# Value count with percentage

df[‘al’].value\_counts(normalize=True)

give percentage of distribution

df[‘’al”].apply(preprocessor)

call preprocessor method

## Ploting distribution

ls = [ 10, 12, 40, 11, 30, 20, 20, 20, 20, , 11, 10]

plt.hist[ls, bins=30]

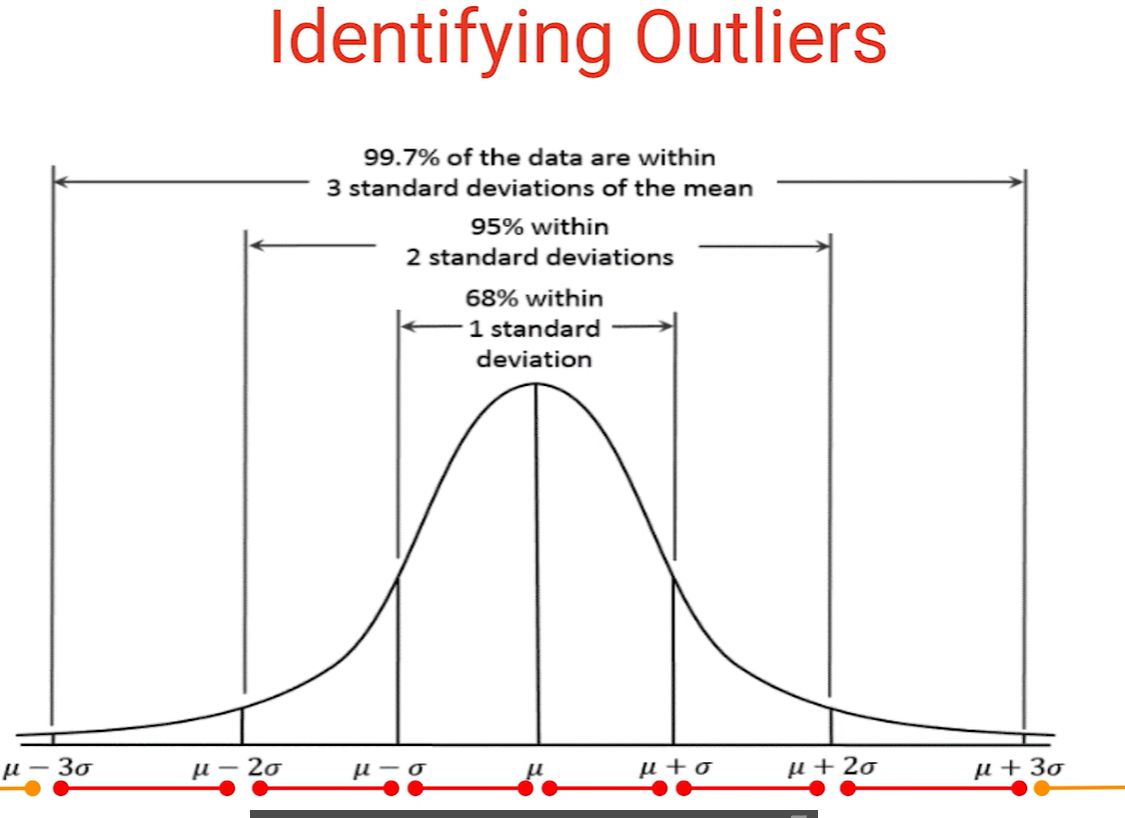
this will draw a distribution and show 10 - 2 times and 20 4 times and so on.

# Univariate missing values

Df.isnull().sum() : give individual count for missing values

## Emperical rules

Any data outside 3 standard deviation is outlier

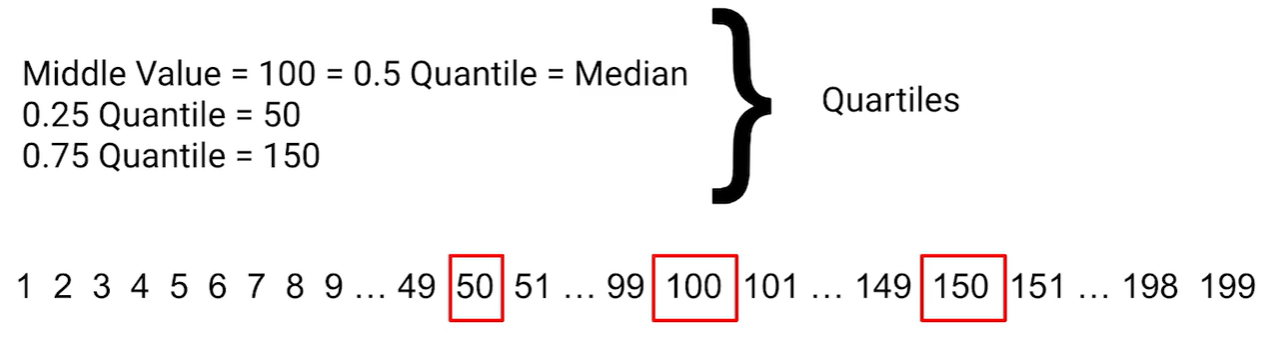


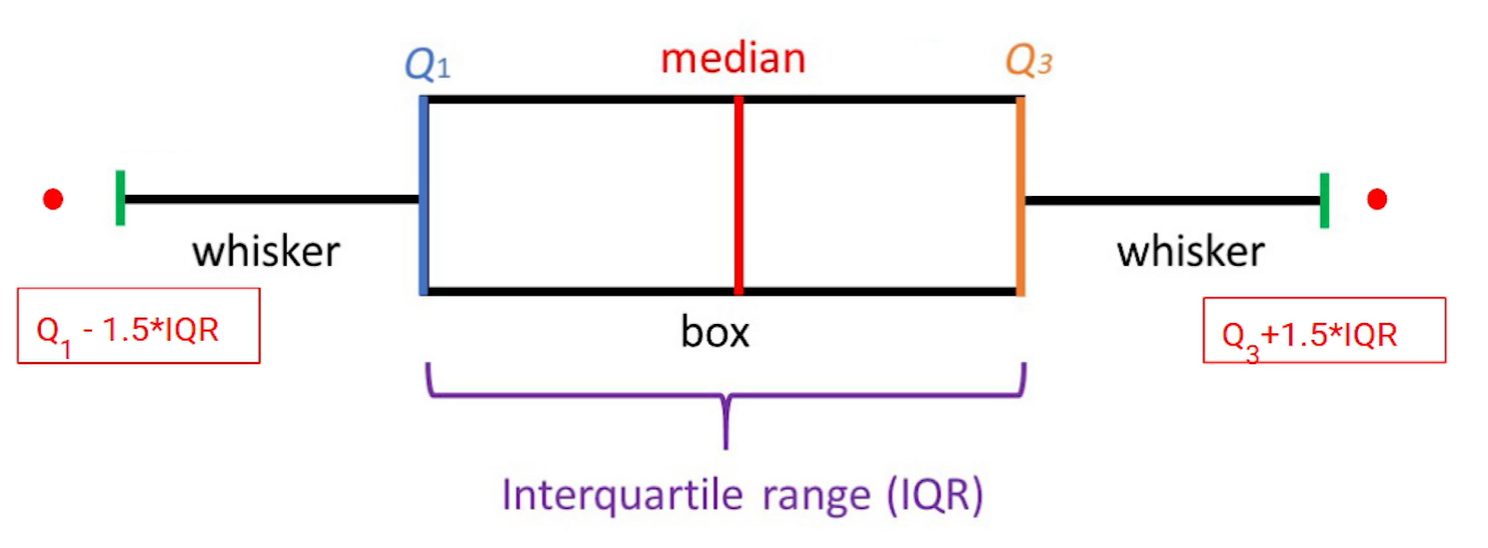
Data with Yellow points are outliers.

Rule only work if data is noraml distributed.but most of the cases data is not normally.

## Quantiles

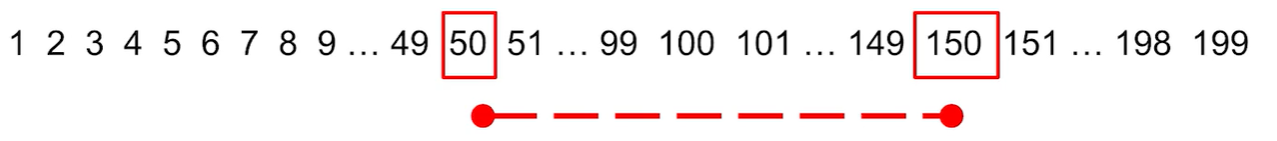
Similar to percanttile





Work with any data set irrespective of its distribution.

## Iterquartile Range vs Range





# Random number creation

import random

ran = [random.randint(1, 2) for i in range(1,20)]

ran

# Panda data frame indexing

1. data.index = random\_list
2. data.set\_index(‘col1’, drop=True, inplace =True)

inplace = true is used to make chanes in the original dataframe

drop = true is used to drop the colums that’s set as index

1. data.reset\_index(inplace=True)

reset data frame to original form whatever experients we performed on index columns it change all

# Subsetting data

## data[data[‘col1’]==’milk’]

return all columns having milk as value

## data[10:15:2]

it will skip every next row and return 10,12,14

## data[-10:0]

return last 10 row

## data.iloc[[1,2,3,4,5,6]]

select specific rows by index number

## data.ilox[[1,4,5,2],[1,3,5]]

select 1,4,5,2 rows and 1,3,5 columns

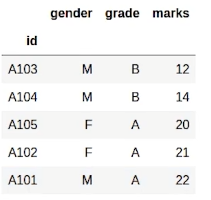
data[] and data.loc[] same

## iloc vs loc

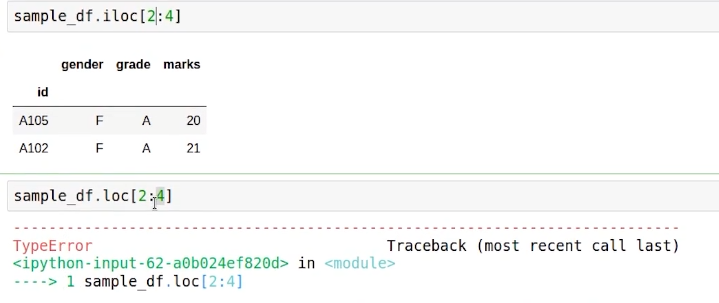
* loc gets rows (and/or columns) with particular **labels**.
* iloc gets rows (and/or columns) at integer **locations**.

iloc is positional index( check underneath position instead of particular row)

loc is label based indexing(check what is value for particular row which we can see)



Iloc work with categorical data as well because it check for underneath position.



Data.loc[‘A104’:’A102’] : this will give right results.

## isin

Data[data.year==2017 and data.year ==2018]

Data[data.year.isin([2017,2018])

Both will give same results.

## Data.dtype

Data.select\_dtypes[‘object’]

Df[] not work properly some time

Df.loc work with label and categorical indexing

## isna

data.isna().sum()

check all the null values

isna() gives true and false

isna only true when value is missing

sum() counts that true and false

## mean

replace all missing item with mean value.

data.loc[data.item.isna()==True), ‘item’] = data.item.mean()

## value\_counts

data.item.value\_counts()

number of values in category

## filllna

data.item.fillna(‘Medium’ , inplace=True)

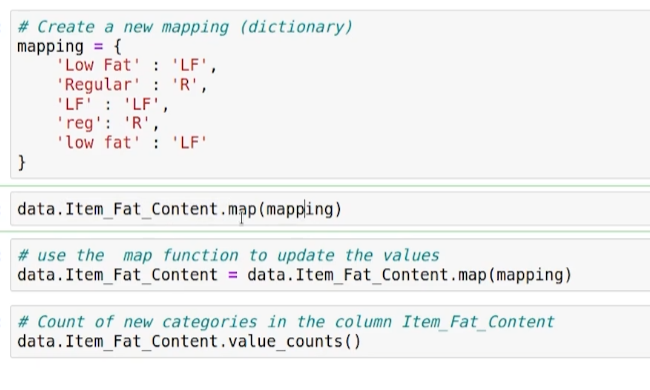
## mode

data.item.mode()

give most frequent category as output

## mapping

can change big categories name to smaller



Last line out put will be based on maping changes.

## Apply

Create new columns in data frame

Inside apply we can use lambda

Data[‘item\_usd’] = data.item.apply(lambda x : x/74)

## Get dummies

Convert categorical variables into mumerical variables

df\_train[df\_train["text"].notnull()]

take only non null

## Sort data

data.sort\_values(by=[roll\_no, marks], ascending=[True, False])

inplace = True update in original dataframe

now new dataframe have unsorted index. Lets use

data.reset\_index(inplace=True, drop = True)

it will remove old unsorted index and add new index

## Row wise merging of data

frames = [df\_train, merged\_df]

df\_train = pd.concat(frames, **axis =0**)

## Column wise merging of dataframe

frames = [df\_train, merged\_df]

df\_train = pd.concat(frames, **axis =1**)

## Join

Df2 = pd.DataFrame({

‘roll\_no’:[102, 103]

})

df.merge(Df2, how= left, on=’roll\_no’) #merge similarly as in sql

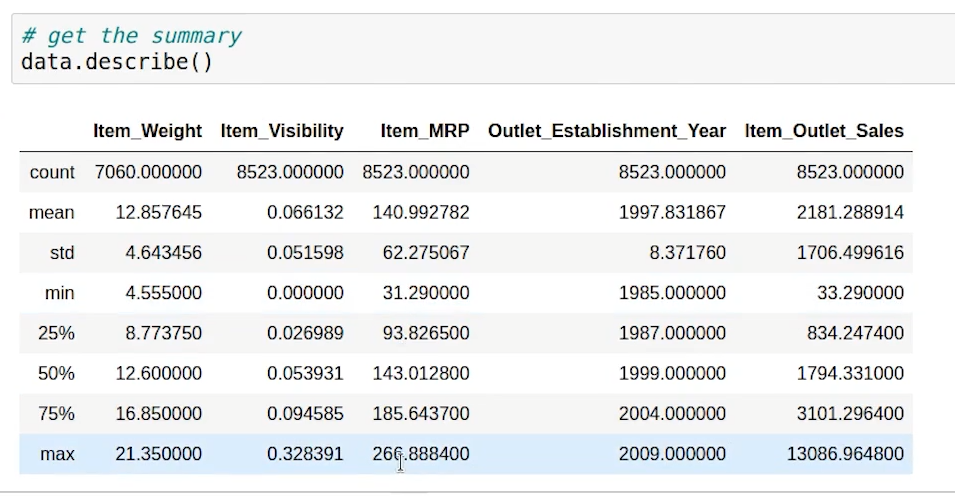
## mean

data.item.mean()

## describe

#get the summary

data.describe()



## Agregation

### Groupby

d1 = df.groupby('Animal')

### Pivot

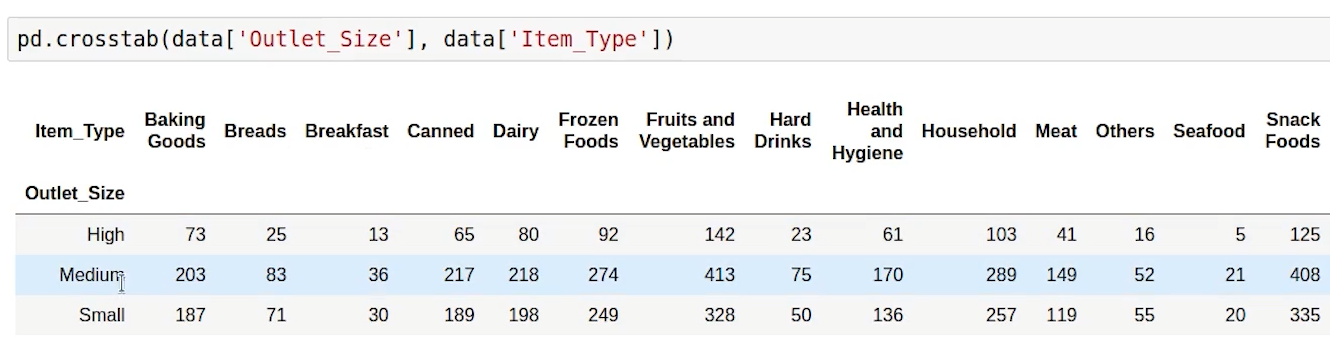
Pd.pivot\_table(data, index = ‘type’ values=’mrp’ , aggfuc=’mean’)

Aggregate type columns and apply mean operation on mrp columns

### Cross tab

Used to aggregate frequency of two or more factors.

Return give count



## Add mean in one columns

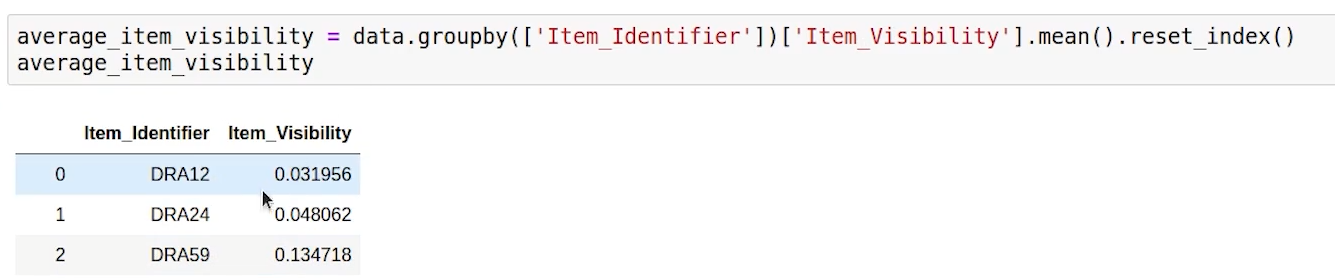
1st way

Data[avg\_item’] = data.groupby([‘item\_iden’])[‘item\_vis’].transform(‘mean’)

Group by item\_iden and take mean of item\_vis and add new column.

2nd way

Calculate avg



Make a method and call method by apply



# Day time library

## to\_datetime

Convert object type to DateTime

## day\_name

2012-08-25 to Monday

## month\_name

2012-08-25 to August

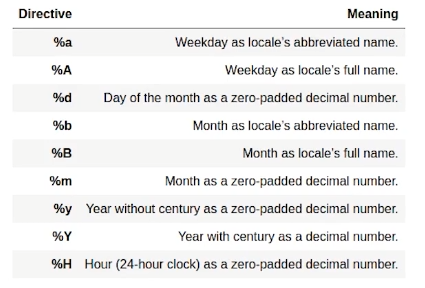
## dayofweek

## dayofyear

import datetime

datetime.date.today()

## Day time library directive



## tz\_convert

data['utc\_timezone'] = data.asia\_timezone.dt.tz\_convert('UTC')

## timestamp

data\_with\_unix\_ts.timestamp = pd.to\_datetime(data\_with\_unix\_ts.timestamp, unit='s')

## dropna(how="any")

# drop the null values

data\_BM = data\_BM.dropna(how="any")

df.dropna(axis=0)

## plot

# draw the plot

plt.plot(calories\_burnt,marker= 'o')

plt.plot(weight,'y--', marker='\*')

# add legend in the lower right part of the figure

plt.legend(labels=['Calories Burnt', 'Weight'], loc='lower right')

# set labels for each of these persons

plt.xticks(ticks=[0,1,2,3], labels=['p1', 'p2', 'p3', 'p4']);

## subplots

# create 2 plots

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(12,6), sharex=True, sharey=True)

# plot on 0 row and 0 column

ax[0].plot(calories\_burnt,'go')

# plot on 0 row and 1 column

ax[1].plot(weight)

# set titles for subplots

ax[0].set\_title("Calories Burnt")

ax[1].set\_title("Weight")

# set ticks for each of these persons

ax[0].set\_xticks(ticks=[0,1,2,3]);

ax[1].set\_xticks(ticks=[0,1,2,3]);

# set labels for each of these persons

ax[0].set\_xticklabels(labels=['p1', 'p2', 'p3', 'p4']);

ax[1].set\_xticklabels(labels=['p1', 'p2', 'p3', 'p4']);

## Line Chart

plt.plot(x, y, marker = 'o');

## Bar Chart

plt.bar(x, y, color=['red', 'orange', 'magenta']);

## Histogram

plt.hist(data\_BM['Item\_MRP'], bins=20, color='lightblue');

## Box Plot

red\_diamond = dict(markerfacecolor='y', marker='D')

plt.boxplot(data.values, labels=['Item Weight', 'Item MRP (price)'], flierprops=red\_diamond);

## Scatter Plot

plt.scatter(data\_BM["Item\_Weight"][:200], data\_BM["Item\_Visibility"][:200]);

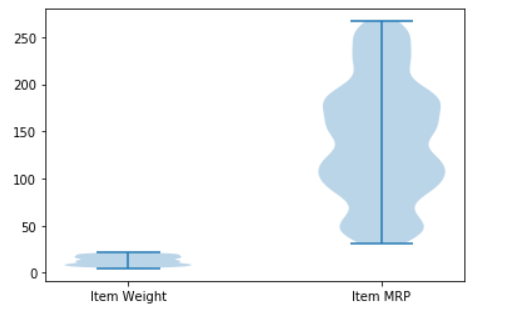
## Violin Plots

# add labels to x axis

plt.xticks(ticks=[1,2], labels=['Item Weight', 'Item MRP'])

# make the violinplot

plt.violinplot(data.values);



## scatter

plt.scatter(data\_BM["Item\_Weight"][:200], data\_BM["Item\_Visibility"][:200]);

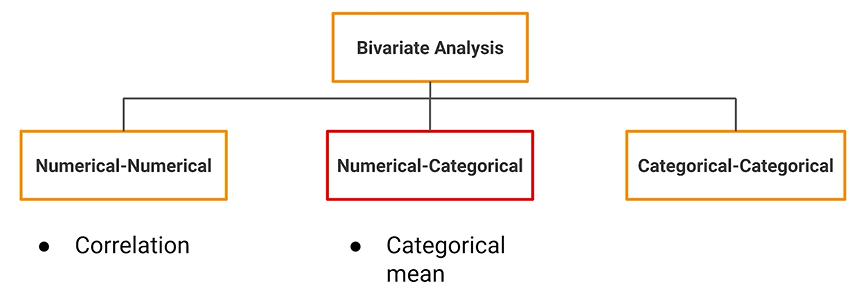
https://courses.analyticsvidhya.com/courses/take/applied-machine-learning-beginner-to-professional/lessons/12903622-data-visualization-with-matplotlib

## Bivariate analysis

Relation between two vairables

Hypothesis analysis

Ideas ofr features engineering and feature selection



Numerical Numerical analysis:

Variance : average squared difference of values from mean(only used for single variables)

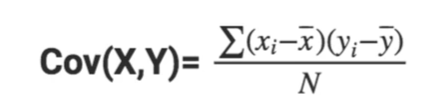
Standard deviataion : square root of vairance

Coveriance

Only give the sense of direction

two variables (-inf to +inf)

Positive negative and zero covariance.



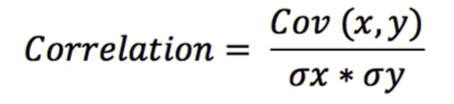
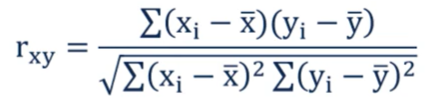
Difficult to work for 0.0000045 and 30000000000000000

So to resolve this issue a concept comes into **correlations.**

### Correlation

**Relationship btn two sets of variables used to describes the direction and strength of the relationship.**

### Pearson Correlation:

Covariance divided by the product of standard deviation of the two variables.

Values between -1 to +1

Strength of linear relation

##### Monotonic relationship

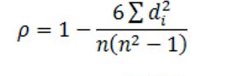
values of one vairables increases other decrases or vice versa,



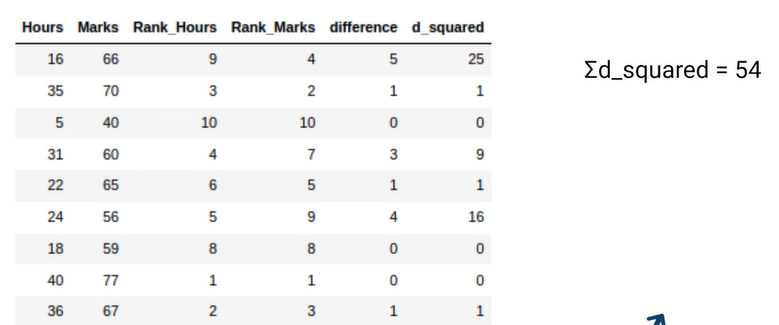
### Spearman Rho Correlation

Determines

-1 to +1



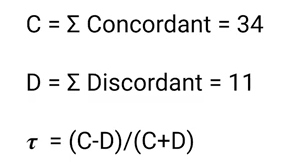
How to calculated d^2:

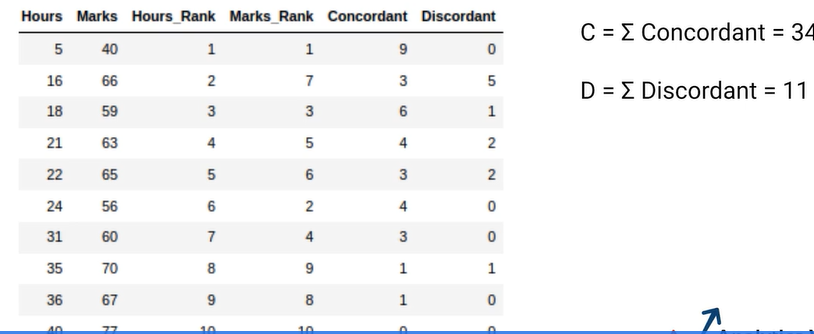


Rho: greek letter

https://courses.analyticsvidhya.com/courses/take/applied-machine-learning-beginner-to-professional/lessons/12906590-spearman-s-correlation-kendall-s-tau

### Kendall's Tau





How many are below a particular rank and vice versa.

KT prefered over SR

SR is sensitive to outlier, KT is robust

SR based on difference btn in rank and KT based on concordnant and discordant pairs.

Small dataset -> KT

Big dataset -> SR

### Usage

#### PCC

Linear relationship

Normally distributed(Assumption)

Continous data

No Outliers

#### SR and KT:

Monotic relationship

Outliers -.> KT

Not assumption of normal distribution of data

Only oridnal variables

### Causation

cause and effect, is when one variables outcomes casue the other variable outocomes.

“Correlation is not causation”

In many cases correlation, are just because of the concidences.

There is presence of one other factoe which is affecting the two variables.

This phenomina is also known as Spuriousness and spurioius coorelation

Population is compounding factor

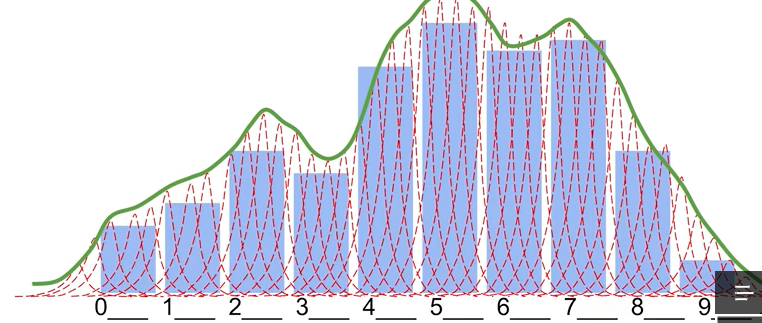
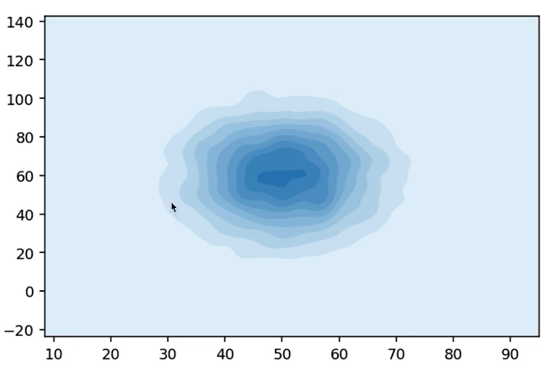
So we have to take care not to make the direct conclusion.



Heat map : good coorelated variables btn data.

Scaterplot : two continous variables are vary and interact with each other.

KDE: too many data with unable to find the pattern or dense data

# Classification algorithm evaluation

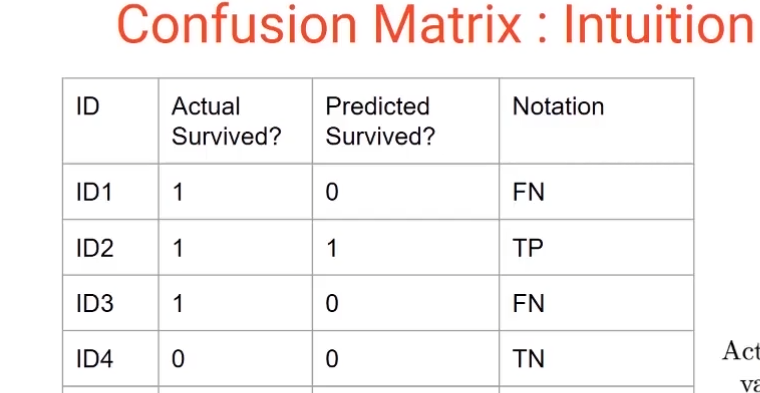
## Evaluation matrix

Confusion matrix-

total actual positive = TP + FN

total actual negative = FP + TN

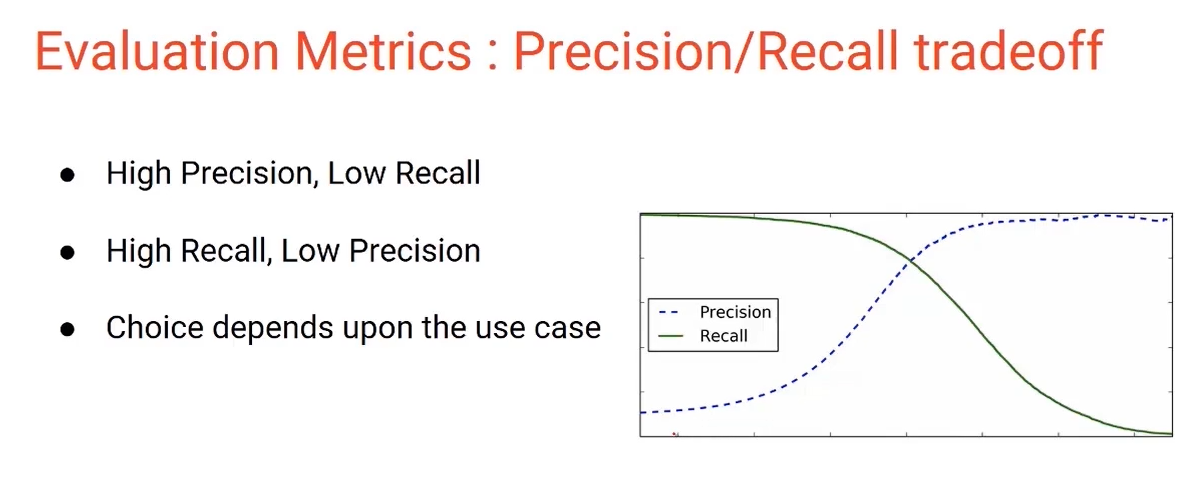
eg FN – predictive value is N means 0, actual value is F means 1





For precision it is important to emphasize on the false positive ( minimize fp)

For recall it is important to emphasze on the false negatice(minmize fn)



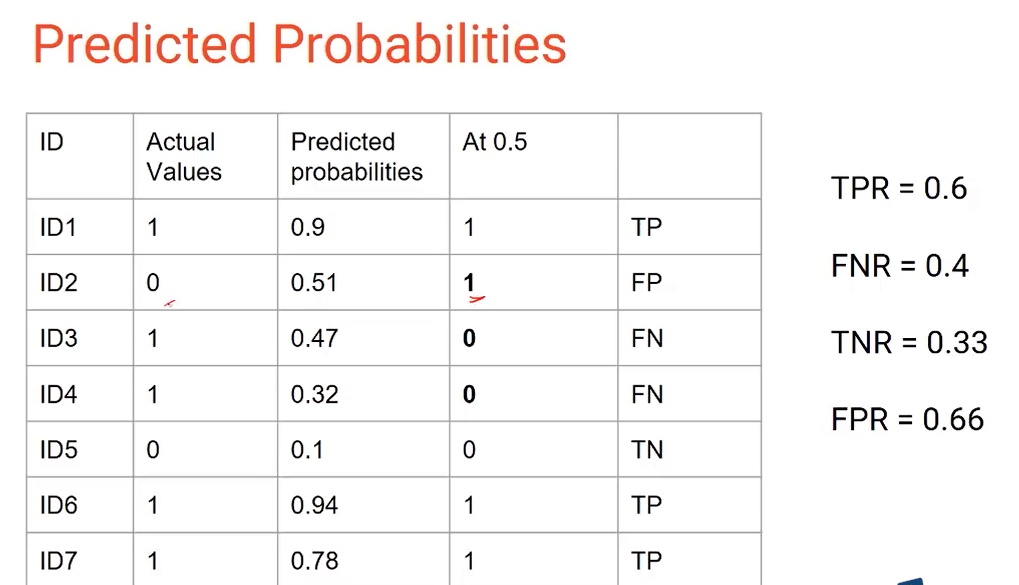
It all depend upon the requirement what is required. But some of the cases we need to balance both which we can do by F! score.

**Lessor the f1 score better will be our model : false**

**Wht is maximum score of f1 when recall = precision**

## Thersholding**:**

**It is way to get different values for TNR, FNR, TPR, FPR by changing the thershold values.**

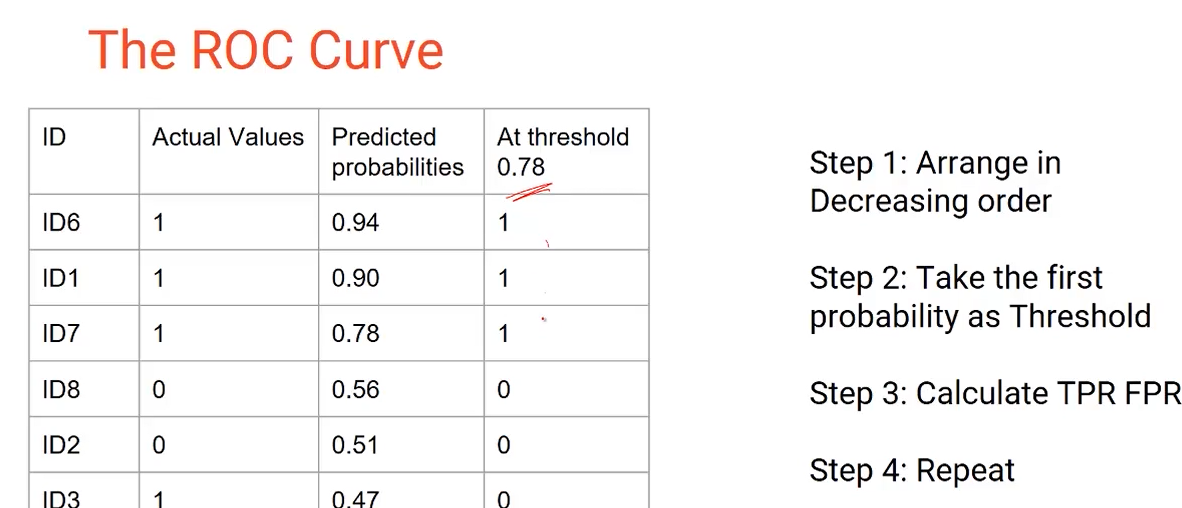
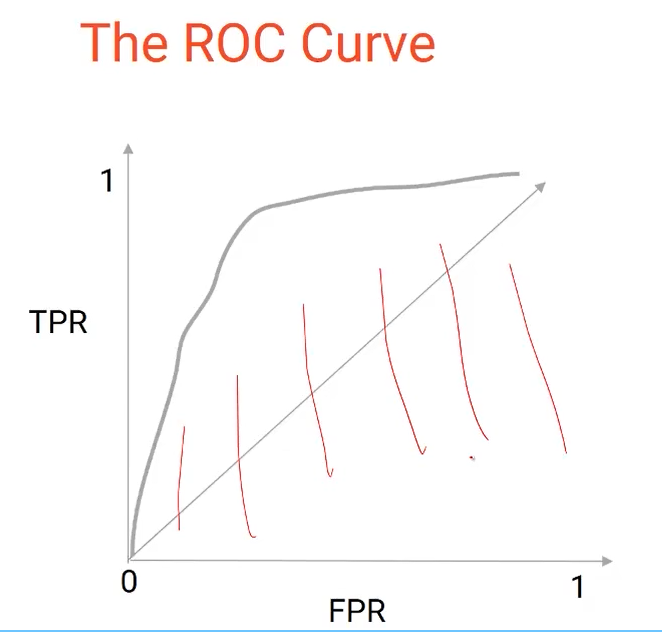


## AUC- ROC

**Area under curve and Receiver operator curve**

Order the predicted probality in ascending order and then calculate tpr and fpr values.

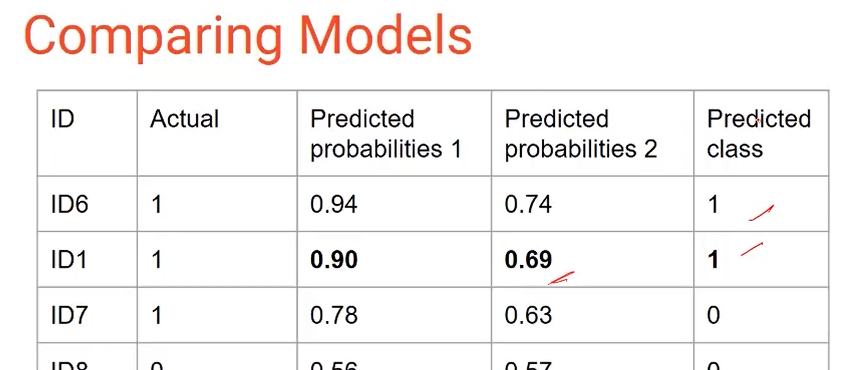
Same step will be performed for all the probality.

When FPR is greater than TPR, the AUC ROC curve would be below the 0.5 area line (generated by random guessing of predictions).

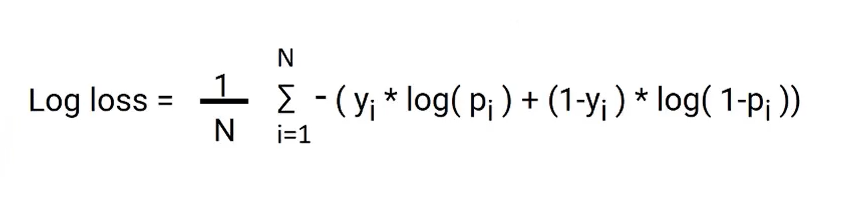
### Issue with auc-roc

* it is completely based on the order of the probality.
* Can not used for comparing the two models.



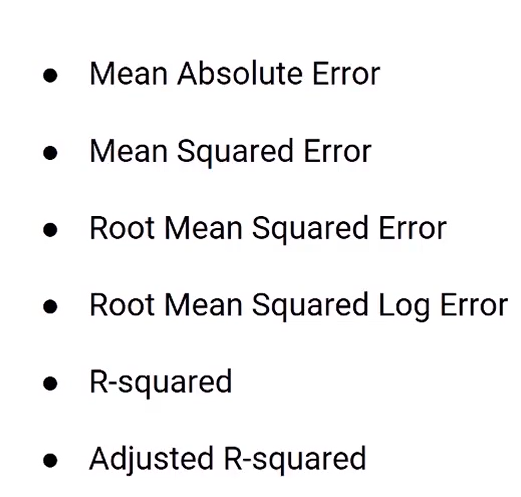
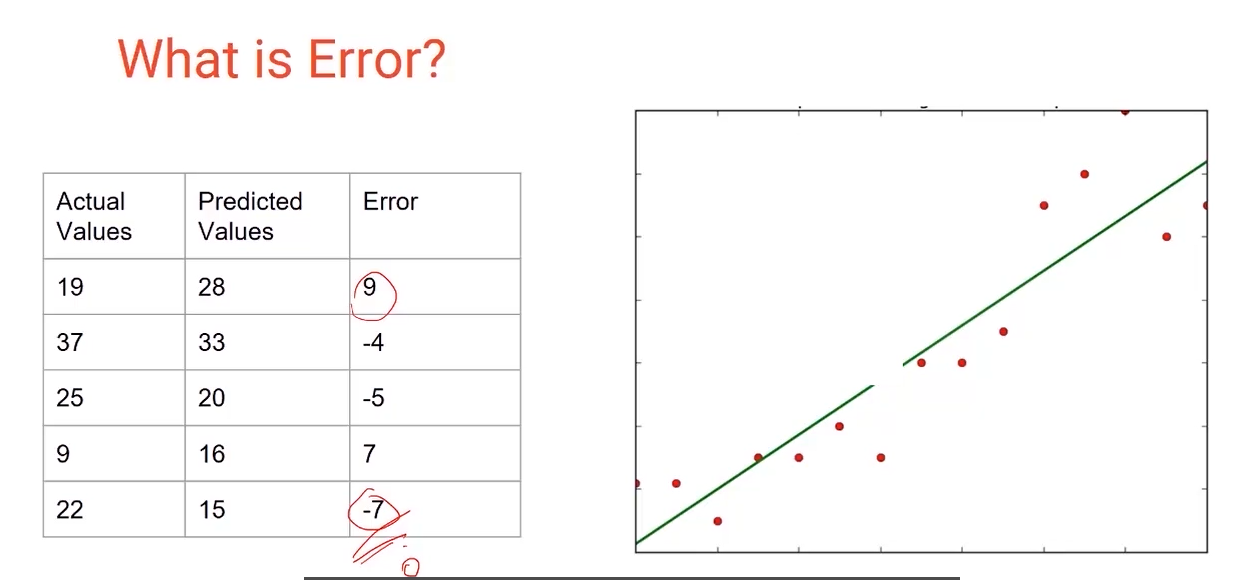
Log Loss:

To compare two model btter to use log loss:

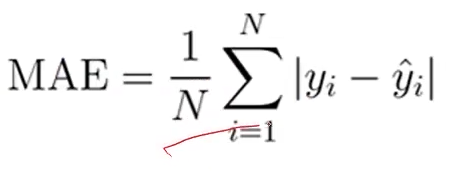


The lower the log loss of a model, the better its predictions. So model B is better than model A.

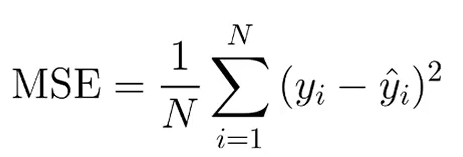
# Regression algorithm evaluation

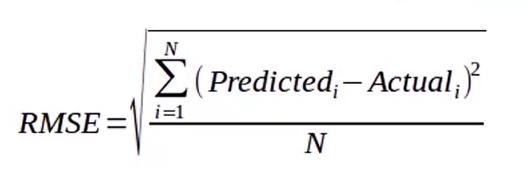
## MAE



## MSE

 this chnge the unit of the error to overcome this issue we will use the square root of the mse.

## RMSE



It will same for these cases

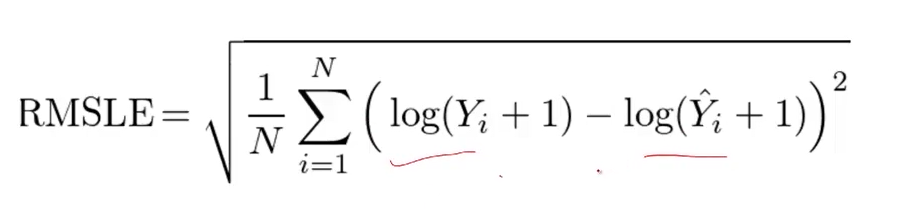
Actual predicted

1. 401

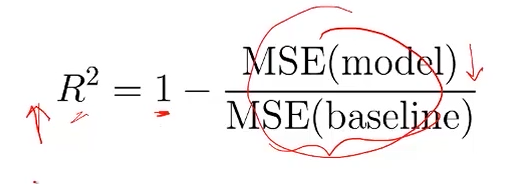
10000 10400

First case is reallly badly predicted but MSE assumes both the cases equally. We will same result of rmse. So to overcome this issue new approach with log formula is comes into picture.

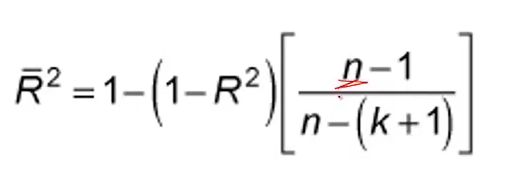
## RMSLE



## R- Squared

 if mse decrease the model will perform much better. On addition of new variables either r-square increase and remain same.

Adjusted R-Square

 n is number of sample and k is number of sample.

# Preprocessing the data

df.corr()

df.nunique()

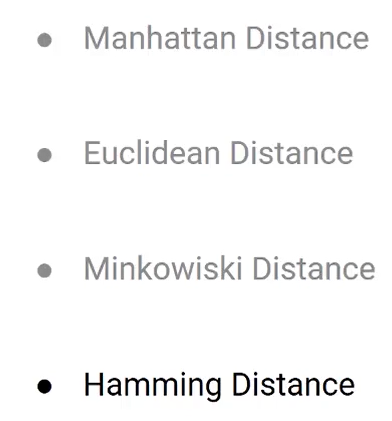
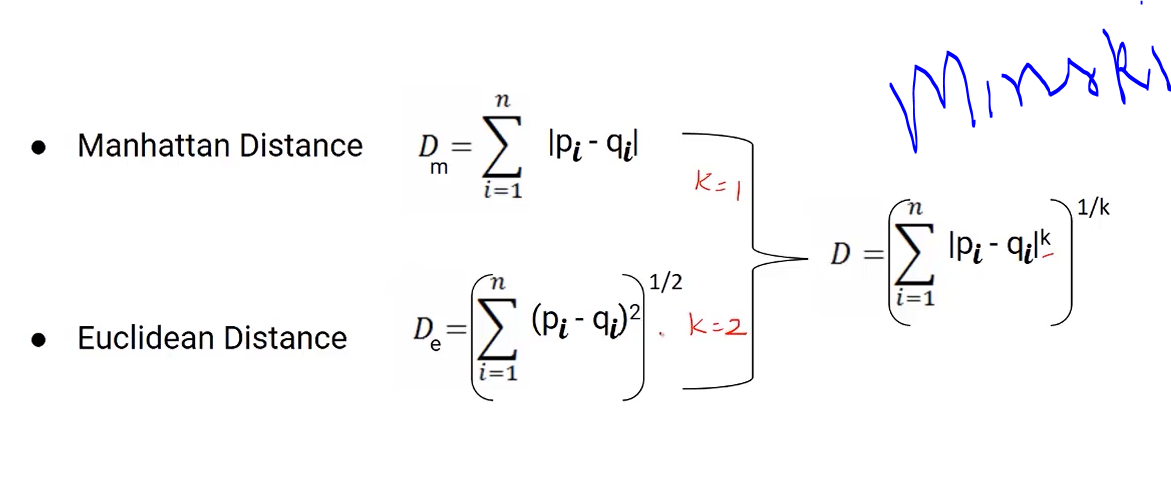
one hot encoding?

pd.get\_dummies(data\_cleaned)

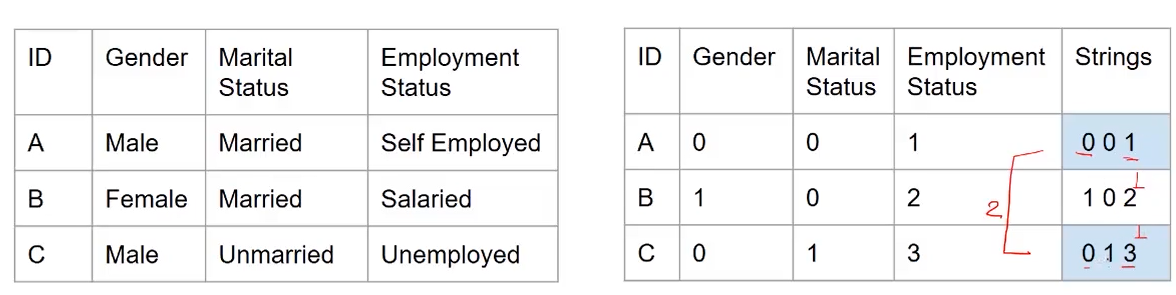
label encoding?

data[“embarked”].map({“S”:0, “Q”:1, “T”:2})

# Different type of distance

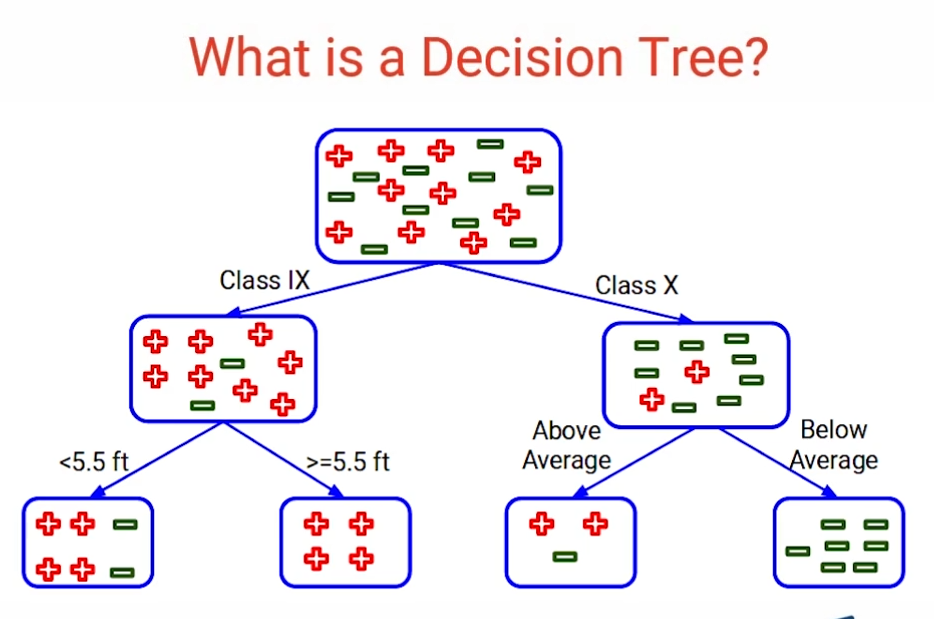
 

Hamming distance is used for the categorical variables.



# Decision Tree

Objective to pure decision tree



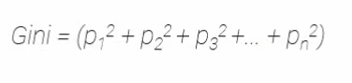
What are different terminlogy to split the data?

## Purity in decision tree

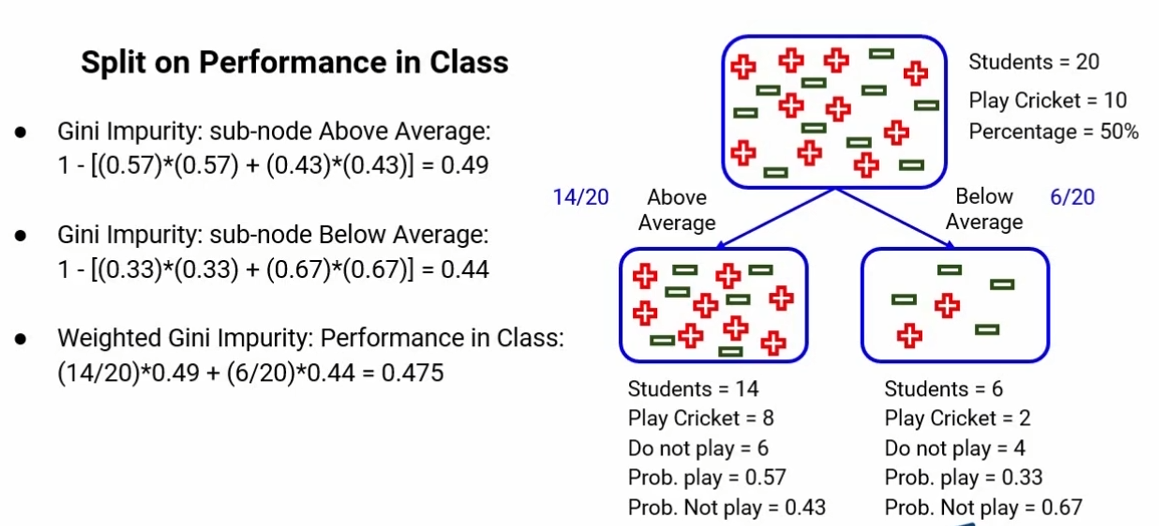
Where all positive and negative segregatted properly.

## Gini impurity

* Higher gini more is purity.
* Only binary split possible with gini
* Node split is decided based on the gini impurity
* Lower the gini impurity higher the homogeneity of nodes
  + Gini impurity = 1 – Gini
* Works only with categoricals not with continous target( no of bike rented, price of the house)
* Sum of square of probablities for each class/category



### Steps to calculate Gini impurity for a split

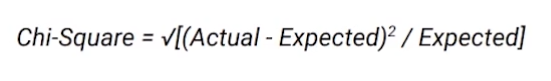


3 impotant point from above:

* Gini impurity for sub node left
* Gini impurity for sub node right
* Gini impurity for for performance in class

## Chi Square

* Statstical difference between child and its parents node.
* It is measured as



Higher the chi square better will purity of tree

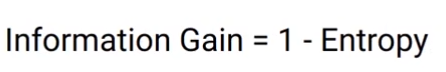
Work only for categorical

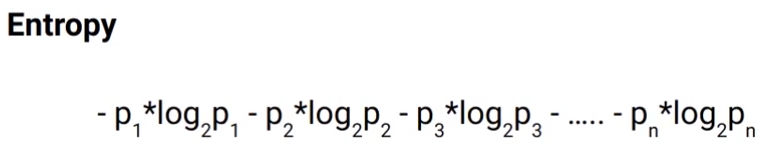
Can perform more than two split

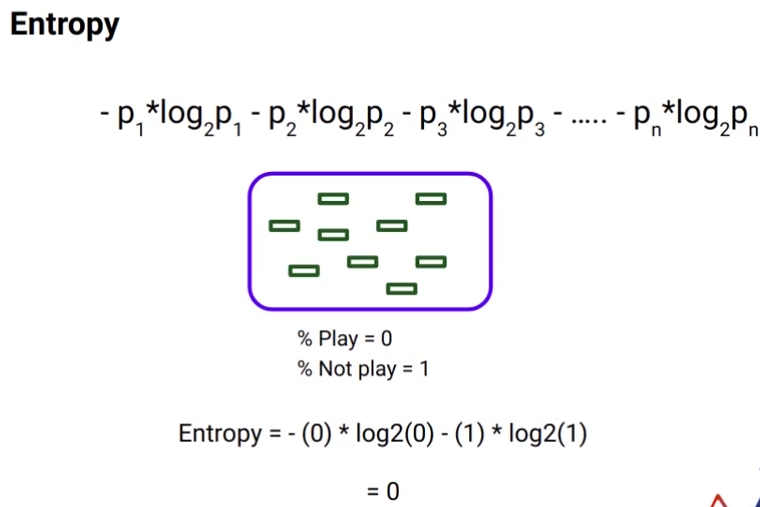
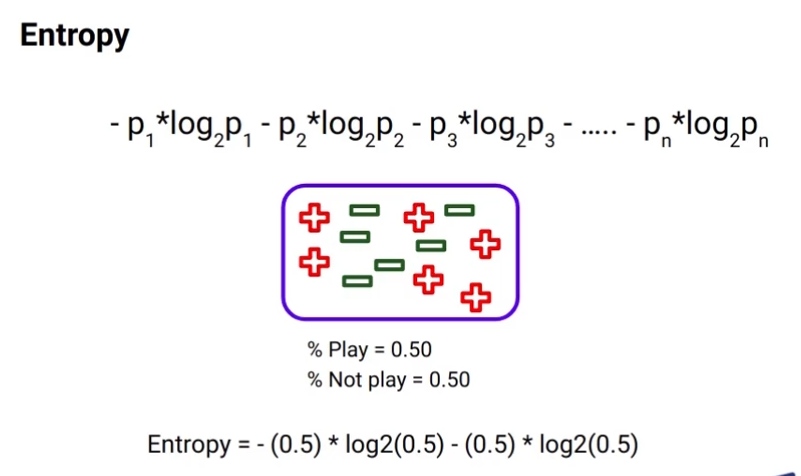
## Information gain

Higher IG better the tree

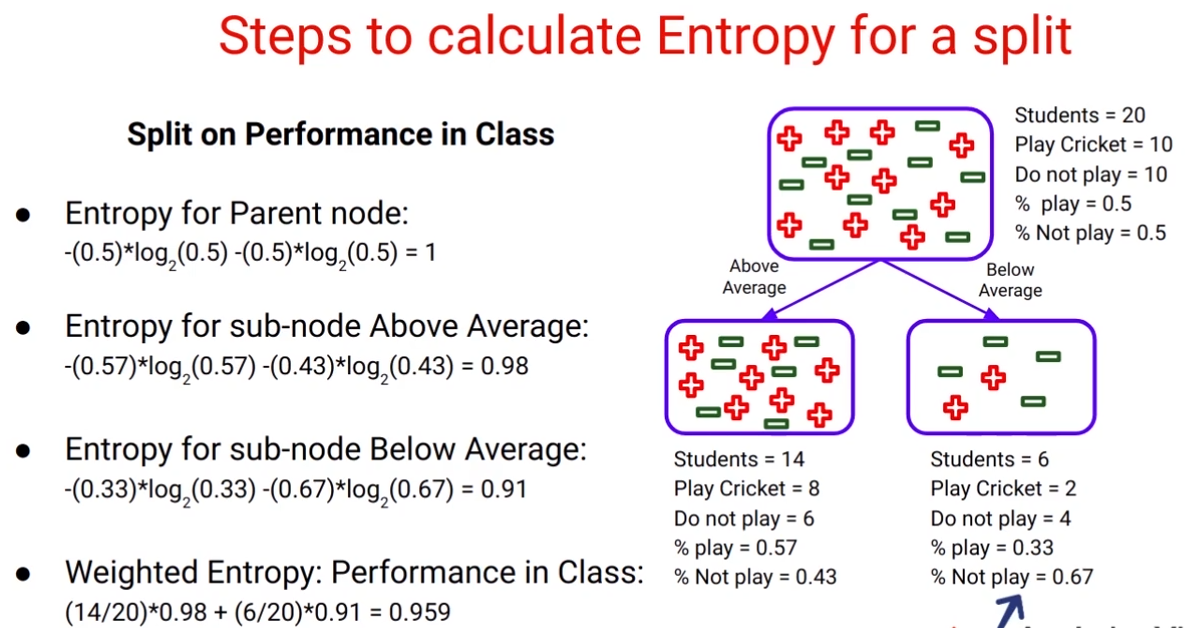
Lower the entropy better the tree

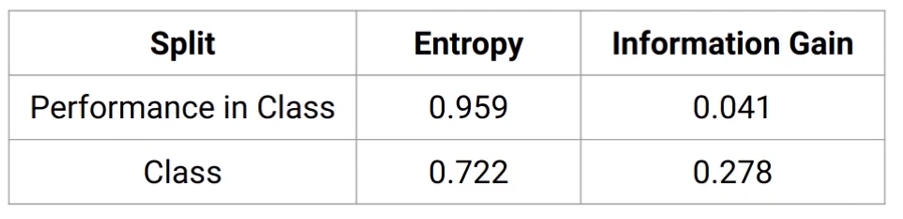




* Work only with categorical varaibles
* Lesser entropy higher the homogeneity
* Steps to calculate the entropy
* Calculate entropy of parent then child, and average entropy of the split





Class split variables **provide highest pure node**.

🔹 Python ([http://python.aman.ai](http://python.aman.ai/)): The swiss army knife for AI and beyond. Most deep learning frameworks are based on Python.  
🔹 PyTorch ([http://pytorch.aman.ai](http://pytorch.aman.ai/)): The holy grail of deep learning frameworks. Developed by Meta, most commonly used.  
🔹 TensorFlow ([http://tensorflow.aman.ai](http://tensorflow.aman.ai/)): The second most common deep learning framework. Developed by Google.  
🔹 NumPy ([http://numpy.aman.ai](http://numpy.aman.ai/)): Numerical computing with multi-dimensional arrays.  
🔹 Pandas ([http://pandas.aman.ai](http://pandas.aman.ai/)): Data manipulation and analysis.  
🔹 Matplotlib ([http://matplotlib.aman.ai](http://matplotlib.aman.ai/)): The most commonly used plotting library.  
🔹 SQL ([http://sql.aman.ai](http://sql.aman.ai/)): Storage, manipulation, and retrieval of structured data.