5437	12.2693	-13.1193	9.9427
21	train_19	0	4.409	-0.7863	15.1828	8.0631	11.2831	-0.7356	6.3801	16.0218	2.4621	8.2108	-1.7085	-15.0385	14.3029	6.4895	6.8158	15.1527	10.3103	-16.812	20.3723
22	train_20	0	12.67	-2.0221	6.893	6.9152	9.5677	-11.2672	5.6061	16.2354	4.0217	6.4026	-2.3974	0.8275	13.9654	5.3329	4.5427	14.0522	11.4889	-9.776	14.339
23	train_21	0	8.3918	1.4806	12.9804	7.5538	11.1531	14.7776	5.9987	15.5148	5.5531	6.9881	2.5733	8.6837	14.0181	8.8653	6.8902	14.6083	15.5724	15.5477	21.5361

As you can see in the fig.1.2.1 above we have the following 200 columns of customer transactions out of which we need to predict the column "target".

Chapter 2

Methodology

2.1 Pre Processing

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis.

2.1.1 Outlier Analysis

Outlier, it is an observation which inconsistence to rest of data.

Causes of the outlier are,

1. Poor data quality or contamination.
2. Low quality measurements.
3. Manual error.
4. Malfunctioning equipment.
5. Correct but exceptional data

Effect of outliers is that it will gives data which is not present in data because of the outlier.

Assume data as 1, 3, 5, 7, and 14 now try to impute using mean value. Mean will be 6 which is not present under data so it might reflect the modelling.

Steps to detect an outlier are,

1. Detect variable with outlier using graphical tools
2. Replace all with NA
3. Apply the missing value analysis on NA records to impute new Values.

2.1.2 Missing Value Analysis

In statistics, missing data, or missing values, occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions that can be drawn from the data.

Why missing the value?

Human Error, refuse to answer, optional

In order to handle Missing Values, there are 2 cases Ignore or Impute Missing value.

Before Imputing Understand why value is missing and by plotting graphs.

Delete the observations where you not to impute.

Techniques to impute missing values:

1. Fill with central statistics like
   i. Mean
   ii. Mode
   iii. Median
2. Distance base/ Data Mining Method – KNN Imputation
   While imputing using KNN method there may be a chance of getting an error: Not sufficient complete cases for computing neighbors.
   In that case we can go for Central Statistics to impute missing values.
3. Prediction method – Machine Learning models.

Selecting the technique for missing values

1. Create a small subset of total data.
2. Delete some values manually.
3. Use multiple methods to fill.
4. See which technique fills correctly.
5. Select that technique for finding missing value analysis.

2.1.3 Feature Selection

It is also called as variable selection or attribute selection.

Selecting a subset of relevant features like variables, predictors for model construction.

Advantages of Feature selection is Dimensionality Reduction (Variable reduction).

Techniques to dimensionality reduction,

1. Correlation analysis.
2. Chi-square test of independence.

2.2 Modeling

2.2.1 Model Selection

You always start your model building from the simplest to more complex so after applying all the models select the best model according to Confusion Matrix, Accuracy and MAPE and etc.

As our data representing target variable is a binary data type variable. So we'll go for classification algorithms in order to detect the target variable data with respect to customer transaction amount.

2.2.2 Decision Tree Classification

Decision Tree builds classification and regression models in the forms of a tree structure. It breaks down dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed.

The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Leaf node represents a classification or decision. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numeric data.

A decision tree can easily transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

So when does Decision Tree classifier terminate?

1. Either it divided into classes that are pure (only containing members of single class).
2. Some criteria of classifier attributes are met.

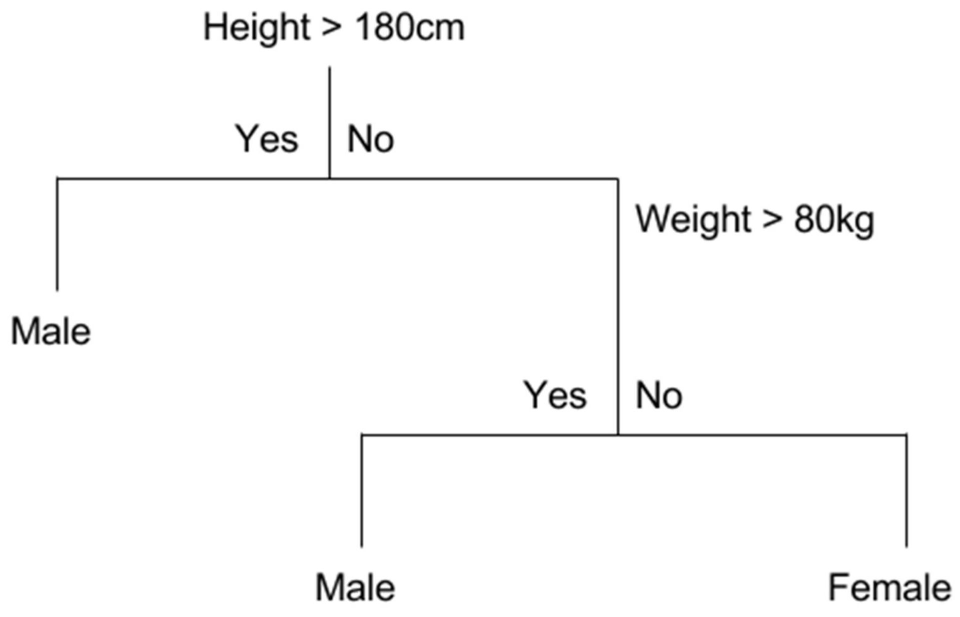
There are two main types of Decision Trees:

1. Classification Trees.
2. Regression Trees.

Classification trees (Yes/No, 1/0 types):

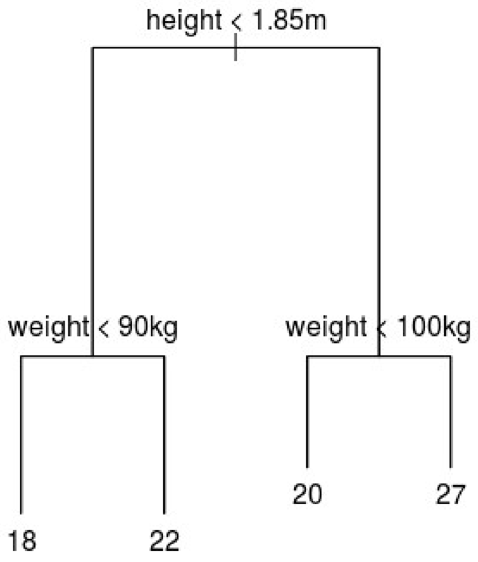
What we've seen above is an example of classification tree, where the outcome was a variable like 'fit' or 'unfit'. Here the decision variable is Categorical/ discrete.

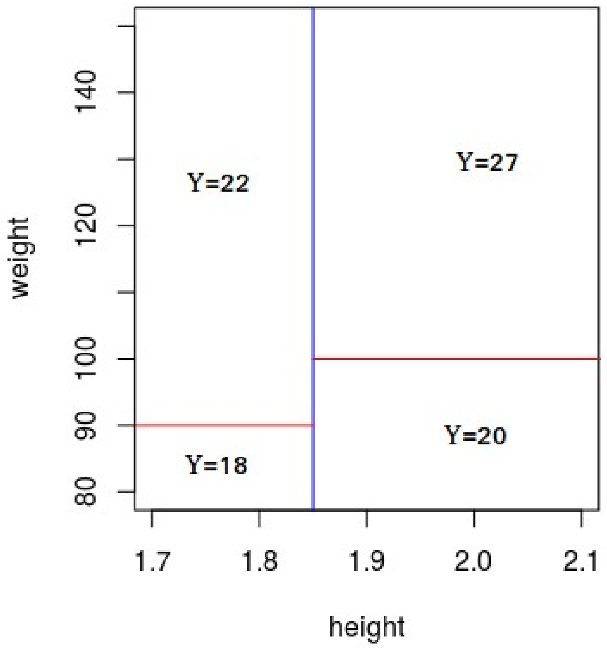
Such a tree is built through a process known as binary recursive partitioning. This is an iterative process of splitting the data into partitions, and then splitting it up further on each of the branches.



Regression trees (Continuous data types):

Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. (E.g. the price of a house, or a patient's length of stay in a hospital)





Creation of Decision Tree:

In this method a set of training examples is broken down into smaller and smaller subsets while at the same time an associated decision tree get incrementally developed. At the end of the learning process, a decision tree covering the training set is returned.

The key idea is to use a decision tree to partition the data space into cluster (or dense) regions and empty (or sparse) regions.

In Decision Tree Classification a new example is classified by submitting it to a series of tests that determine the class label of the example. These tests are organized in a hierarchical structure called a decision tree. Decision Trees follow Divide-and-Conquer Algorithm.

Divide and Conquer

Decision trees are built using a heuristic called recursive partitioning. This approach is also commonly known as divide and conquer because it splits the data into subsets, which are then split repeatedly into even smaller subsets, and so on and so forth until the process stops when the algorithm determines the data within the subsets are sufficiently homogenous, or another stopping criterion has been met.

Basic Divide-and-Conquer Algorithm:

1. Select a test for root node. Create branch for each possible outcome of the test.
2. Split instances into subsets. One for each branch extending from the node.
3. Repeat recursively for each branch, using only instances that reach the branch.
4. Stop recursion for a branch if all its instances have the same class.

Decision Tree Classifier

- Using the decision algorithm, we start at the tree root and split the data on the feature that results in the largest information gain (IG) (reduction in uncertainty towards the final decision).
- In an iterative process, we can then repeat this splitting procedure at each child node until the leaves are pure. This means that the samples at each leaf node all belong to the same class.
- In practice, we may set a limit on the depth of the tree to prevent overfitting. We compromise on purity here somewhat as the final leaves may still have some impurity.

Entropy

Entropy is degree of randomness of elements or in other words it is ***measure of impurity.*** Mathematically, it can be calculated with the help of probability of the items as:

$$H = -\sum p(x) \log p(x)$$

P(x) is probability of item x.

It is negative summation of probability times the log of probability of item x.

Information Gain

Suppose we have multiple features to divide the current working set. What feature should we select for division? Perhaps one that gives us less impurity.

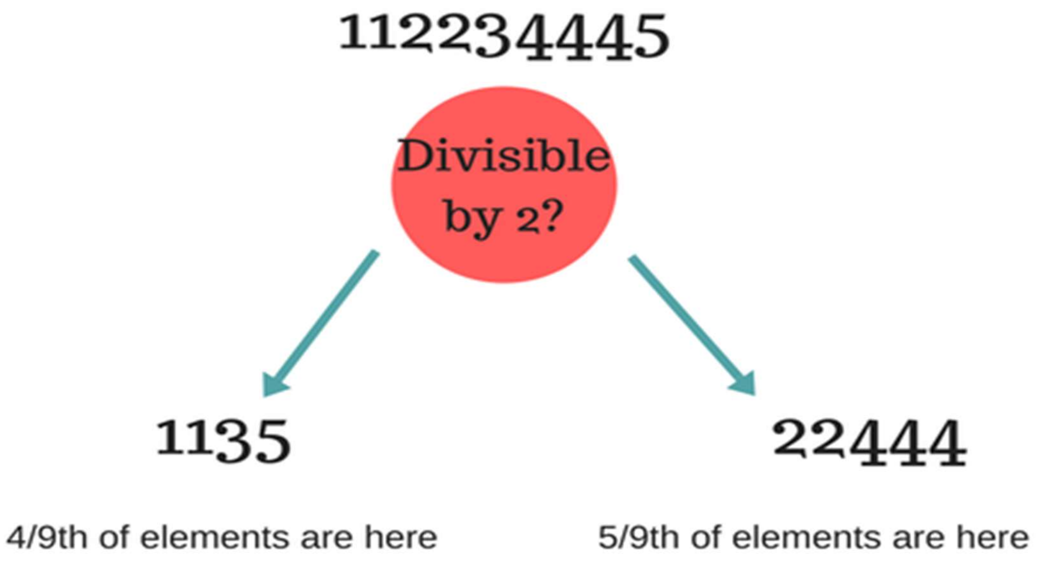
Suppose we divide the classes into multiple branches as follows, the information gain at any node is defined as
Information Gain (n) =
  Entropy(x) — ([weighted average] * entropy (children for feature))

This need a bit explanation!

Suppose we have following class to work with initially

112234445

Suppose we divide them based on property: divisible by 2



Entropy at root level: 0.66
Entropy of left child: 0.45, weighted value = (4/9) * 0.45 = 0.2
Entropy of right child: 0.29, weighted value = (5/9) * 0.29 = 0.16
Information Gain = 0.66 - [0.2 + 0.16] = ***0.3***

Check what information gain we get if we take decision as prime number instead of divide by 2. Which one is better for this case?

Decision tree at every stage selects the one that gives best information gain. ***When information gain is 0 means the feature does not divide the working set at all.***

Dividing efficiently based on maximum information gain is key to decision tree classifier. However, in real world with millions of data dividing into pure class in practically not feasible (it may take longer training time) and so we stop at points in nodes of tree when fulfilled with certain parameters.

Advantages of Classification with Decision Trees:

1. Inexpensive to construct.
2. Extremely fast at classifying unknown records.
3. Easy to interpret for small-sized trees

4. Accuracy comparable to other classification techniques for many simple data sets.
5. Excludes unimportant features.

Disadvantages of Classification with Decision Trees:

1. Easy to over fit.
2. Decision Boundary restricted to being parallel to attribute axes.
3. Decision tree models are often biased toward splits on features having a large number of levels.
4. Small changes in the training data can result in large changes to decision logic.
5. Large trees can be difficult to interpret and the decisions they make may seem counter intuitive.

Applications of Decision trees in real life:

1. Biomedical Engineering (decision trees for identifying features to be used in implantable devices).
2. Financial analysis (Customer Satisfaction with a product or service).
3. Astronomy (classify galaxies).
4. System Control.
5. Manufacturing and Production (Quality control, Semiconductor manufacturing, etc.).
6. Medicines (diagnosis, cardiology, psychiatry).
7. Physics (Particle detection).

Chapter 3

Conclusion

3.1 Model Evaluation

The idea of building machine learning models works on a constructive feedback principle. You build a model, get feedback from metrics, make improvements and continue until you achieve a desirable accuracy. Evaluation metrics explain the performance of a model. An important aspect of evaluation metrics is their capability to discriminate among model results.

In our industry, we consider different kinds of metrics to evaluate our models. The choice of metric completely depends on the type of model and the implementation plan of the model.

When we talk about predictive models, we are talking either about a regression model (continuous output) or a classification model (nominal or binary output). The evaluation metrics used in each of these models are different.

In classification problems, we use two types of algorithms (dependent on the kind of output it creates):

1. Class output: Algorithms like Decision Tree, SVM and KNN create a class output. For instance, in a binary classification problem, the outputs will be either 0 or 1. However, today we have algorithms which can convert these class outputs to probability. But these algorithms are not well accepted by the statistics community.
2. Probability output: Algorithms like Logistic Regression, Random Forest, Gradient Boosting, Adaboost etc. give probability outputs. Converting probability outputs to class output is just a matter of creating a threshold probability.

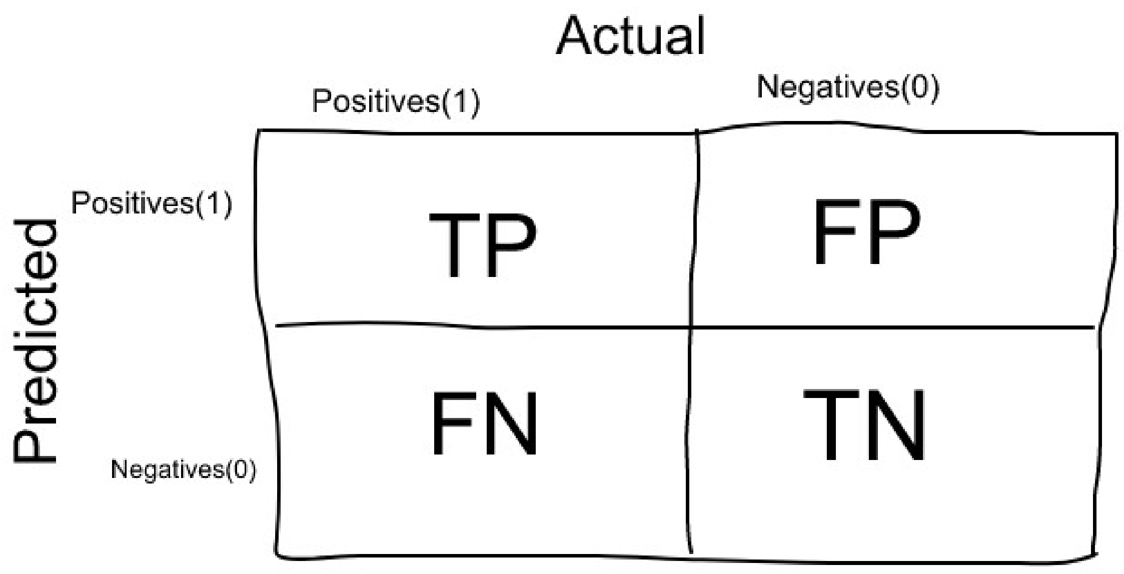
In regression problems, we do not have such inconsistencies in output. The output is always continuous in nature and requires no further treatment.

3.1.1 Confusion Matrix

The Confusion matrix is one of the most intuitive and easiest (unless of course, you are not confused) metrics used for finding the correctness and accuracy of the model. It is used for Classification problem where the output can be of two or more types of classes.

Before diving into what the confusion matrix is all about and what it conveys, Let’s say we are solving a classification problem where we are predicting whether a person is having cancer or not.

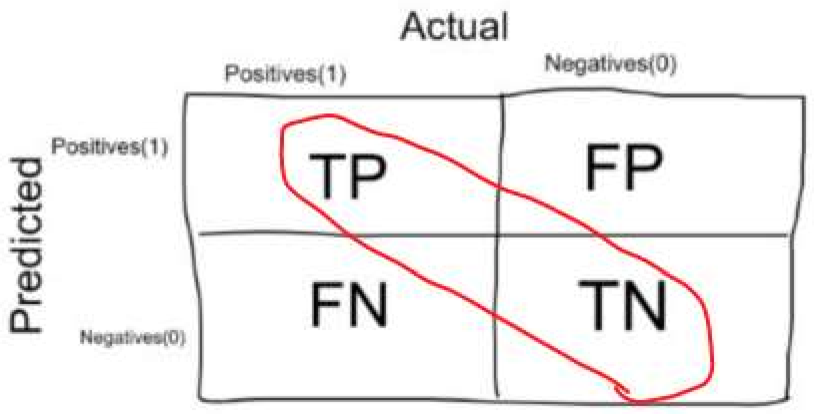
Alright! Now that we have identified the problem, the confusion matrix, is a table with two dimensions (“Actual” and “Predicted”), and sets of “classes” in both dimensions. Our Actual classifications are columns and Predicted ones are Rows.



The Confusion matrix in itself is not a performance measure as such, but almost all of the performance metrics are based on Confusion Matrix and the numbers inside it.

3.1.2 Accuracy

Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made.



$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

In the Numerator, are our correct predictions (True positives and True Negatives)(Marked as red in the fig above) and in the denominator, are the kind of all predictions made by the algorithm(Right as well as wrong ones).

When to use Accuracy:
Accuracy is a good measure when the target variable classes in the data are nearly balanced.

Ex: 60% classes in our fruits images data are apple and 40% are oranges.

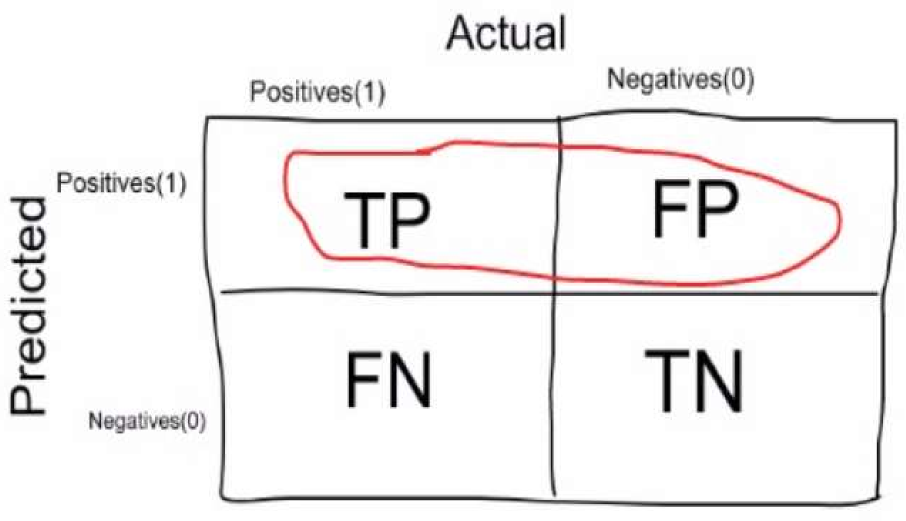
A model which predicts whether a new image is Apple or an Orange, 97% of times correctly is a very good measure in this example.

When NOT to use Accuracy:
Accuracy should NEVER be used as a measure when the target variable classes in the data are a majority of one class.

Ex: In our cancer detection example with 100 people, only 5 people has cancer. Let's say our model is very bad and predicts every case as No Cancer. In doing so, it has classified those 95 non-cancer patients correctly and 5 cancerous patients as Non-cancerous. Now even though the model is terrible at predicting cancer, the accuracy of such a bad model is also 95%.

3.1.2 Precision

Precision is a measure that tells us what proportion of patients that we diagnosed as having cancer, actually had cancer. The predicted positives (People predicted as cancerous are TP and FP) and the people actually having a cancer are TP.

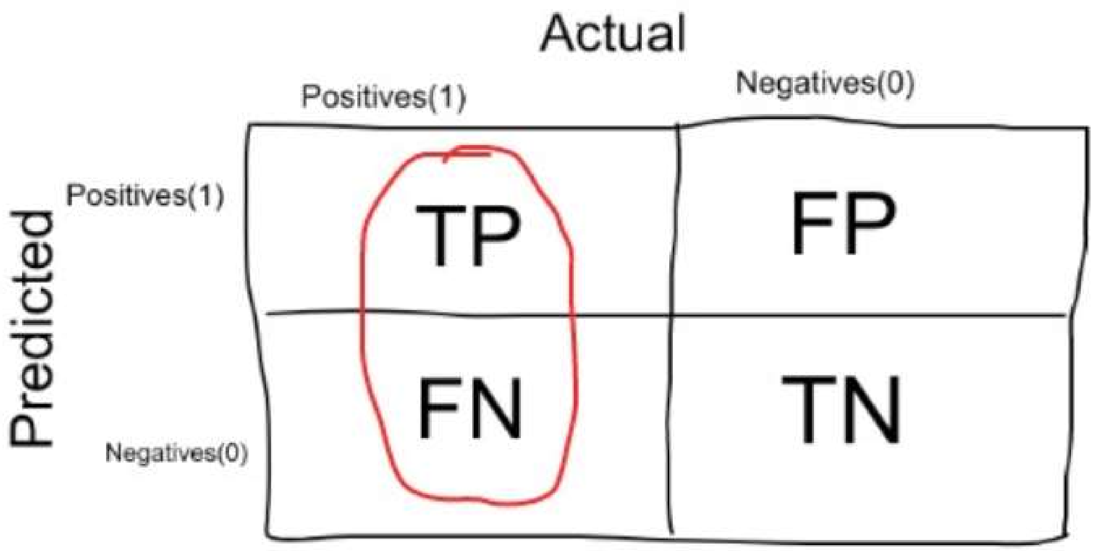


$$\text{Precision} = \frac{TP}{TP + FP}$$

Ex: In our cancer example with 100 people, only 5 people have cancer. Let's say our model is very bad and predicts every case as Cancer*. Since we are predicting everyone as having cancer, our denominator (True* *positives and False Positives) is 100 and the numerator, person having cancer and the model predicting his case as cancer is 5. So in this example, we can say that **Precision** of such model is 5%.*

3.1.2 Recall or Sensitivity

Recall is a measure that tells us what proportion of patients that actually had cancer was diagnosed by the algorithm as having cancer. The actual positives (People having cancer are TP and FN) and the people diagnosed by the model having a cancer are TP. (Note: FN is included because the Person actually had a cancer even though the model predicted otherwise).



$$\text{Recall} = \frac{TP}{TP + FN}$$

Ex: In our cancer example with 100 people, 5 people actually have cancer. Let's say that the model predicts every case as cancer.

*So our denominator (True positives and False Negatives) is 5 and the numerator, person having cancer and the model predicting his case as cancer is also 5(Since we predicted 5 cancer cases correctly). So in this example, we can say that the **Recall** of such model is 100%. And Precision of such a model (As we saw above) is 5%*

When to use Precision and When to use Recall?

It is clear that recall gives us information about a classifier's performance with respect to false negatives (how many did we miss), while precision gives us information about its performance with respect to false positives (how many did we caught).

Precision is about being precise. So even if we managed to capture only one cancer case, and we captured it correctly, then we are 100% precise.

Recall is not so much about capturing cases correctly but more about capturing all cases that have "cancer" with the answer as "cancer". So if we simply always say every case as "cancer", we have 100% recall.

So basically if we want to focus more on minimizing False Negatives, we would want our Recall to be as close to 100% as possible without precision being too bad and if we want to focus on minimizing False positives, then our focus should be to make Precision as close to 100% as possible.

Future Work

Based on this empirical study, the researcher visualized the following areas of further comprehensive research:

1. The relationship between efficiency and perception of customers can also be carried out to best judge the performance of the Customer Transaction.

2. A large sample of customers and employees may be surveyed to have deep perceptions of e-banking service per transaction costs of the banks working across world.

Appendix

R code:

```
rm(list=ls())

install.packages(c("dmm","dplyr","plyr","reshape","ggplot2","data.table","
psych","usdm","caret","DMwR","C50"))


#getwd()

setwd("C:/Users/gopin/Documents/R/CustomerTransaction")

data = read.csv("train.csv", header = T)

savedData = data


# Outlier Analysis

data1 = data[!names(data)%in% c("ID_code")]

for(i in colnames(data1))

{

  #print(i)

  values = data1[,i][data1[,i]%in%boxplot.stats(data1[,i])$out]

  #print(length(values))

  data1[,i][data1[,i]%in%values] = NA

}


#Removing outliers using KNN imputation

require(DMwR)

data1 = knnImputation(data1, k=5)

# Not sufficient complete cases for computing neighbors.


# Removing outliers using MEAN

for(i in colnames(data1))

{

  data1$i[is.na(data1$i)] = mean(data1$i,na.rm = T)
}

# Check if any NA present or not
for(i in colnames(data1))
{
  print(i)
  sum(is.na(data1$i))
}
data1$target[is.na(data1$target)] = 1
newData = data1

# Select Dependent & Independent columns
DependentCols = names(data1)%in% c("target")
dependentData = data1[DependentCols]
independentData = data1[!DependentCols]
independentCols = colnames(independentData)

data = data1[!names(data1)%in% c("ID_code")]

#Check Multicollinearity
install.packages("usdm")
library(usdm)
vif(data[,-201])
vifcor(data[,-201], th=0.9)

# Logistic Regression because of Categorical Target variable
logistic_model = glm(target~. , data = data)
logistic_prediction = predict(logistic_model)
logistic_prediction = ifelse(logistic_prediction > 0.5, 1, 0)

# KNN
# data = data1[!names(data1)%in% c("ID_code")]
trainData_index = sample(1:nrow(data), 0.8 * nrow(data), prob = NULL)
trainData = data[trainData_index,]
testData = data[-trainData_index,]

# library(class)
# knn_pred = knn(trainData[,2:200], testData[,2:200], trainData[,1])

#install.packages("C50")
library(C50)
cols = names(trainData)
trainData[cols] = lapply(trainData[cols] , factor)
str(trainData)

C50_model = C5.0(target~., trainData, trails = 100, rules = TRUE)
write(capture.output(summary(C50_model)), "C50_rules.txt")
C50_prediction = predict(C50_model, testData[,2:201], type="class")

confusionMatrix = table(target, C50_prediction)
ConfusionMatrix(confusionMatrix)
TN = confusionMatrix[0,0]
FN = confusionMatrix[1,0]
TP = confusionMatrix[1,1]
FP = confusionMatrix[0,1]
totalObservations = (TN + FN + TP + FP)

# Accuracy
accuracy = (TP + TN)/totalObservations

# Precision
precision = TP/(TP + FP)
```

```
# Recall
recall = TP/(TP + FN)

# False Negative Rate
fnRate = FN/(FN + TP)

library(ggplot2)
library(scales)
library(psych)
library(gplots)

# ggplot(aes(x='var_0', y='target'), data = data) +
#   geom_point(stat = "identity", fill="Blue") +
#   theme_bw()+
#   ylab("target") + xlab("var_0") +
#   ggtitle("Scatter plot")

ggplot(data, aes(as.factor("var_1"), "target")) +
  geom_point() +
  labs(y = "target", x = "var_1");

ggplot(data, aes(as.factor("var_2"), "target")) +
  geom_point() +
  labs(y = "target", x = "var_2");

ggplot(data, aes(as.factor("var_3"), "target")) +
  geom_point() +
  labs(y = "target", x = "var_3");

ggplot(data, aes(as.factor("var_4"), "target")) +
  geom_point() +
  labs(y = "target", x = "var_4");
```

Python code:

```
import os

os.getcwd()
os.chdir("C:/Users/gopin/Documents/R/CustomerTransaction")

import pandas as pd
import numpy as np
import matplotlib as mlt
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier

data = pd.read_csv("train.csv")

savedData = data

data1 = savedData.iloc[:, 1:202]

# Detect outliers & delete(Casual)
for i in data1.columns[1:202]:
   q75, q25 = np.percentile(data1.loc[:,i],[75,25])
   iqr = q75 - q25
   innerfence = q25 - (float(iqr) * 1.5)
   outerfence = q75 + (float(iqr) * 1.5)
```

```
    data1 = data1.drop(data1[data1.loc[:,i] < innerfence].index)
    data1 = data1.drop(data1[data1.loc[:,i] > outerfence].index)
# Replace with NA
for i in data1.columns[1:202]:
    q75, q25 = np.percentile(data1.loc[:,i],[75,25])
    iqr = q75 - q25
    innerfence = q25 - (iqr * 1.5)
    outerfence = q75 + (iqr * 1.5)
    data1.loc[data1[i] < innerfence,:i] = np.nan
    data1.loc[data1[i] > outerfence,:i] = np.nan

# Impute using Mean method
for i in data1.columns[1:202]:
    data1[i] = data1[i].fillna(data1[i].mean())

# Decision Tree Classification
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split

data1['target'] = data['target'].replace(1,'yes')
data1['target'] = data['target'].replace(0,'no')

x = data1.iloc[:, 1:202]
y = data1.iloc[:, 0]
xTrain, xTest, yTrain, yTest = train_test_split(x, y, test_size=0.2)

clf = tree.DecisionTreeClassifier(criterion = 'entropy').fit(xTrain,yTrain)
```

```
clf_predicted = clf.predict(xTest)


from sklearn.metrics import confusion_matrix

confusionMatrix = confusion_matrix(yTest, clf_predicted)

confusionMatrix = pd.crosstab(yTest, clf_predicted)


TN = confusionMatrix.iloc[0,0]

FN = confusionMatrix.iloc[1,0]

TP = confusionMatrix.iloc[1,1]

FP = confusionMatrix.iloc[0,1]

totalObservations = (TN + FN + TP + FP)


# Accuracy

accuracy = ((TP + TN) * 100)/totalObservations


# Precision

precision = (TP * 100)/(TP + FP)


# Recall

recall = (TP * 100)/(TP + FN)


# False Negative Rate

fnRate = (FN * 100)/(FN + TP)
```