# Data Science and Machine Learning

Syllabus

* Prerequisites
  + Python
  + Python Installation • Jupyter Notebook Tutorial • Variable • Function • Lambda Expression • Loops • List • Tuple • Set • Dictionary
  + Advanced Python
    - Introduction to Numpy

• Creating Arrays

• Selection and Indexing

• Basic Operations on Arrays

• Mathematical Operation on Arrays

• Linear Algebra Operation on Arrays

• Stacking Arrays

• Data Types and Type Conversion

• Assignments

* + - Introduction to Pandas

• Creating Data Frames

• Reading and Writing Operation

• Selection and Indexing

• Conditional Selection

• Assignmet-5

• Groupby and Pivot table

• Merge,join,concat

• Assignment-6

• Missing Value Treatment

• Drop Duplicates

• Dealing with Date Time Data

• Apply()

• Introduction to Series

• Series Operation

• Pandas Builtin Functions for Data Visualization

• Assignment-7

* + Jupyter Notebook
    - Shortcuts
    - Export to presentation file
  + Visualizations
    - Introduction To Plotly

• Scatter Plot

• Line Plot

• Scatter Matrix

• Box Plot

• Bar Chart

• Histogram

• Sun Burst Chart – used to present hierarchal data in dognut.

* + - * Libraries
      * Bookeh
      * Matplot
      * Seaborn
      * Plotly
  + Statistics
    - Central Limit Theorem

• Measure of Dispersion

• Quartiles, Inter Quartile Ranges

• Variance

• Standard Deviation

• Z Score

• Normal Distribution

• Pearson Correlation Coefficient- R

• R Square

• Adjust R2

• Outliers Detection and Removal

• Assignment-9

* [Overview, Concepts and Terms](#_Overview_-)
  + Essemble Learning
  + Bagging
  + ML Lifecycle/Phases
  + [Least Square Method](#_Least_Square_Method)
  + [OLS - Ordinary Least Squares](#_OLS_-_Ordinary)
  + [RSS](#_RSS)
  + [TSS](#_TSS)
  + [Mean square error and rsquare error](#_Mean_square_error)
* Statistical Foundation
  + [Basic Mathematics](#_Basic_Mathematics)
  + Statistical Maths
  + Physics

* [Data Analysis and Preparation](#_Data_Analysis)
  + Exploratory Data Analysis
    - Univariate EDA
      * Graphical Methods
      * Non-Graphical Method
    - MultiVariate EDA
    - Categorical variables correlation
    - Tidy dataset
    - Sampling –
      * BootStrap
      * Random
  + Classical Statistical Data Analysis Methods(Quantitative methods)
    - Interval Estimation
    - Hypothesis Tests
  + Feature Engineering –
    - [Feature Types](#_Feature/Variable_Types_–)
    - Analyze data using Describe command
    - Find feature correlation using heat map chart. Find relationship using scatter
    - Permutations and Combination
    - [Feature selection](#_Feature_selection)
  + Data Pre-processing
    - [Data set preparation](#_Data_set_preparation)(data wrangling, fill nan, join files, encrypt PII fields, replace category with numbers/enums/Label Encoding/Hot Encoding)
    - Model generalization
    - [Data Standardization and Normalize](#_Data_Standardization_and)
    - Data Under fitting
    - Data Over fitting
    - Data scaling methods
    - [Data Smoothening](#_Data_Smoothing_Technique)
    - [Dimensionality Reduction](#_Dimensionality_Reduction) 
      * [PCA – Principal Component Analysis](#_Principal_Component_Analysis(PCA))
      * [LDA – Linear Discrimination Analysis](#_Linear_Discrimination_Analysis)
    - [ML Type](#_Machine_learning_types-)
      * Supervised Learning,
      * Un-Supervised Learning
      * Reinforcement Learning
    - [Matrics-](#_Matrix-)
      * Decision Tree,
      * [Confusion Metrics](#_Confusion_Matrix) Accuracy, Precision, F1Score, Recall
      * Covariance Matrix
    - Model Selection and Boosting
    - [Association Rules Mining](#_Association_Rule_Mining/Market)
      * Apriori Algorithm
      * Market Basket Analsis
    - [Recommendation System](#_Types_of_Recommendation)
      * Collaborative Filtering
      * Content-Based Filtering
* [Algorithms](#_Algorithms)
  + Trend Algorithms
    - [Logarithmic](#_Logarithmic)
    - Linear
    - [Exponential](#_Exponential)
    - Polynomial
  + Algorithm types –
    - Classification models
      * KNN
      * Naïve Byes classifier(Byes theorem)
      * Support Vector Machine
      * Logistic Regression
      * Decision Tree
      * Random Forest
      * XGBoost
      * Convolutional Neural Netrowk
      * Recurrent Neural Network
    - Clustering
      * [KMean](#_--------------------K-Mean---------)

• What is Euclidian Distance

• Introduction to Unsupervised Learning

• Step By Step Mathematical Derivation

• Pros and Cons Of K Means

• Elbow Method to Find K

• Project-5

* + - Regression
    - Segmentation
  + Z-Score scaling
  + Lazy Algorithm
  + Greedy algorithms
  + [Random forest](#_Random_Forest)
  + [Linear Regression](#_Linear_Regression_Algorithms)

• Theory of Linear Regression

• Cost Function

• Optimization Using Gradient Descent

• Mathematical Interpretation of Gradient Descent

• Project-2

• Understanding Why Linear Regression may fail?

• Model Validation Techniques

• Mean Squared Error

• Root Mean Squared Error

• Mean Absolute Error

* + Decision Tree

• ID3 Algorithm vs CART

• Entropy

• Information Gain

• Step by Step Understanding of How Decision Tree Work

• How to overcome overfitting in DT

• Cross Validation

• Bootstrap Aggregation/Bagging

• Introduction to Random Forest

• How Random Forest Works

• Feature Selection

• Model Validation Techniques

* Accuracy
* Confusion Matrix
* Classification Report
* Recall
* Precision

• Project-4

• Hyper parameter Tuning

* + logistic regression
  + Clustering Algorithms
* [Time Series Analysis](#_Time_Series_Analysis)
  + AutoRegressive(AR) Model
  + Moving Average(MA) Model
  + [ARIMA Model](#_ARIMA_Autoregressive_)
  + Stationary/Non-Stationary
  + [Dickey Fuller Test](#_Dickey_Fuller)
  + ACF (Auto Correlation Function)
  + PACF (Partial Auto Correlation Function)
* [Model Evaluation Metrics, performance tuning, scoring and selection –](#_Model_Evaluation)
  + Sampling based evaluation - KFold, Leave one out Sampling Method, Leave POut method, Repeated KFold, Shuffled KFold, BootStrap
  + Metric based evaluation
    - Metrics for model Evaluation
      * Overall Model Evaluation(Accuracy, Balanced Accuracy, Cohen-Kappa)
      * Class of interest/Focused class model evaluation(Precision, Recall, F1 Score)
      * ROC (Receiver Operating Characterstic)
  + [Model and Variance evaluation parameters](#_Model_evaluation_parameters)
  + Validation
  + [Boosting](#_Boosting_–)
  + [AdaBoost Mechanism](#_AdaBoost_Classifier)

### Model Evaluation Metrics

* + Beyond Accuracy Score
  + Cohen Kappa Score
  + Precesion and Recall Score
  + Sensitivity and Specificity
  + ROC Curves
  + AUC Score
  + Feature Importance – Generate each feature importance explanation in model.
  + Exit Criteria – based on model evaluation, setting exit criteria on threshold or time.
  + Concurrent setting – setup number of concurrent training runs.
  + Scoring Metrics for Model Evaluation – Metrics are used to evaluate the model. Diff. metrics used for diff models. Based on requirement, we can use single or multiple metric for model. Note that metrics are for evaluating model performance not tuning or optimization; for that loss function is there. However some metrics works as Loss function as well.
    - For Classification models
      * Accuracy – (number of correct predictions/total predictions)\*100
      * Precision – True Positive/(True Positive+False Positive)

Precision is more focused on correct results. For example, if we need to predict cat out of multiple images. Then the total correct predicted cats divided by total cat images. It is useful when data is unbalanced as there are chances we trained model on cat images only but ignored other non-cat images.

* + - * Recall
      * F1 Score
      * ROC(Receiver Operator Characteristic)
      * AUC
    - For Regression
      * MSE(Mean Squared Error)
      * RMSE(Root Mean Squared Error)
      * MAE
      * Coefficient of Determination(R2)
    - Ranking Metrics
      * MRR
      * DCG
      * NDCG
    - Statistical Metrics
      * Correlation
    - Computer Vision Metrics
      * PSNR
      * SSIM
      * IoU
    - NLP Metrics
      * Perplexity
      * BLEU Score
    - Deep Learning
      * Inception Score
      * Frechet Inception Distance
  + Loss Functions- Loss functions are functions to show measure of Model performance and are used to train the model with optimize performance.
    - * Cross-Entropy
      * Hinge
      * Huber
      * Kullback-Leibler
      * RMSE
      * MAE (L1)
      * MSE (L2)

**Model Selection Methods using HyperParameters-**

* Grid Search CV
* Random Search CV
* Ensemble Techniques – (Random Forest, Bagging, Stacking classifer)
* [Visualizations tools](#_Visualization_tools_-)
  + ML with Power BI
  + Chart types
    - [Compare values across categories](#_Compare_values_across)
    - [Part to whole relationship](#_Part_to_whole)
    - [Relationship among variables](#_Relationship_among_variables)
    - [Compare over time](#_Compare_over_time)
    - [Frequency (Count) of values](#_Frequency_(Count)_of)
    - [Maps](#_Maps)
    - [KPI](#_KPI)
* [Case Study](#_Case_Studies-)
  + Prepare you own data set and run each algorithm on that
  + Resolve industry case studies – UCI ML Repository
  + Other terms-
    - Entropy
    - Depth - Depth of tree
* [Other practices](#_Other_practices)/Resources
  + [Requirement Elicitation/Gathering](#_Requirement_Analysis,_Elicitation)
  + White papers
  + [Python LibrariesPyPi, Seaborn, Scikit Learn, pydotplus](#_Libraries)
    - Numpy
    - Scipy
    - Pandas
    - Matplotlib
    - Seaborn
    - Plotly
    - Scikitlearn
    - TensorFlow – deep Learning
    - Torch/Py-Torch
    - Caffe
    - OpenCV – Deep learning inference on Image and Videos
    - Sickit-image - image processing
    - FER - Face detection file/images
    - Librosa - Audio processing
    - Geeksforgeeks.com
  + Other topics -
  + Regex Extract title, first and last name
  + Visit Communities - <https://data-science-infinity.teachable.com/courses/data-science-infinity?affcode=716157_jcwmqdhh>
  + <https://www.ibm.com/in-en/topics/knn>
  + https://iopscience.iop.org/
  + Theorems Pythagorean theorem
  + Methods - Euclidian, Manhattan, Canberra
  + Go through statquest videos
  + Take assessments
  + Participate in statical maths challenges
  + Deep Machine Learning and Neural Networks
  + Hackathon/challenges - https://datahack.analyticsvidhya.com/
  + Learning Portals- <https://www.analyticsvidhya.com/>
  + <https://waikato.github.io/weka-wiki/datasets/>
  + <http://saedsayad.com/data_mining_map.htm>
  + Kaggle.com
  + Javapoint.com
  + Mathisfun.com
  + Huggingface.co - for pre-trained models
  + Datasets
    - https://urbansounddataset.weebly.com/urbansound8k.html - Audio files download
    - Mnist dataset - face images files
* Competitor Languages tools
  + Julia
  + Weka
* [Certifications](#_Certifications)
  + Go through skill pipe book
  + Azure Data Science Certification
  + Azure ML Certification
  + Python certification
  + TCS ML Assessments

# Full Stack Deep Learning –

**Deep Learning -**

* What is Deep Learning
* Deep Learning VS Machine Learning
* What is a Perceptron
* How Neural Network Learns
* Multi Layer Perceptron
* Activation Function
* Introduction to Keras
* What is Feed Forward Network
* Detail Explanation of ANN
* What is Cost Function
* Optimization Technique
* Vanilla Gradient Descent
* Mini Batch Gradient Descent
* Stochastic Gradient Descent
* Softmax
* Cross Entropy Loss
* MSE vs Cross Entropy
* Project-6

Libraries -

* PyTorch – Sentiment analyzer, Image classifier
* Keras – for prediction. Hybrid of Tensorflow and Theano
* Tensorflow – It is similar to Numpy
  + Vector Operation
  + Dot Product Matrix Multiplication
  + What is Gradient
  + What is Loss function
* Theano
* OpenCV – Image classifier, Computer Vision
* Natural Language Processing
  + Spacy
  + Gensim
  + NLTK
* Run code on big data cluster
* Build recommender engine using Apache Spark
* Web Scraping/Collecting data -
  + Beautifulsoup
  + Selenium
* Intro to Deep Learning
* What is a Tensor
* Vector Operation
* Dot Product, Matrix multiplication
* What is Gradient
* What is Loss Function

##### Artificial Neural Network (TensorFlow and Keras)

* The Neuron
* The activation function
* How Neural Network Work
* How Neural Network Learn
* Softmax
* Cross Entropy
* Gradient Descent
* Stochastic Gradient Descent
* Full vs Batch vs Stochastic Gradient Descent
* Back Propagation
* Evaluating ANN
* Improving and FineTuning ANN
* Introduction to Theano And Transflow
* Introduction to Keras

##### Convolutional Neural Network (PyTorch)

* What is Convolution
* Convolution Example
* Convolution Operation
* ReLu Layer
* Pooling
* Flattening
* Full Connection
* Introduction to Image Processing
* Image Processing Theory Deep Dive

##### Natural Language Processing(Gensim and spacy)

* What is Unstructured Data
* Introduction to NLTK and Spacy
* Tokenization
* Stop Word Removal
* Stemming
* Lemmatization
* N-Grams
* What is Word Embedding
* Count Vectorizer
* Tf-Idf Vectorizer
* Pattern Matching
* Regular Expression
* Project-7
* Web Crawling
* Web Scrapping
* Regular Expression
* Beautifulsoup
* Selenium
* Project
  + Collecting Product Reviews from Social Media
  + Data PreProcessing
    1. Word Tokenization
    2. Stop Word Removal, Lemmatization, Stemming, n-grams, POS tagging
    3. TF-IDF
    4. Deep Dive int Naïve Bayes Algorithm
    5. Building the Sentiment Classifier

##### Word Embeddings

* + What is Word Embeddings
  + Word2Vec Implementation
  + Implementing Word2Vec with Theano

##### Data Visualization Using t-SNE

##### Named Entity Recognition

Pyspark

Installation

Spark DataFrames

Build a Recommender Engine Using PySpark

* Introduction to Recommendation Engine
* Collaborative Filtering
* Deep Dive into Alternative Least Square(ALS) algorithm
* Building a Recommendation engine using PySpark

**Model Deployment-**

* Introduction to AWS
* Create AWS Account
* Intro to AWS ECS
* Create Free Windows EC2 Instance
* Connect EC2 Instance
* Install required packages in EC2
* Deploy ML model on EC2 server
* Project-8- Deploy ML project on EC2

----------------------------------------------Overview, Concepts and Terms-----------------------

#### Overview –

Machine Learning Life cycle -

1. Problem Definition
2. Data Collection – Collect data from csv/excel and various sources
3. Data Analytics(EDA)
4. Data Cleansing
5. Data Visualization
6. Model Fitting/Training(ML)
7. Evaluation of Model
8. Model performance improvement
9. Deployment

## Machine learning types-

1. Supervised - This is learning model where we know input and output in advance. Eg. Labeling fruits as banana and apple. Accordingly expect ML alogirthm to judge output fruit.
2. Unsupervised - In unsupervised learning, we don’t have precise information of inputs and output. So we expect high level grouping into clusters. For eg. Group fruits based on its color. So as input we will just provide the color and don’t know the input fruits hence it can be anything. Once fruits are grouped the second run could be for further group based on fruit size. It is useful for raw data where identifying groups is tough. It cluster the data into classes/groups/partitioning based on statical properties.

There are 3 types of clustering-

1. Exclusive Clustering – Here item will group into one cluster only. One item cant go into multiple groups.

Eg. K-Mean Clustering

1. Overlapping Clustering – Here the items can go into more than one cluster due their properties.

#### Eg. Fuzzy C-Means Clustering

It is extension of K-Mean algorithm. Here the data point can group to multiple clusters with degree(distance from Centriod) of membership(ranging from 0 to 1).

Advantages –

1. More natural representation of the behaviour of genes.

Disadvantages-

1. Need to provide C i.e. number of clusters
2. Need to determine membership cutoff value
3. It is not deterministric algorithm.
4. [Hierarchical Clustering](#_Hierarical_Clustering) – Here the cluster have Parent-Child relationship. It consider every point as cluster then start merging them nearest cluster until achieve one cluster.

Agglomerative Clustering

Division Clustering

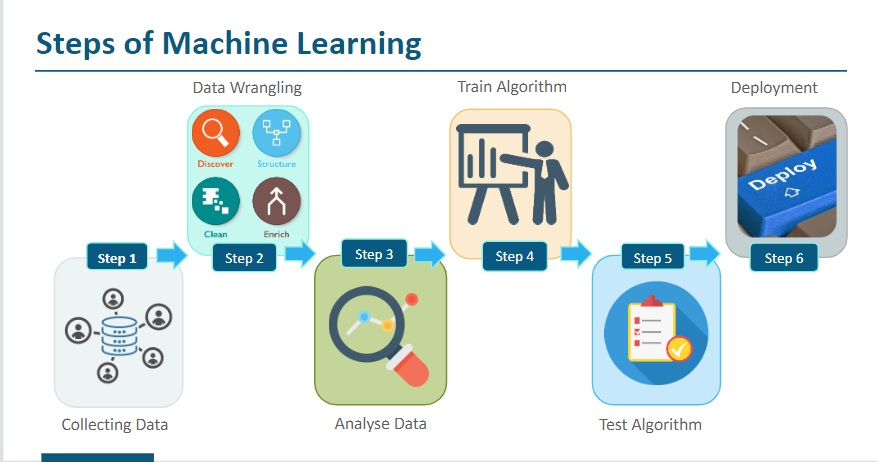
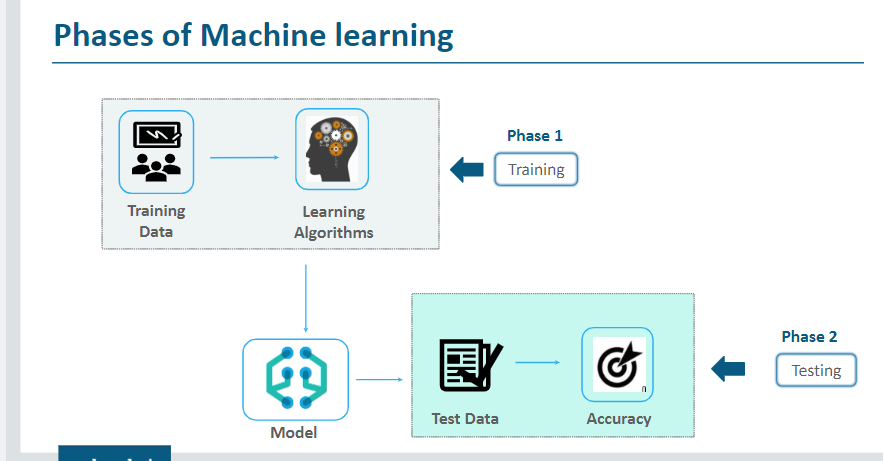
1. Reinforcement - Model learn from real time result from events. Eg. Driving car can learn from action while rating their actions a Good or Bad. It has Q Table having Qvalues(Good/Bad reward) into it.   
   Steps -   
   Initialize all Qvalues with 0 and randomly choose action  
   Get reward against action.  
   Update add/subtract QValue in previous value.  
     
   There are two ways for action -  
   Expolitatin Where old data will use for future action  
   Exploration - To take action randomly.

|  |  |  |  |
| --- | --- | --- | --- |
| **Supervised Learning** | **Un-Supervised Learning** | **Reinforcement Learning** | **Time Series Analysis(TSA)** |
| Function : y=f(x) | It is useful for where information is neither classified nor labeled. | It has agent to take decision and get feedback to learn |  |
| Algorithms :  KNN | K-Mean Clustering | K-Armed Bandit | Auto Regressive Integrated Model(ARIMA) |
| Linear regression(Used to estimate real values based on continues variable) | C-Means Clustering | Epsilon Greedy Algorithm | Moving Average(MA)   1. Simple Moving Average(SMA) 2. Exponentially weighted moving average |
| Logistic Regression | Hierarchical Clustering | Markov Decision Process(MDP) | ETS ( Error Trend Seasonability) |
| Ploynomial regression |  |  | Dickey Fuller Test |
| Decision Tree(Used for classification problems. It works for both categorical and continuous dependent variables) |  |  | ACF(Auto Correlation Function) and PACF(Partial-Correlation function) |
| Random Forest |  |  |  |
| Naive Bayes Classifier |  |  |  |
| SVM(Support Vector Machine) |  |  |  |
| Naïve Bayes Classifier |  |  |  |
| Support Vector Machine(SVM) |  |  |  |
| Hperparameter Optimization |  |  |  |
| Grid Search vs Random Search |  |  |  |

#### ML Lifecycle/Phases

Steps:

1. Collecting data
2. Data Wrangling
3. Analyze Data
4. Train Algorithm
5. Testing Algorithm
6. Deployment



### Least Square Method

Coefficient - a number which is placed before another quantity and which multiplies it, for example 3 in the quantity

##### OLS - Ordinary Least Squares

##### RSS

##### TSS

##### Mean square error and rsquare error – Squared Error means variation percentage that unable to explain, that means error in regression models. The difference between Squared Error and actual total is RSquare.

Coefficient of determination - The coefficient of determination is a number between 0 and 1 that measures how well a statistical [model](https://www.scribbr.com/frequently-asked-questions/what-is-a-model/) predicts an outcome.

* 0 means the model does not predict outcome
* 0 to1 means partially predicted outcome.
* 1 means perfect predicted outcome.

### Model Fitting – Fitting a model means that you’re making your algorithm learn the relationship predictors and outcome so that ou can predict the future values of the outcome.

So the best fitted model has a specific set of parameters which best defines the problem at hand.

In linear regression (y=mx+c), model will learn from m and c.

Three types –

* Under Fit
* Appropriate Fit
* Over Fit

We need to decide which fitting we are expecting from model based on requirement.

------------------------Usage based algorithms----------------------

# Algorithms Usage

|  |  |  |
| --- | --- | --- |
| Algorithm | Requirement | Description |
| Random Forest | NPA prediction. If borrower will pay the loan or not. |  |
| Decision Tree and Random Forest | Predict whether horse will survive or not based on medical details. |  |
| Heatmap, Scatter | Find correlation between attributes. |  |
| Apriopri Algorithm | To identify items together bought in super market. | It checks items together bought called item set. It also finds item subset within item sets. |
| KNN | Predict group/class, simple recommendation systems, pattern recognition, data mining, financial market predictions, intrusion detection, and more. | If data values are missing then this algorithm will work.  Based on user past online behaviour, it can decide it group and recommend contents but dataset should small.  In finance, Loan/Credit Card application grouping or Credit score calculation,.  Healthcare – Predict Heartattack based on health parameters.  Pattern Recognition – example we need to classify handwritten text. |
|  |  |  |
|  |  |  |

------------------------Usage based algorithms End----------------------

##### Boosting - AdaBoost Classifier

##### BellMan equation

----------------------------------------------Overview, Concepts and Terms END-----------------------

---------------------------------------------Statical Foundation-----------------------------------------------

#### Basic Mathematics

**Scalar** – Scalar means element or data point without direction. Eg. Distance, Running, Work

**Vector** – Vector means data points with direction. Eg. 25KM towards Left, Running 10KM toward North. It has line on top saying direction eg.

**Eigen Vector**

**Lambda(Λ)**

**Probability** – Chances of get particular output. For eg. Probability of getting Head or Tail from coin. Toss.

Let say we will toss coin 3 times and we have to find Probability of getting Head(H) 2 times.

First we will list down all the possible output i.e.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr.** | **Output** | **Probability** |  |
| 1 | HHH | Probability of 3 heads is 1/8 |  |
| 2 | HHT | Probability of 3 Head is 3/8 |  |
| 3 | HTH |  |
| 4 | THH |  |
| 5 | TTH | Probability of 1 Head = 3/8 |  |
| 6 | THT |  |
| 7 | HTT |  |
| 8 | TTT | Probability of 0 Head = 1/8 |  |

Now we have we can see the probability/chances of getting Head is {0,1,2,3} so this will be called as Probability Distribution. Here the {0,1,2,3} values are Random Variable(RV) values.

When Random Variable(RV) value is discrete then it is called Discrete Probability Distribution or Category Probability Distribution.

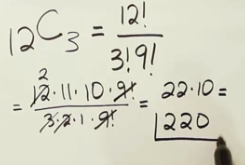
There are two types of Variables –

* Categorical/Discrete - as some specific attribute value like product color, size etc
* Continues - it can be anything like price of product, Age, Credit Limit

**Simple probability formula** – No. of possible outcomes/total of chances

**Combination Probability** (subset) – It is to calculate probability of combinations of values within set. Example, if we have 12 members nomination for 3 seats in committee then how many combinations of members we can have in committee?

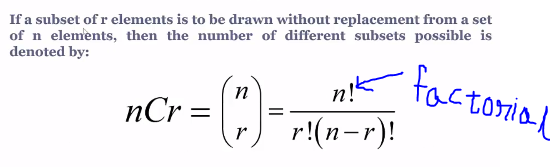
The formula is 12C3, where 12 is total size and 3 is combination/subset values.



Rules –

* If dataset contains subset then order of values w within subset will be ignored. For example, in given set is {1,2,3,{4,5},{5,4}}. Here subsets have same values in different order that will consider as identical subset.
* Order of value is not matters
* There are no duplicate or repeated value not exist.

Formula to get probability for set with subset-



**Permutation Probability:** In permutation the order of variable matters that mean {A,B,C} is different than {C,B,A}

If we are calculating Probability of multiple events together then its notation is

* P(AUB) – Probability of either event A or event B
* P(AᴖB) – Probability of A and B both together.

**Addition of two Probabilities:** In order to add two probabilities, we must deduct the permutations from overlap chances.

For example, Out of 200 students, 140 are from psychology(where 90 female students), 10 are from methamatics(3 female) and 50 from biology(28 female). If one these students is randomly selected, what is the Probability the student is from Psychology or a female P(AUB)?

* P of Total Female students are = 90+3+28 = 121 so the P(121/200)
* P of Total Psychology students are – 140(Including female students) so the P(140/200)
* P of Female student already covered in Psychology – P(90/200)
* The addition formula – P(121/200)+P(140/200) – P(90/200) = 0.855

**Probability dependency between events-**

**Mutually Exclusive Events –** The events which cannot occur simultaneously like student present on time or late on a day. But events have influence on each other. Either he can On-time or late so they are exclusive and Late has influence on On-time as he is not on-time. Here both events are mutually exclusive hence P(AᴖB)=0.

**Dependent Variable** – If event A has influence on event B. For example, Mathematics students and female students.

**Independent Variable** – Independent events without influence on each other. For example, toss the coin and roll the die together have no influence on each other.

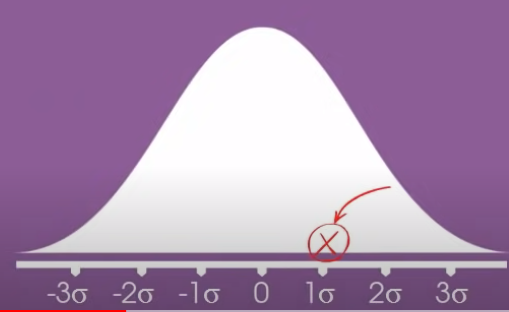
**Standard Deviation(σ) –** Standard Deviation is the distribution of the data. Lower STD means data values are close to average. Higher STD means data is well spreaded or scattered.

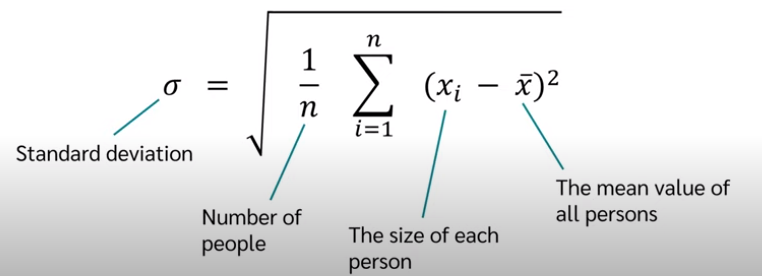
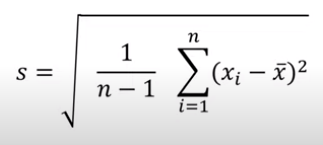
Formula – Root(Square root) of the sum of the squared deviations.

There are two type of Std –

Average Std – Sum of each element diff. from mean value, devide by number of values.

Quadratic Mean Std -



**Modal**

**Sum**

**Variance**

**Range**

**Quartile 1**

**Quartile 2**

**Quartile 3**

**Skew**

**Kurtosis**

**95% Confidence interval of mean**

**Mean+- Std**

**Test for Normal Distribution**

https://datatab.net/statistics-calculator/descriptive-statistics

###### Algorithm Explaination –

---------------------------------------------Statical Foundation END-----------------------------------------------

------------------------------Data Analysis-----------------------------

#### Data Analysis

##### Feature/Variable Types –

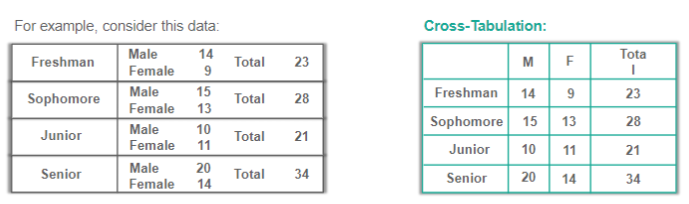
1. Categorical - as some specific attribute value like product color, size etc.
   1. Categorical have two type -
      1. Ordinal - Ordinal mean if values can ordered like small size , large size or Academic Degree – Bachelor, Master etc..
      2. Nominal - Nomial means- the attribute than can't ordered like color of product, Gender(Male,Female)
2. Continues - it can be anything like price of product, Age, Credit Limit

##### Exploratory Data Analysis –

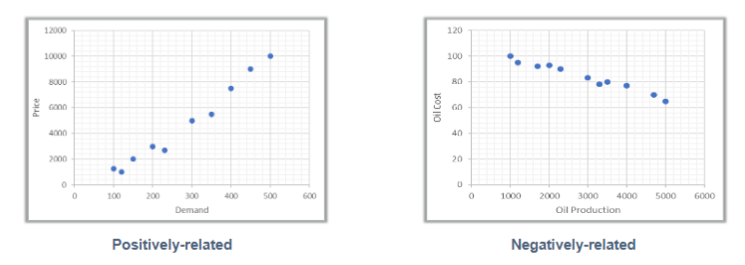
* + Exploratory Data Analysis
  + Steps –
    - * Identify variables like predictors/inputs and output/target. Categorize them among continues and Categorical variables.
      * Fix Nulls, blanks, fix data types and error values. Check in source, Delete them or fill them using mean, median, KNN Imputation, Linear.
      * Analysie input variables Individually(Univereable) and inter dependent(BiVariables)
      * Identify outliers and fix them. This could Data Entry/source/Processing or natural error.
      * Feature Engineering - Variable Transformation and Creation
      * Repeat steps 4 to 6 until get correct model
    1. Univariate way(Observer Individual Column) – Like weight column have average 60kg but there is one value having 40kg.
       - Graphical Methods – Analyze data using graphs like Bar, Histogram, Line, Box and Dot plots.
       - Non-Graphical Method – A simple table presenting Group count and frequency of each category. Eg.

|  |  |  |
| --- | --- | --- |
|  | **Group Count** | **Frequency (%)** |
| Red Pens | 30 | 75 |
| Blue Pens | 10 | 25 |
| **Total** | **40** | **100** |

* + 1. Multivariate way(Observer all or group of columns together) – Now check outliers found from Univariate with other fields. Eg. The 40Kg row might be for a Kid or Female and other rows are for Male.
* Multivariate Non-Graphical EDA –
  + Cross-tabulation – For categorical data with few measures, it can use to analyze data. Eg.



* + Calculate Correlation & Covariance – It helps to analyze linear relationship in variables. Eg.

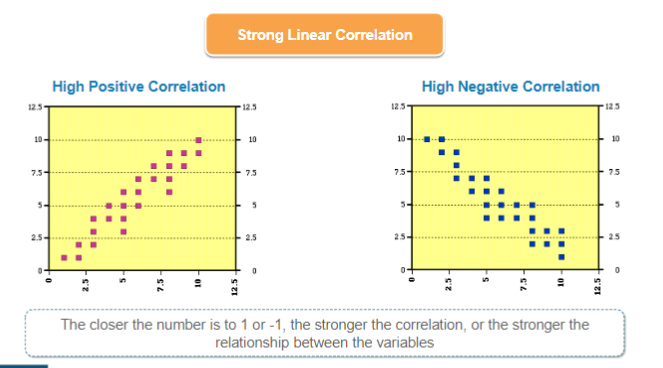


|  |  |
| --- | --- |
| Co-variance | Correlation/Association |
| It indicates whether variables are positively or negatively related. | It indicates the degree to which the variables are related |
| It indicates direction | It measures both strength and direction of linear relationship |
| It has dimensions | It is dimensionless unit |
|  | It can achieved by dividing the two variables by the sum of their standard differences. It depend on co-variance. |
| Python code = print(“Covariance:”,X.cov(Y)) | Eg. Print(‘Correlation:’, X.corr(Y)) |
|  |  |

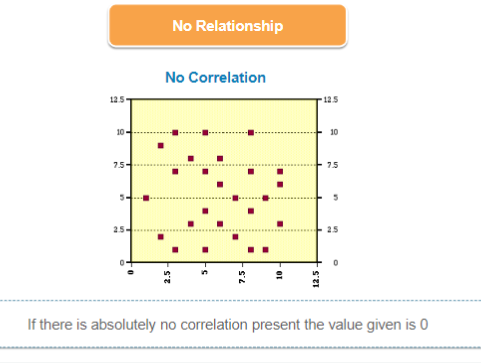
**Causation –** It means change in one measure will impact other measure. For eg. Increase in Cigarette sales will impact Cancer cases.

* + - * **Correlation Matrix –** Correlation matrix is a matrix in which the i-j position defines the correlation between the ith & jth parameter of the dataset.
        + The value ® rovides the strength & direction of association.
        + R has value between -1 to 1
        + Negative value(-1) indicates Negative association and 1 for Positive
        + If R=1 means strong relationship.
* Multivariate Graphical EDA- Visualize correlation sing Scatter or Heat maps.
  + Scatter – It shows Linear relationship. Below are presentation of various relationship type –

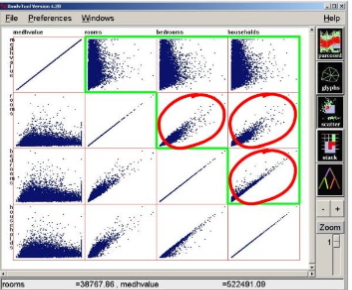






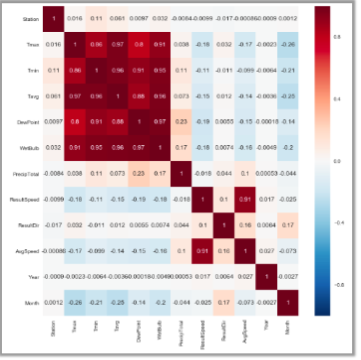


* Scatter Plot Matrix- It generates pair wise scatter plot in single page like rooms, household etc.



* Heatmaps – These visualize the collinearity of the data and help distinguish which rows/columns should/should not be included as part of result.

Darker Red indicates Strong Correlation and darker blue negative correlation.



Summary –

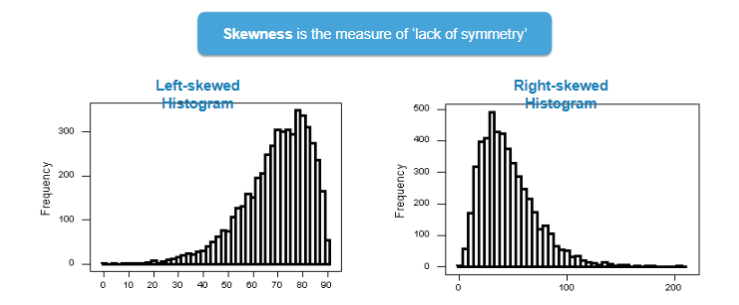
|  |  |  |
| --- | --- | --- |
| **Data type** | **Objective** | **EDA Technique** |
| Categorical | Distribution | Pie Charts, Bar Plots |
| Univariate Continuous | Distribution of variable, Finding Outlier | Line Plot, Histograms |
| BiVariate Continuous | Finding Outliers | 2D Scatter Plots |
| 2D Arrays | Relationship b/w two variables and one outcome variable | Heat Maps |
| Multiple Groups | Finding Outliers | Side-by-side Box Plots |

* + Classical Statistical Data Analysis Methods(Quantitative methods)
    - Interval Estimation
    - Hypothesis Tests

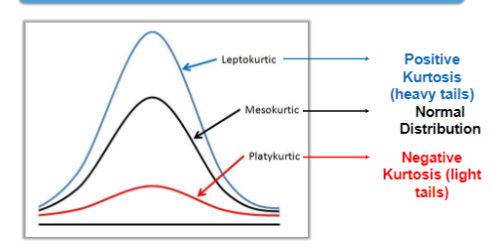
**Finding Outliers in measures -**

1. **Detect Skewness in measures** –

Skewness is the lack of symmetry. We can check it using histogram-



1. **Kurtosis –** Kurtosis is the measure of peaked-ness relative to Gaussian shape. Basically finding extreme high or very low or negative values in data.

****

**Detecting Outliers in Continues Variables-**

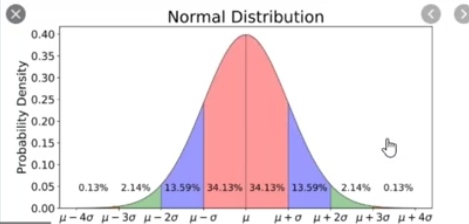
* Mean, Median and Mode
* IQR – Values beyond range IQR\*1.5 to IQR\*-1.5
* Negatives
* ZScore
* BoxPlot

Describe() the numerical features and observe such as huge diff. between figures like Max() and Avg|() or 75%.

Now check data in columns if it is Normally distributed or not. If yes then More or Less normally distributed?

Normal Distribution/Probability Distribution-

* It is always have Bell Curve.
* It is continues probability
* Data is symmetric meaning middle line is mean and both side are mirror image.
* The middle value is the Mean, Median and Mode(the highest vale)
* If Normal distribution chart shows it is not normal variable then plot boxplot. Then from box apply Inter Quantile(IQRL) method to cap the outliers and make it normal distribution.

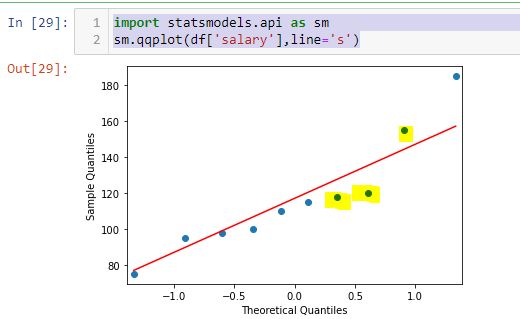


* In Normal distribution STD, the observation beyond Mean+3 are outliers.
* Plot sns.distplot(df['salary'],kde=True).

**QQPlot/Quantile plot -** It plot chart based on quantiles and plot line for that. If values(dots) are variant from line meaning are outliers.

import statsmodels.api as sm

sm.qqplot(df['salary'],line='s')



* **BoxPlot –** It plot values in ascending order and tried to plot box for that until values are in normal range. But has there is huge variance the values are plotted far from box and box is stopped there. Al values far from box are outliers.

Observe it all outline are single on making cluster or multiple outliers are together.

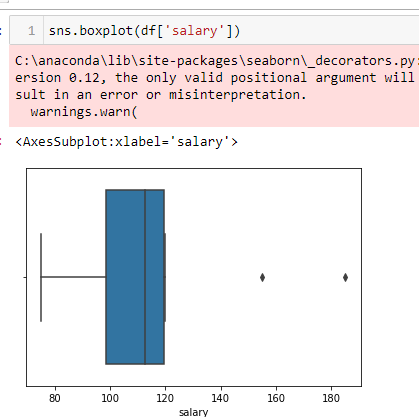
It splits data points into quartiles and check in which quartile the maximum values reside. Then maximum quartiles are make a box and rest are outliners out of box.

In box, the last line shows upper limit and first line minumum limit and the middle line is the median. All points beyond upper or lower lines are outliers. Lines calculation is below-

iqrl = np.percentile(df['salary'],75) - np.percentile(df['salary'],25)

upperlimit = np.percentile(df['salary'],75) + 1.5\*iqrl

minlimit = np.percentile(df['salary'],25) - 1.5\*iqrl

****

**Actions on outliers -**

* If there are very few outliers, we can drop those.
* As per boxplot, If there are many values(cluster) close to box range line then set max cap using inter quartile range(IQR) – IQR = 75th parcentile(Q3 – 25th Percentile(Q1)

iqrl = np.percentile(df['salary'],75) - np.percentile(df['salary'],25)

print(np.percentile(df['salary'],75))

(np.percentile(df['salary'],25)

upperlimit = np.percentile(df['salary'],75) + 1.5\*iqrl

df.loc[df['salary']>upperlimit,'salary'] = upperlimit

After appllying IQRL method, plot normal distribution chart and check, if still it is not normal distributed then apply ZScore.

* Apply Z-Score>3 or Mean+-3sd –

ZScore formula is = value-mean/3rd standards deviation or <-3 SD

Code : upperlimit = df['salary'].mean() + (3\*df['salary'].std())

lowerlimit = df['salary'].mean() - (3\*df['salary'].std())

df.loc[df['salary']>upperlimit,'salary'] = upperlimit

df.loc[df['salary']<lowerlimit,'salary'] = lowerlimit

* **BiVariate Continuous variables explortory–** To find relation between multiple variables. First plot correlation matrix to find relation among variables.
  + Correlation Coefficient is the strength of linear relation. Non Linear relation called as Negative correlation. Correlation Coefficient is always +1 to -1. If values is -1 then highly correlation in negative dirction similarly 1 is positive relation.
  + Formula= (N\*SUM(X\*Y)) – SUM(X)\*SUM(Y)/SQRT((N\*SUM(X^2) – (SUM(X)^2)) \* (N\*(SUM(Y^2) – (SUM(Y)^2)))
  + Correlation can be positive or negative
  + If the two variables have positive correlation then use anyone variable instead of taking both variables. It is called multicolinearity.
  + To find correlation among all variables(Correlation matrix) - df.corr()
  + We can get correlation matrix in graphical heatmap. - sns.heatmap(df.corr()). Where light color in red is strong correlation to dark blue color means weak relationship.
  + Better we should copy or populate tabular correlation matrix to excel using (df.corr().to\_csv('correlationmatrix.csv')) and replace colors based on below matrix.
  + If Correlation Coefficient values meaning –

|  |  |
| --- | --- |
| Correlation Coefficient | Meaning |
| >0.3 | Medium Strong Corr |
| >0.5 | Strong Correlation |
| >0.7 | Very Strong C orrelation |
| <0.3 | No Correlation |

* + - 1. First replace diagonal 1 t 0.
      2. Round values to 2 decimal points.
      3. Create rule->Format>If cell value greater than 0.3 then mark it Green.
      4. Rule - <-0.3, mark it orange.
      5. Now all green are correlated are positively and orange are negative correlated. I can consider only one or few columns as all these are linear correlated.

**Categorical Variables Relation** : - These are non-numerical fields like CreditCard Type, Gender etc.

Use Group by and find parent->Child order. Eg. Df.groupby([‘CreditCardType’,’Gender’]).count()

If count is symmetric then there is no much relation between fields. Usuall we use Pie, Donut or Stacked Bar chart with count or count% for this. Chi-Squar test can be used to find relatinship.

--------------------Feature Selection----------------

##### Feature selection

Feature Selecition is the method to identify features from given datase variables. Usually we receive huge datasets with lot of information. We need to identify the input data only using some methods. It helps into improve model accuracy, Low computational cost and easy to understand, explain and maintenance.

There are 3 methods for feature selection –

1. Correlation
2. Univariate testing
3. Recursive Feature Elimination with Cross-Validation (RFECV)

Cmap – mapping from data values to color space

Annot – bool or rectangular dataset, if true means write data value in each cell

1. Correlation - Correlation is the way to identify relationship among features/variables. Mostly the heatmap is used to identify correlation. It is powerfull visual method to get data pattern relation from huge datasets.

It helps into features selection.

Code eg.

import pandas as pd

import seaborn as sns

import numpy as np

import matplotlib.pyplot as plt

dfbs = pd.read\_csv('./class7/BostonHousing.csv')

dfbs.head(10)

x = dfbs.iloc[:,0:13]

y = dfbs["medv"]

names = []

#creating a correlation matrix

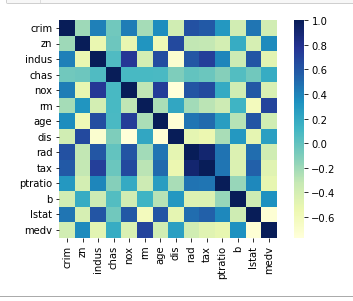
correlations = dfbs.corr()

sns.heatmap(correlations,square = True, cmap = "YlGnBu")

plt.yticks(rotation=0)

plt.xticks(rotation=90)

plt.show()



**2. Univariate testing:**

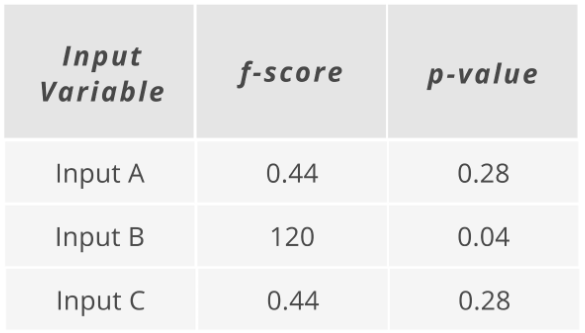
*Univariate Feature Selection*or*Testing* applies statistical tests to find relationships between the output variable and each input variable in isolation. Tests are conducted one input variable at a time. The tests depends whether you are running a regression task or a classification task.

The disadvantage of Univariate is that if there is any relationship within features then we will miss that as it check each variable independently.

An example output of these tests are tables showing relationships scores between each input variable and the output. *(\*see image below)*

**Regression Task**

In a regression task, you may be provided with an [*f-score*](https://deepai.org/machine-learning-glossary-and-terms/f-score) and a [*p-value*](https://www.youtube.com/watch?v=vemZtEM63GY) for each variable and gives you a view of the statistical significance of their relationships between the input and the output variables. This will help you assess how confident you should be with the variables you have used in your model.

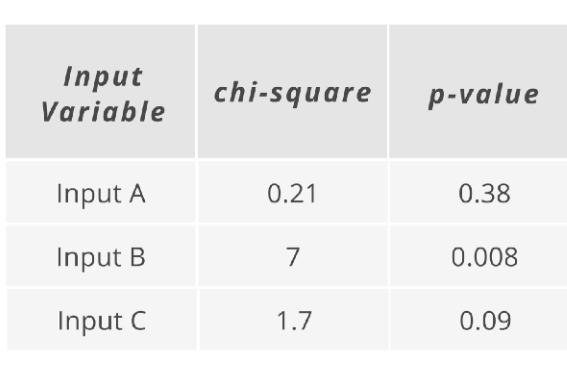


Code eg.

|  |
| --- |
| import pandas as pd |
|  |  |
|  | from sklearn.feature\_selection import SelectKBest, f\_regression |
|  |  |
|  | # import data |
|  | my\_df = pd.read\_csv("feature\_selection\_sample\_data.csv") |
|  |  |
|  | X = my\_df.drop(["output"], axis = 1) |
|  | y = my\_df["output"] |
|  |  |
|  | feature\_selector = SelectKBest(f\_regression, k = "all") |
|  | fit = feature\_selector.fit(X,y) |
|  |  |
|  | p\_values = pd.DataFrame(fit.pvalues\_) |
|  | scores = pd.DataFrame(fit.scores\_) |
|  | input\_variable\_names = pd.DataFrame(X.columns) |
|  | summary\_stats = pd.concat([input\_variable\_names, p\_values, scores], axis = 1) |
|  | summary\_stats.columns = ["input\_variable", "p\_value", "f\_score"] |
|  | summary\_stats.sort\_values(by = "p\_value", inplace = True) |
|  |  |
|  | p\_value\_threshold = 0.05 |
|  | score\_threshold = 5 |
|  |  |
|  | selected\_variables = summary\_stats.loc[(summary\_stats["f\_score"] >= score\_threshold) & |
|  | (summary\_stats["p\_value"] <= p\_value\_threshold)] |
|  | selected\_variables = selected\_variables["input\_variable"].tolist() |
|  | X\_new = X[selected\_variables] |

**Classification Task**

Depending on what test you use, you might be provided a [*chi-square score*](https://www.statisticshowto.com/probability-and-statistics/chi-square/) and a [*p-value*](https://www.youtube.com/watch?v=vemZtEM63GY) for each variable. Again, this gives you a view of the statistical significance of their relationships between the input variables and the output variables.



Classification task code eg.

|  |
| --- |
| # import packages |
|  | import pandas as pd |
|  |  |
|  | # import data |
|  | my\_df = pd.read\_csv("feature\_selection\_sample\_data.csv") |
|  |  |
|  | from sklearn.feature\_selection import SelectKBest, chi2 |
|  |  |
|  | X = my\_df.drop(["output"], axis = 1) |
|  | y = my\_df["output"] |
|  |  |
|  | feature\_selector = SelectKBest(chi2, k = "all") |
|  | fit = feature\_selector.fit(X,y) |
|  |  |
|  | p\_values = pd.DataFrame(fit.pvalues\_) |
|  | scores = pd.DataFrame(fit.scores\_) |
|  | input\_variable\_names = pd.DataFrame(X.columns) |
|  | summary\_stats = pd.concat([input\_variable\_names, p\_values, scores], axis = 1) |
|  | summary\_stats.columns = ["input\_variable", "p\_value", "chi2\_score"] |
|  | summary\_stats.sort\_values(by = "p\_value", inplace = True) |
|  |  |
|  | p\_value\_threshold = 0.05 |
|  | score\_threshold = 5 |
|  |  |
|  | selected\_variables = summary\_stats.loc[(summary\_stats["chi2\_score"] >= score\_threshold) & |
|  | (summary\_stats["p\_value"] <= p\_value\_threshold)] |
|  | selected\_variables = selected\_variables["input\_variable"].tolist() |
|  | X\_new = X[selected\_variables] |

**3-Recursive Feature Elimination with Cross-Validation *(RFECV)***

*Recursive Feature Elimination* fits a model that starts with all the input variables, then iteratively removes those with the weakest relationship with the output until the desired number of features is reached. It actually fits a model instead of just running statistical tests unlike the Univariate Testing.

“RFE is popular because it is easy to configure and use and because it is effective at selecting those features in a training dataset that are more or most relevant in predicting the target variable.”

The *CV* in RFECV means *Cross-Validation*. It gives you a better understanding on what the variables will be included in your model.

In the Cross-Validation part, it splits the data into different chunks and iteratively trains and validates models on each chunks separately. This simply means that each time you assess different models with certain variables included or eliminated, the algorithm also knows how accurate each model was from the model scenarios that are created and can determine which provided the best accuracy and concludes the best set of input variables to use.

Code eg.

|  |
| --- |
| # import packages |
|  | import pandas as pd |
|  |  |
|  | # import data |
|  | my\_df = pd.read\_csv("feature\_selection\_sample\_data.csv") |
|  |  |
|  | from sklearn.feature\_selection import RFECV |
|  | from sklearn.linear\_model import LinearRegression |
|  | import matplotlib.pyplot as plt |
|  |  |
|  | X = my\_df.drop(["output"], axis = 1) |
|  | y = my\_df["output"] |
|  |  |
|  | regressor = LinearRegression() |
|  | feature\_selector = RFECV(regressor) |
|  |  |
|  | fit = feature\_selector.fit(X,y) |
|  |  |
|  | optimal\_feature\_count = feature\_selector.n\_features\_ |
|  | print(f"Optimal numer of features: {optimal\_feature\_count}") |
|  |  |
|  | X\_new = X.loc[:, feature\_selector.get\_support()] |
|  |  |
|  | plt.plot(range(1, len(fit.grid\_scores\_) + 1), fit.grid\_scores\_, marker = "o") |
|  | plt.ylabel("Model Score") |
|  | plt.xlabel("Number of Features") |
|  | plt.title(f"Feature Selection using RFE \n Optimal number of features is {optimal\_feature\_count} (at score of {round(max(fit.grid\_scores\_), 4)})") |
|  | plt.tight\_layout() |
|  | plt.show() |

----------------------------End of feature selection

------------- Data set preparation(data wrangling, fill nan, join files, encrypt PII fields, replace category with numbers/enums) and Feature Engineering------------------

##### Data set preparation /Preprocessing

* Remove NA/Not Available value from dataframe using dropna function. Or fill blank values. Eg. df.dropna(inplace=True)
* Parse date as date column using pandas.to\_datetime(). Eg. df.index = pd.to\_datetime(df.index)
* Normalize or Scale data. Data normalization formula = (x-min(x)/(max(x)-min(x))
* Check duplicate dates or aggregate dataset to have unique dates.
* Set date as index using set\_index. pd.read\_csv('airline\_passengers.csv',index\_col='Month') or df.set\_index('Month')
* Sort data frame by Date/Index. Command - df.sort\_index(inplace=True)

**Binning –** Grouping numerical data

**Label Encoding -** mean give numeric value to label or data value. Eg. Tshirt small size = 1, medium =2 and large=3. This is part of feature engineering. This can done using Map() function or sklearn-

Method 1 – Using sklearn.LabelEncoder –

Step 1 - import library and module

from sklearn.prerocessing import LabelEncoder

#step2 - create instance of class

le = LabelEncoder()

#step3 = copy features to X. pls mind values in end as it will create an array instead of pandas frame

X=df[['buying','maint','safety']].values

#step 4 – Loop through all the features in array and encode all values

for i in range(len(X[0])):

X[:,i] = Le.fit\_transform(X[:,i])

Method2: Using Dictionary and Map function

#Map() eg.

size\_mapping={'s':1,'m':2,'L':3}

df['size']=df['size'].map(size\_mapping)

**Hot\_encoding/one hot encoding-** it is part of data pre-processing. It mean unpivot data having boolean value. It will populate columns like Red,Yellow, Green for color and each will have 0 or 1 for row.

Df\_cat=pd.getdummies(df\_cat['color','size','price']

----------------------Data set preparation(data wrangling, fill nan, join files, encrypt PII fields, replace category with numbers/enums) END ---------------------

**Feature Engineering**

Tranform variables - from non-linear to linear using Logarithm, Square/Cube root or binning.

Create new variables - Like derive Weeks, Day from Date. Create dummy variables to replace character/string to numbers like 0 to Male and 1 for Female.

------------------Data Smoothing Technique -----------------

##### Data Scaling-

Min-Max Scale

Z-Score

##### Data Smoothing Technique

HP Filter - The Hodrick-Prescott (HP) filter refers to a data-smoothing technique. The HP filter is commonly applied during analysis to remove short-term fluctuations associated with the business cycle. Removal of these short-term fluctuations reveals long-term trends.

------------------Data Smoothing Technique END -----------------

---------------Data Standardization and Normalization-----------------------------

##### Data Standardization and Normalization

|  |  |
| --- | --- |
| **Standardization vs Normalization** | |
| **Standardization** | **Normalization** |
| If your data is already distributed(Gaussian distribution) | data is not distributed I Guassian distribution |
| When you have outlier | When you don't or minimum have outlier. |
| It is not bound by range | It scales within [0,1] or [1,-1] |
| If algorithm needs assumption then standardization is good. | The algorithm does not need assumption then Normalization is good |
| Formula= x-mean(x column)/stddev(x) |  |
| frm sklearn.preprocessing import StandarScalerstd = StandardScaler() x=df.fit\_tranform(df[['Sciense','Math']]) |  |

---------------Data Standardization and Normalization END-----------------------------

----------Dimensionality reduction-------------------------

#### Dimensionality Reduction

Over the time as data keeps growing we get more and more dimensions to add into ML alogorithm. Reducing these dimensions with maintaining accuracy is called dimension reduction.

It can resolve using below two methods-

Two methods- PCA and LDA

###### Principal Component Analysis(PCA)

It convert data from multidimension to normalize(vertical) model. – It is unsupervised. Principal Component Analysis – In PCA, we check variance for each dimension and check if we are loosing important data by removing the dimension.

Steps:

1. Order all features based on Variance in descending order
2. Give first dimension is PC1 and second as Pc2.
3. Find relation between PC1 and PC2
4. Now check if we remove PC1 or PC2, what percent of information I am loosing. If loss of information is very less then we can drop that feature.

Eg. Code –

From sklearn.decomposition import PCA

Pca = PCA(n\_components=2)

Pca.fit(existing\_df)

Existing\_2d = pca.tranform(existing\_df)

Existing\_df\_2d = pd.DataFrame(existing\_2d)

Existing\_df\_2d.index = existing\_df.index

Existing\_df\_2d.columns = [‘PC1’,’PC2’]

Existing\_df\_2d.head()

Print(pca.explained\_variance\_ratio\_)

This code will return two values. First PC already explains variance and the second one says another of variance. So total variance is first+second.

###### Linear Discrimination Analysis (LDA) –

It is useful for multiclass data LDA(Linear Discrimination Analysis) – It is supervised. It can used for Preprocessing dimensionality reduction and Classification both.

It create separator line and group the data variance in classes. Then will apply formulat –

Separation (S)= Between Group Variance/Within Group variance

(VarA + VarB)/2.

### Matrics-

---------------------Decision Tree - Beginning ---------------

#### Decision Tree –

It has root node, sub-root, branches then leaf or terminal nodes.

Pure subset means there is always one value in decision. For eg. I below if Cloudy then it’s always play so no need to check for other features like humidity, temperature or wind.

If there is pure subset, do not split feature further.

**Most Homogeneous subnode**– it means highest confidence feature node to take the decision. In decision tree, the algorithm will calculate all the features along with result count. For eg. If Rainy then confidence is 50% and Cloudy has 100% confidence so Cloudy feature is most homogeneous feature.

In the same vein, Rainy has two branches i.e. Slow wind and Storm wind. If Slow wind has most chances of play but Storm has less chances of play so Slow Wind is most homogeneous.

**Entropy** – Entropy is method to quantify the dataset for homogeneity. Entropy = 0 is good and 1 is bad. It measures the data disorder.

It is 0 if data is disorder and more chances to identify the most homogeneous feature.. It have value b/w 0 t 1.

Information Gain = 1-entropy. Highest Information gain is best to select.

Entropy formula H(s) = -p1lnp1 – p2lnp2 – p3lnp ….

P is Probability here. P2 is second probabiliy.for example if weather is cloudy then kids will play outside 9 times and 5 time not play.

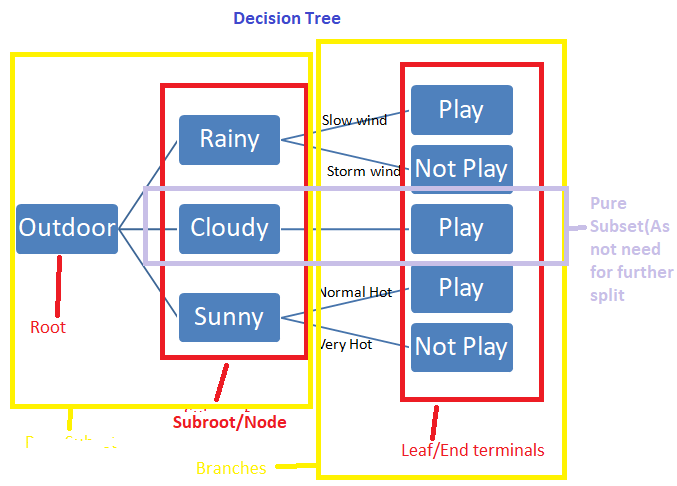
P1 is Yes(Will play) and p2 is No(Not play)

=(-pYes ln pYes) –( pNO ln PNo)

=(-9/14) ln (9/14) – (5/14) ln (5/14)

=0.94

Entropy of Parent Node

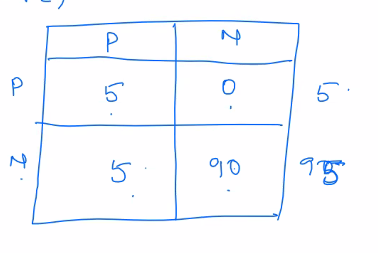


---------------------Decision tree - End ---------------

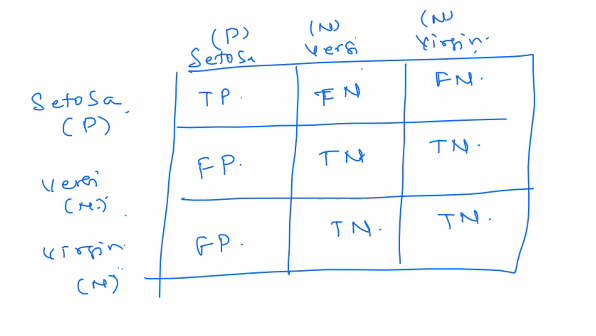
------------------------Confusion Matrix -------------------

##### Confusion Matrix

In Confusion matrix



If out class of interest(means we are interested only in one class usuall first class i.e Setosa in below) then –



Here Predicted values are on columns and Actual values are on Rows and the class of interest (usually first class) is Positive and others class are negative. Always take Predict as base for first character i.e True and False and then second letter is predicted class.

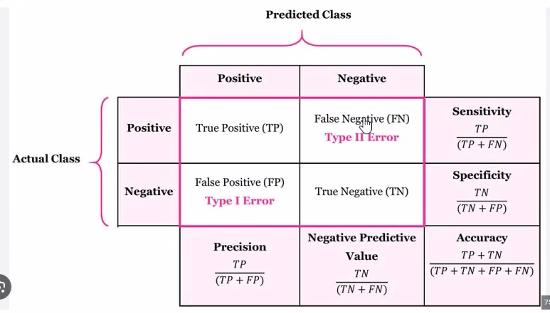
Actual Positive = TP+FN

Actual Negative = TN+FP

Predicted Positive = TP+FP

Predicted Negative = FN+TN

Accuracy =(TP+TN)/(TP+FN+TN+FP)



Code eg.

Cnf\_matrix\_gnb = confusion\_matrix(y\_test,y\_pred\_gnb)

Print(cnf\_matrix\_gnb)

----------------------Confusion Matrix End------------------

----------------------Covariance Matrix------------------

**Covariance Matrix**

----------------------Covariance Matrix ENd------------------

--------------Association Rule Mining --------------------------------------------

#### Association Rule Mining/Market Basket Analysis –

It is method to finding relation between variables in large datasets. It is like grouping items together with some context like while buying keyboard customer usually buy mouse as well.

It has 3 parameters -

1. Support = No. of times item X occurred/Total number of transactions. 600/1000
2. Confidence = No. of times item X&Y occurred/total occurrences of X. Support/X
3. Lift = No. of times item X & Y occurred/ Total occurrence of X multiplied by Total Occurrence of Y. X+Y/X^2\*y^2

Eg. 10 customer bought items as below

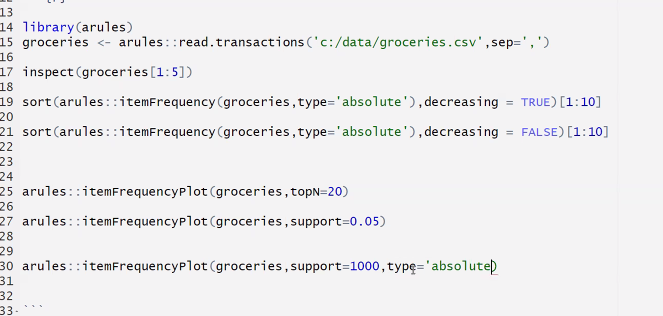
10 Milk and 8 butter and 6 Milk and Butter both

Support - A+B / Total Transactions

Confidence – (both/butter) or (confidence/Seconditem) 6/8 = 75%

Lift - both/10 or support/Milk = 75%

This is more supported by R. Below is the code-



--------------Association rule mining End --------------------------------------------

-------------------Recommendation Engines-------------

#### Types of Recommendation Engine

1. Collaborative or User Based Filtering – The recommendation will based on user’s previous history data and decision taken by other users.

Algorithms-

* KNN (K Nearest Neighbour)
* Pearson Correlation

It has limitations like user’s history and other users decisions data is required hence it won’t work for new uers.

1. Content-Based Filtering – It check other items to recommend with similar characteristics.

It keep existing content/data into centre and keep suggesting user based on his ratings, history etc and engine accuracy will keep growing.

1. Hybrid – Combination of both.

-------------------Recommendation Engines END-------------

------------------------------Data Analysis END-----------------------------

------------------------------------------------------Algorithms------beginning –

#### Algorithms

--------------------------Trend Line Algorithms -----------------------

--------------------------------Logarithmic------------------

#### Logarithmic

Logarithmic is represent the rate of change between variables which increase and decrease quickly but gradually levels out. It should avoided when there are negative values in column otherwise that values will be ignored.



Ln(X) is natural logarithm of X

----------------------------Logarithmic end-----------------------

----------------------------Exponential -----------------------

##### Exponential

It is good when trend is upward. Exponential Trend lines are lines that represent rise in the rate of changes between variables at increasingly higher rates.



It first transform the dependent variable by natural logarithm ln(Y) then estimate the model. It filter out the negative values.

----------------------------Exponential END -----------------------

----------------------------Polynomial -----------------------

Polynomial

It deals with fluctuations. It order is decided by observing number of fluctuatins or bends in the data.



Here last term in he sequence has an exponent equal to the order

----------------------------Polynomial END-----------------------

--------------------------Trend Line Algorithms END -----------------------

##### -----------Apriori Algorithm ---------------

It algorithm to identify items together(item set). It checks item subsets within item sets. It focuses on Support from Association Rule Mining/Market Basket Analysis.

Eg. Customer buys Keyboard and mouse together.

Steps -

1. it will find the maximum item set bought together. For eg. 4 items bought together(item set)
2. Remove items which are not frequently bought within itemsets for further computation.
3. Third Second it will check less items(subset) bought together within sets.
4. Repeat above steps till we found most frequent subset.

import pandas as pd

from mlxtend.frequent\_patterns import apriori

from mlxtend.frequent\_patterns import association\_rules

df = pd.read\_excel('Online\_Retail.xlsx')

df.head()

df['Description'] = df['Description'].str.strip()

df.dropna(axis=0, subset=['InvoiceNo'], inplace=True)

df['InvoiceNo'] = df['InvoiceNo'].astype('str')

df = df[~df['InvoiceNo'].str.contains('C')]

df

basket = (df[df['Country'] =="France"]

.groupby(['InvoiceNo', 'Description'])['Quantity']

.sum().unstack().reset\_index().fillna(0)

.set\_index('InvoiceNo'))

Basket

def encode\_units(x):

if x <= 0:

return 0

if x >= 1:

return 1

basket\_sets = basket.applymap(encode\_units)

basket\_sets.drop('POSTAGE', inplace=True, axis=1)

basket\_sets

frequent\_itemsets = apriori(basket\_sets, min\_support=0.07, use\_colnames=True)

rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=1)

rules.head()

rules[ (rules['lift'] >= 6) &

(rules['confidence'] >= 0.8) ]

--------------------Apriori algorithm end---------------------------------

---------------Naïve Byes Classifier----------------------------------

#### Naïve Byes Classifier

This is supervised learning model classifier algorithm. It consider that every variable is independent and there is no relation between them. It check the percentage of each variable on target and sum the scores for each class from each variable output. The highest score would be the result.

For example, let say if there are 3 features to classify if kids will play outside or not.

Feature1 – Wind – If Fast then no play. If slow then play

Feature2 - Sunlight – If Hot then not play. If Normal then will play

Feature3 – Rain – If heavy then play. If normal then will play.

The algorithm will calculate the percent of Play and Not play for each feature then sum if for Play and Not play. Higher score will win.

Code eg.

From sklearn.naive bayes import GaussianNB

Gnb = GaussianNB()

Y\_pred\_gnb = gnb.fit(X\_train, y\_train).predict(X\_test)

---------------Naïve Byes Classifier END----------------------------------

------------------ K-Armed Bandit -----------------------------

#### Armed Bandit

It is Reinforcement ML. It means take decision randomly and learn. Hence it is not good model.

------------------ K-Armed Bandit  END-----------------------------

---------------The Epsilon Greedy-----------------------------------

#### The Epsilon Greedy

It is Reinforcement Algorithm.

It has exploration and exploitation.

---------------The Epsilon Greedy END-----------------------------------

---------------Markov Decision Process(MDP)-----------------------------------

##### Markov Decision Process(MDP)

It has policies – Policy to define reward on action. For eg. Small reward 0.01

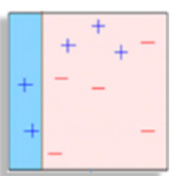
Discounting –

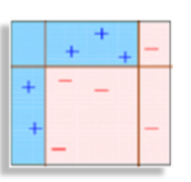
---------------Markov Decision Process(MDP END)-----------------------------------

----------------------- AdaBoost Classifier and KNN begin ------------------------------

##### AdaBoost Classifier details

1. This model decide one equal weight for all points then make a line for that and then classify data points as Positive or Negative based on their values.
2. At the end, all values must classify as positive and negative.
3. If values are not classified correctly then another equal value needs to add. These line can be made vertically or horizontally.
4. The lines will keep adding until the positive and negatives are classified.





#### KNN details -

* KNN(Nearest Neighbours Classification) Algorithm is classifier algorithm.
* It is supervised algorithm and part of Lazy Learning models family.
* It makes prediction of grouping/class of data point
* It can used for regression or classification tasks.
* If we have discrete class values then we use classification and there it does voting for new data point and most voted class/group is assigned to label.
* When it use with Continues values then we need regression KNN where it will take average of nearest neighbours to make prediction about classification.
* It stores all the training data and go through it on each prediction run hence it is memory intensive and time consuming(lazy) model. As data grows, its performance degrade.
* If data set is small then we should usually keep K as small number and for large dataset a large number.
* Keep K as odd number.
* It is commonly used for simple recommendation, class prediction, patter recognition, data mining , Financial market prediction and intrusion prediction.
* It basically calculates distance between numbers and set boundaries. K is the how many nearest data point needs to check. Hence, if data set is small then keep K small and vice versa. Higher K given more accuracy as it will take care of outliers but affect performance.
* As input, it require K and distance metric name like Euclidean distance, Minkowski, Manhattan distance, Minkowski distance, Hamming distance.

Advantages – Easy to implement, Adapt new data set quickly as it loads training dataset into memory and few hyper-parameters.

Tips:

* Check min/max for all features from describe command if there huge difference then apply scaling on features.
* Generally keep K value as odd/negative upto 20 or go upto square root of number of samples. For example if total samples are 426 then square root of 426\*\*0.5 = 20.
* Loop through 1 to 20 or sample 426\*\*0.5 with step 2 and different parameter like uniform or distance and get accuracy. Plot comparison line chart to get highest accuracy parameter combination to apply.
* if life science data or continuous data then use KNN

Disadvantages –

* Does not scale well, Memory intensive, Lazy Learning Model
* Curse of dimensionality – If there are many dimensions then accuracy and performance will degrade. Also, more dimensions with small observations will result classification error
* Overfitting/Underfitting – If K is not provided correctly then it could result overfitting or under fitting. High K = Underfit and Low K = Overfit

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics

df = pd.read\_csv ('mtcars\_for\_manymerge.csv')

x = df[['cyl','hp']]

y = df['Feedback']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,random\_state =5)

knn = KNeighborsClassifier(n\_neighbors=4)

knn.fit(x\_train,y\_train)

y\_pred = knn.predict(x\_test)

metrics.accuracy\_score(y\_test,y\_pred)

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn import metrics

df = pd.read\_csv ('mtcars\_for\_manymerge.csv')

x = df[['cyl','hp']]

y = df['Feedback']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,random\_state = 10)

knn = KNeighborsClassifier(n\_neighbors=4)

knn.fit(x\_train,y\_train)

y\_pred = knn.predict(x\_test)

metrics.accuracy\_score(y\_test,y\_pred)

from sklearn.model\_selection import cross\_val\_score

knn = KNeighborsClassifier(n\_neighbors=4)

print(cross\_val\_score(knn, x, y, cv=10, scoring ='accuracy').mean())

from sklearn.linear\_model import LogisticRegression

logreg = LogisticRegression()

print (cross\_val\_score(logreg, x, y, cv=10, scoring = 'accuracy').mean())

----------------------- AdaBoost Classifier end---------

----------------------------Linear regression Algorithms--------------------

##### Linear Regression Algorithms

Linear Regression analysis is a powerful technique used for predicting the unknown value of a variable(Dependent Variable/DV) from he known value of another variables (Independent Variable/IDV).

Linear forecasting is good when data trend is increasing or decreasing.

IDV - Input variables to train the model.

DV – Output variables to be predicted.

**Simple Linear Regression** – Y=a+bX

Lm.fit – Fits a linear model

Lm.predict – Predict Y using the linear model with estimated coeffciants

Lm.score =- Return the coefficient of determination(R^2)

L.Coef\_ = Estimated Coefficients

l.intercept\_ = Estimated intercepts

code eg.

import pandas as pd

import seaborn as sns

import numpy as np

import matplotlib.pyplot as plt

dfbs = pd.read\_csv('./class7/BostonHousing.csv')

dfbs.head(10)

x = dfbs.iloc[:,0:13]

y = dfbs["medv"]

names = []

#creating a correlation matrix

correlations = dfbs.corr()

sns.heatmap(correlations,square = True, cmap = "YlGnBu")

plt.yticks(rotation=0)

plt.xticks(rotation=90)

plt.show()

#split data into train and test

from sklearn.model\_selection import train\_test\_split

#testing data size is of 33% of entire data

x\_train, x\_test, y\_train, y\_test =train\_test\_split(x,y, test\_size = 0.33, random\_state =5)

#fitting our model to train and test

from sklearn.linear\_model import LinearRegression

lm = LinearRegression()

model = lm.fit(x\_train,y\_train)

#predict values

pred\_y = lm.predict(x\_test)

#plot prediction on scater chart

plt.scatter(y\_test,pred\_y)

plt.xlabel('Y Test')

plt.ylabel('Predicted Y')

----------------------------Linear regression Algorithms END--------------------

---------------Meansquare error----------------

#### Mean Square Error

It is useful to get the mean square error of predicted value and original values.

Code eg.

from sklearn.metrics import mean\_squared\_error

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

#split data

x = df.iloc[:,3:15]

y = df["rating"] #to be predict

x\_train, x\_test, y\_train, y\_test =train\_test\_split(x,y, test\_size = 0.33, random\_state =5)

#regression mean square error model

#fitting our model to train and test

lm = LinearRegression()

model = lm.fit(x\_train,y\_train)

#predicted rating(x) for test data

pred\_y = lm.predict(x\_test)

#get mean square error

mean\_squared\_error(y\_test, pred\_y)

---------------Meansquare error end----------------

---------------------Linear Algorithm - End---------------

### Logistic regression –

Used to estimate discrete values(0/1 or true/false) based n given set f indpendent variables. Y=C+B1X1+B2X2…

**Eg.code**

from sklearn.model\_selection import cross\_val\_score

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv), data manipulation as in SQL

import matplotlib.pyplot as plt # this is used for the plot the graph

import seaborn as sns # used for plot interactive graph. I like it most for plot

from sklearn.linear\_model import LogisticRegression # to apply the Logistic regression

from sklearn.model\_selection import train\_test\_split # to split the data into two parts

from sklearn import metrics # for the check the error and accuracy of the model

from sklearn.tree import DecisionTreeClassifier

# Any results you write to the current directory are saved as output.

# dont worry about the error if its not working then insteda of model\_selection we can use cross\_validation

data = pd.read\_csv("data.csv",header=0)

# here header 0 means the 0 th row is our coloumn name

# have a look at the data

print(data.head(6))# as u can see our data have imported and having 33 columns

# head is used for to see top 5 by default I used 2 so it will print 6 rows

# now lets look at the type of data we have. We can use

data.info()

# now we can drop this column Unnamed: 32

data.drop("Unnamed: 32",axis=1,inplace=True) # in this process this will change in our data itself

# if you want to save your old data then you can use below code

# data1=data.drop("Unnamed:32",axis=1)

# here axis 1 means we are droping the column

# here you can check the column has been droped

data.columns # this gives the column name which are persent in our data no Unnamed: 32 is not now there

# like this we also don't want the Id column for our analysis

data.drop("id",axis=1,inplace=True)

# As I said above the data can be divided into three parts.lets divide the features according to their category

# now as you know our diagnosis column is a object type so we can map it to integer value

data['diagnosis']=data['diagnosis'].map({'M':1,'B':0})

data.describe() # this will describe the all statistical function of our data

# lets get the frequency of cancer stages

sns.countplot(data['diagnosis'],label="Count")

plt.show()

corr = data.corr() # .corr is used for find corelation

plt.figure(figsize=(14,14))

sns.heatmap(corr, cbar = True, square = True,

cmap= 'coolwarm')

plt.show()

#Based on corrplot let's select some features for the model ( decision is made in order to remove collinearity)

prediction\_var = ['texture\_mean','perimeter\_mean','smoothness\_mean','compactness\_mean','symmetry\_mean']

# now these are the variables which will use for prediction

#now split our data into train and test

train, test = train\_test\_split(data, test\_size = 0.3)# in this our main data is splitted into train and test

# we can check their dimension

print(train.shape)

print(test.shape)

train\_X = train[prediction\_var]# taking the training data input

train\_y=train.diagnosis# This is output of our training data

# same we have to do for test

test\_X= test[prediction\_var] # taking test data inputs

test\_y =test.diagnosis #output value of test dat

logistic = LogisticRegression()

logistic.fit(train\_X,train\_y)

temp=logistic.predict(test\_X)

print(metrics.accuracy\_score(temp,test\_y)) # to check the accuracy

clf = DecisionTreeClassifier(random\_state=0)

cross\_val\_score(clf, train\_X, train\_y, cv=10)

clf.fit(train\_X,train\_y, sample\_weight=None, check\_input=True, X\_idx\_sorted=None)

clf.get\_params(deep=True)

clf.predict(test\_X, check\_input=True)

clf.predict\_log\_proba(test\_X)

clf.predict(test\_X,check\_input=True)

print(clf.score(test\_X,test\_y, sample\_weight=None))

**---------------Logistic regression end ------------------**

------------------RandomForest Classifier----------

### Random Forest

Random forest is an ensemble of decision trees. It gives better prediction and accuracy than decision tree. Decision tree runon each class and most vote decide the result. It provides better accuracy than decision tree. Good for large datasets. It can run on all the variables and then find which variables are important. It split data into subsets and run each subset on decision tree. After vote from each tree the majority will votes will be the decision.

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv), data manipulation as in SQL

import matplotlib

matplotlib.use("Agg")

%matplotlib inline

# used for plot interactive graph. I like it most for plot

import seaborn as sns # this is used for the plot the graph

from sklearn.model\_selection import train\_test\_split # to split the data into two parts

from sklearn import metrics # for the check the error and accuracy of the model

from sklearn import tree

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import accuracy\_score

# To import the svm classifier

import random

data = pd.read\_csv('pacific.csv')

print(data.head(6))

#print(data.info())

#data['Status'] = data['Status'].map({'TS':1,'HU':0})

#data.describe() # this will describe the all statistical function of our data

data.Status = pd.Categorical(data.Status)

data['Status'] = data.Status.cat.codes

print(data.head())

# lets get the frequency of different typhoons

#sns.countplot(data['Status'],label="Count")

#plt.show()

random.seed(2)

pred\_columns = data[:]

pred\_columns.drop(['Status'],axis=1,inplace=True)

pred\_columns.drop(['Event'],axis=1,inplace=True)

pred\_columns.drop(['Latitude'],axis=1,inplace=True)

pred\_columns.drop(['Longitude'],axis=1,inplace=True)

pred\_columns.drop(['ID'],axis=1,inplace=True)

pred\_columns.drop(['Name'],axis=1,inplace=True)

pred\_columns.drop(['Date'],axis=1,inplace=True)

pred\_columns.drop(['Time'],axis=1,inplace=True)

prediction\_var = pred\_columns.columns

print(list(prediction\_var))

# now these are the variables which will use for prediction

#now split our data into train and test

train, test = train\_test\_split(data, test\_size = 0.3)# in this our main data is splitted into train and test

# we can check their dimension

print(train.shape)

print(test.shape)

train\_X = train[prediction\_var]# taking the training data input

train\_y= train['Status']# This is output of our training data

#print(list(data.columns))

print(list(train.columns))

# same we have to do for test

test\_X= test[prediction\_var] # taking test data inputs

test\_y =test['Status'] #output value of test dat

v#RandomForest classifier

model=RandomForestClassifier(n\_estimators=100)# a simple random forest model

model.fit(train\_X,train\_y)# now fit our model for traiing data

prediction=model.predict(test\_X)# predict for the test data

#prediction will contain the predicted value by our model predicted values of diagnosis column for test inputs

print(metrics.accuracy\_score(prediction,test\_y)) # to check the accuracy

# here we will use accuracy measurement between our predicted value and our test output values

#Decision Tree

model = tree.DecisionTreeClassifier()

model.fit(train\_X,train\_y)# now fit our model for traiing data

prediction=model.predict(test\_X)# predict for the test data

# prediction will contain the predicted value by our model predicted values of diagnosis column for test inputs

df=pd.DataFrame(prediction,test\_y)

print(df)

print(metrics.accuracy\_score(prediction,test\_y)) # to check the accuracy

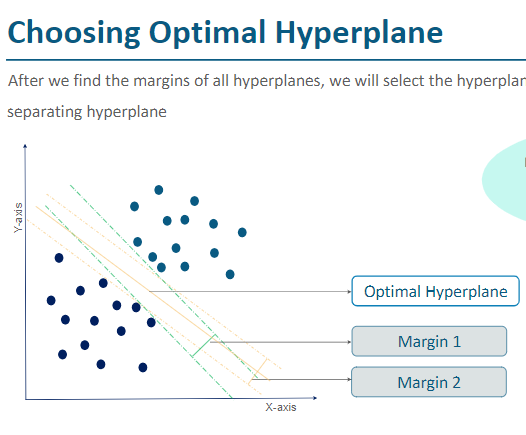
# here we will use accuracy measurement between our predicted value and our test output values

**------------------RandomForest Classifier END----------**

**---------------------------------Support Vector Machine----------------------**

#### Support Vector Machine

It is supervised machine learning algorithm which used for both Classification or Regression Challenges. It is concept of hyperplane line. It basically plo multiple lines between points to divide them equally then it calculate margin from line to closest data points. The largest margin line will be considered as HyperPlane line.



Code eg.

From sklearn import svm

From sklearn.metrics import accuracy\_score

Model = svm.svc(karnel=’linear’, c=1,gamma =1)

Model.fit(train\_X, train\_Y)

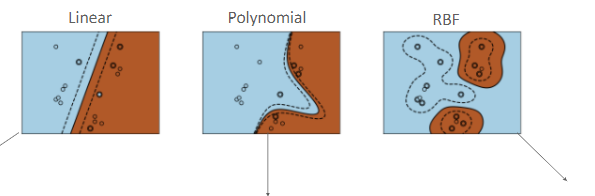
#Predict output

Predicted = model.predict(test\_X)

Print(“SVM accuracy:”, accuracy\_score(test\_y, predicted)

There are 3 types of kernels –

1. Linear – When data s linearly separable, we use linear kernel
2. Polynomial – When data is separable with curvy line.
3. RBF when data is not separable using linear or polynomial



‘C’ value determine the width of the margin, larger the C value smaller is the margin. It directly affects the misclassification error.

Recommended C value is 2^-10 to 2^10

Gamma is used for non-linear classification. When the data values are distributed in a way so we can plot Liner or Polynomial separator lines and we have to use RBF then Gamma is required.

**---------------------------------Support Vector Machine END----------------------**

### --------Clustering Algorithms-----------------

#### --------------------K-Mean------------------------

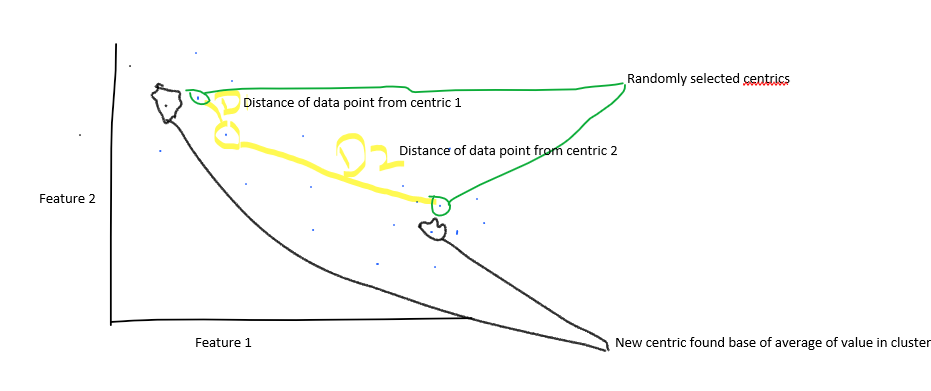
It is unsupervised machine learning clustering method. It is clustering method where data points are distributed amont clusters. K stands for number of clusters.

If centroids are not coming properly then it would be because of gap in data points. As solution, we need to normalize the data.

Steps –

|  |
| --- |
| 1. Define number of clusters(K). For eg. 2 is given. |
| 1. Algorithm will select randomly 2 data points(each for once cluster) called centrode from given dataset. |
| 1. It will calculate distance(d1 and d2) for data point from both the selected points(centrodes). |
| 1. it sill check which distance(d) is lower and accordinly data point will assign to that cluster. |
| 1. Now it will take average of data points on each cluster and set this as cluster centrode |
| 1. Now it will repeat the step from 3 to 5 until the mean or centrode is not changing/moving. |

Note: inertia is SSE(Sum of squared error)



Advantage –

1. Simple
2. Items are automatically assigned to cluster

Disadvantages –

1. Must define number of cluster
2. All items are forced into clusters
3. Unable to handle noisy data and outliers

Code eg.

import pandas as pd

import numpy as np

from matplotlib import pyplot as plt

from sklearn.cluster import KMeans

from sklearn.preprocessing import MinMaxScaler

df = pd.read\_csv('Income-1.csv')

df

plt.scatter(df['Age'], df['Income($)'])

plt.xlabel('Age')

plt.ylabel('Income($)')

km = KMeans(n\_clusters=3)

y\_predicted = km.fit\_predict(df[['Age', 'Income($)']])

y\_predicted

df.head()

df['Cluster'] = y\_predicted

km.cluster\_centers\_

df0 = df[df['Cluster'] == 0]

df1 = df[df['Cluster'] == 1]

df2 = df[df['Cluster'] == 2]

plt.scatter(df0['Age'], df0['Income($)'], color='green')

plt.scatter(df1['Age'], df1['Income($)'], color='red')

plt.scatter(df2['Age'], df2['Income($)'], color='blue')

plt.scatter(km.cluster\_centers\_[:,0],km.cluster\_centers\_[:,1], color='black', marker='\*', label='centroid')

plt.legend()

df0

#normalize my data

x\_n = (x - x\_min)/(x\_max-x\_min)

scaler = MinMaxScaler()

scaler.fit(df[['Income($)']])

df['Income($)'] = scaler.transform(df[['Income($)']])

scaler.fit(df[['Age']])

df['Age'] = scaler.transform(df[['Age']])

df.head()

plt.scatter(df.Age, df['Income($)'])

#the shape of the dataset is preserved

km = KMeans(n\_clusters=3)

y\_predicted = km.fit\_predict(df[['Age', 'Income($)']])

y\_predicted

df['Cluster'] = y\_predicted

km.cluster\_centers\_

df0 = df[df['Cluster'] == 0]

df1 = df[df['Cluster'] == 1]

df2 = df[df['Cluster'] == 2]

plt.scatter(df0['Age'], df0['Income($)'], color='green')

plt.scatter(df1['Age'], df1['Income($)'], color='red')

plt.scatter(df2['Age'], df2['Income($)'], color='blue')

plt.scatter(km.cluster\_centers\_[:,0],km.cluster\_centers\_[:,1], color='black', marker='\*', label='centroid')

plt.legend()

k\_rng = range(1,10)

sse = []

for k in k\_rng:

km = KMeans(n\_clusters=k)

km.fit(df[['Age', 'Income($)']])

sse.append(km.inertia\_)

plt.xlabel('K')

plt.ylabel('sse')

plt.plot(k\_rng, sse)

fit

predict

fit -> centroid

predict -> map the points to diff centroids based on distance

--------------------K-Mean END------------------------

------------------------------Elbow method----------------------------

------------------------------Elbow method END----------------------------

#### ------------------------------------Elbow method---------------------------

------------------------------------Elbow method END---------------------------

#### Hierarichal

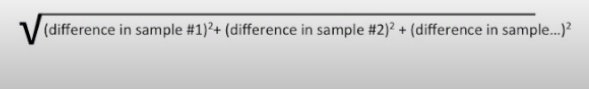
--------Clustering Algorithms END-----------------

--------------------Hierarchal Clustering ------------------

##### Hierarical Clustering

It is way to group the items into cluster based on their similarities. It will iterate over the all items and check for similarity. In first iteration, it will check first item similarity with all other items and club them in one cluster. Now it will consider this cluster as one item and will compare 2nd item with all other items and will cluster them accordingly. This process will keep going on until all items are clustered.

To identify similarities, it usually check distance from X. below is the formula –



Usually before hierarichal clustering, the are some data scaling methods like F1 score etc needs to apply to fix the outliers.

--------------------Hierarchal Clustering END------------------

------------------------------------------------------Algorithms END------

----------------------Time Series Analysis------------------

### Time Series Analysis –

Time series can be defined as a set of data dependent on time where time is independent variable to estimate other variables.

It has different types of trends-

1. Trend – Upward/Downward or horizantal
2. Cyclical Pattern
3. Seasonal
4. Irregular - It has white noise.

White Noise – Zero means constant variance. Autoregressive (AR) or Moving Average (MA) models used to correct noise.

----------------------------Autoregressive Integrated Model(ARIMA)--------------

##### ARIMA Autoregressive Integrated Model-

Auto regressive integrated Moving Average model.

It is good for Sales Estimation/future forecasting but does not work good with historical stock data.

It has two type - Non-Seasonal and Seasonal

Arima model does no work good on hisorical data.

Mostly it used for Non-Stationary data to find the variance to integrate into it and make it stationary.

----------------------------Autoregressive Integrated Model(ARIMA) END--------------

----------------------------Simple Moving Average --------------

##### Simple Moving Average(SMA)

Moving Average and is average with given window like last 6 months etc. We achieve it using rolling(window) function in Python. Eg. df['SMA for 6 Months'] = df['Thousands of Passengers'].rolling(window=6).mean()

----------------------------Simple Moving Average END--------------

---------------------------- EWMA(Exponentially weighted moving average) --------------

#### EWMA(Exponentially weighted moving average)

It give more weight to recent data. Based of recent data, it decide the parameter value to give it weight and then derive values. It is more useful for future estimations. Eg. df['Count of passenger'].ewm(span=12).mean()

---------------------------- EWMA(Exponentially weighted moving average) END --------------

----------------------------ETS(Error Trend Seasonability --------------

ETS Models will take each of those terms for smoothing and may add them, multiply them or even just leave of them out. We can analyze the data by decompose it by timeseries.

In simple terms, it check the data from different ways, first it take obsereved values i.e. actual value then compare it with diff. trends like seasonality, overal trend and last residual trend i.e. remaining values that are not part of any above trend.

in below example, the colde will return all 5 columns in object result.

eg. from statsmodels.tsa.seasonal import seasonal\_decompose

result = seasonal\_decompose(df['Thousands of Passengers'],model='additive')

result.plot(observed=True,

seasonal=True,

trend=True,

resid=True,

weights=False)

Code eg. decomp = seasonal\_decompose(df['Milk in Pounds per cow'],period=12)

fig = decomp.plot()

fig.set\_size\_inches(12,6)

----------------------------ETS(Error Trend Seasonability END --------------

-------------Dickey fuller test ---------------------------------------

##### Dickey Fuller

It is used to test whether the data is Stationary or Non-Stationary.

ACF and PACF help to determine p,d and q that will require for ARIMA model. These are basically lag within data.

import requests, pandas as pd, numpy as np

from pandas import DataFrame

from io import StringIO

import time, json

from datetime import date

from statsmodels.tsa.stattools import adfuller, acf, pacf

from statsmodels.tsa.arima\_model import ARIMA

from statsmodels.tsa.seasonal import seasonal\_decompose

from sklearn.metrics import mean\_squared\_error

import matplotlib.pylab as plt

%matplotlib inline

from matplotlib.pylab import rcParams

rcParams['figure.figsize'] = 15, 6

df\_fx\_data = pd.read\_csv('BOE-XUDLERD.csv')

df\_fx\_data

df\_fx\_data['Date'] = pd.to\_datetime(df\_fx\_data['Date'])

indexed\_df = df\_fx\_data.set\_index('Date')

ts = indexed\_df['Value']

ts.head(5)

plt.plot(ts)

ts\_week = ts.resample('W').mean()

plt.plot(ts\_week)

def test\_stationarity(timeseries):

#Determing rolling statistics

rolmean = timeseries.rolling(window=52,center=False).mean()

rolstd = timeseries.rolling(window=52,center=False).std()

#Plot rolling statistics:

orig = plt.plot(timeseries, color='blue',label='Original')

mean = plt.plot(rolmean, color='red', label='Rolling Mean')

std = plt.plot(rolstd, color='black', label = 'Rolling Std')

plt.legend(loc='best')

plt.title('Rolling Mean & Standard Deviation')

plt.show(block=False)

#Perform Dickey-Fuller test:

print('Results of Dickey-Fuller Test:')

dftest = adfuller(timeseries, autolag='AIC')

dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observations Used'])

for key,value in dftest[4].items():

dfoutput['Critical Value (%s)'%key] = value

print(dfoutput)

test\_stationarity(ts\_week)

ts\_week\_log = np.log(ts\_week)

ts\_week\_log\_diff = ts\_week\_log - ts\_week\_log.shift()

plt.plot(ts\_week\_log\_diff)

ts\_week\_log\_diff.dropna(inplace=True)

test\_stationarity(ts\_week\_log\_diff)

#ACF and PACF

lag\_acf = acf(ts\_week\_log\_diff, nlags=10)

lag\_pacf = pacf(ts\_week\_log\_diff, nlags=10, method='ols')

#Plot ACF:

plt.subplot(121)

plt.plot(lag\_acf)

plt.axhline(y=0,linestyle='--',color='gray')

plt.axhline(y=-7.96/np.sqrt(len(ts\_week\_log\_diff)),linestyle='--',color='gray')

plt.axhline(y=7.96/np.sqrt(len(ts\_week\_log\_diff)),linestyle='--',color='gray')

plt.title('Autocorrelation Function')

#Plot PACF:

plt.subplot(122)

plt.plot(lag\_pacf)

plt.axhline(y=0,linestyle='--',color='gray')

plt.axhline(y=-7.96/np.sqrt(len(ts\_week\_log\_diff)),linestyle='--',color='gray')

plt.axhline(y=7.96/np.sqrt(len(ts\_week\_log\_diff)),linestyle='--',color='gray')

plt.title('Partial Autocorrelation Function')

plt.tight\_layout()

model = ARIMA(ts\_week\_log, order=(2, 1, 1))

results\_ARIMA = model.fit(disp=-1)

plt.plot(ts\_week\_log\_diff)

plt.plot(results\_ARIMA.fittedvalues, color='red')

plt.title('RSS: %.4f'% sum((results\_ARIMA.fittedvalues-ts\_week\_log\_diff)\*\*2))

print(results\_ARIMA.summary())

# plot residual errors

residuals = DataFrame(results\_ARIMA.resid)

residuals.plot(kind='kde')

print(residuals.describe())

predictions\_ARIMA\_diff = pd.Series(results\_ARIMA.fittedvalues, copy=True)

print (predictions\_ARIMA\_diff.head())

predictions\_ARIMA\_diff\_cumsum = predictions\_ARIMA\_diff.cumsum()

predictions\_ARIMA\_log = pd.Series(ts\_week\_log.iloc[0], index=ts\_week\_log.index)

predictions\_ARIMA\_log = predictions\_ARIMA\_log.add(predictions\_ARIMA\_diff\_cumsum,fill\_value=0)

predictions\_ARIMA = np.exp(predictions\_ARIMA\_log)

plt.plot(ts\_week)

plt.plot(predictions\_ARIMA)

plt.title('RMSE: %.4f'% np.sqrt(sum((predictions\_ARIMA-ts\_week)\*\*2)/len(ts\_week)))

size = int(len(ts\_week\_log) - 15)

train, test = ts\_week\_log[0:size], ts\_week\_log[size:len(ts\_week\_log)]

history = [x for x in train]

predictions = list()

size = int(len(ts\_week\_log) - 15)

train, test = ts\_week\_log[0:size], ts\_week\_log[size:len(ts\_week\_log)]

history = [x for x in train]

predictions = list()

print('Printing Predicted vs Expected Values...')

print('\n')

for t in range(len(test)):

model = ARIMA(history, order=(2,1,1))

model\_fit = model.fit(disp=0)

output = model\_fit.forecast()

yhat = output[0]

predictions.append(float(yhat))

obs = test[t]

history.append(obs)

print('predicted=%f, expected=%f' % (np.exp(yhat), np.exp(obs)))

error = mean\_squared\_error(test, predictions)

print('\n')

print('Printing Mean Squared Error of Predictions...')

print('Test MSE: %.6f' % error)

predictions\_series = pd.Series(predictions, index = test.index)

fig, ax = plt.subplots()

ax.set(title='Spot Exchange Rate, Euro into USD', xlabel='Date', ylabel='Euro into USD')

ax.plot(ts\_week[-60:], 'o', label='observed')

ax.plot(np.exp(predictions\_series), 'g', label='rolling one-step out-of-sample forecast')

legend = ax.legend(loc='upper left')

legend.get\_frame().set\_facecolor('w')

-------------Dicky fuller test End ---------------------------------------

Stationary Trend - Means data values or period cycle are almost same/constant over the time.(horizantal trend).

Non-Stationary Trend - data values or period cycle is not constant, either it is going increasing or decreasing or there are jumpsupward or downward trend. For eg. In one month we have 1000 transaction but second month has 50000 transaction.

----------------------Time Series Analysis END------------------

-----------------------------------------Model Evaluation-----------------------------

# Model Evaluation

##### **Sampling based evaluation –**

Test model on different samples by split data in multiple sets-

*X\_train1,X\_test1,y\_train1,y\_test1 = train\_test\_split(X,y,random\_state=10)*

*X\_train2,X\_test2,y\_train2,y\_test2 = train\_test\_split(X,y,random\_state=20)*

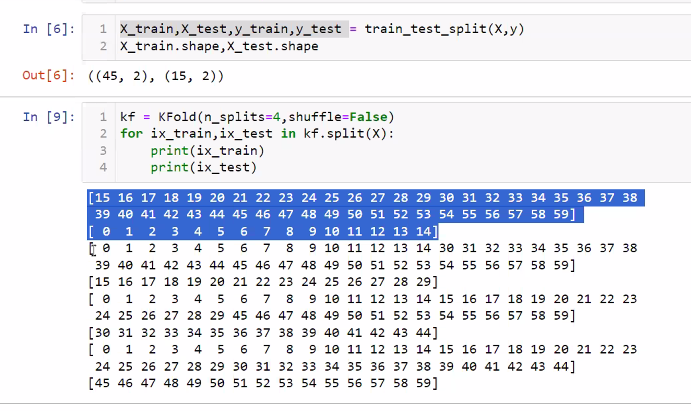
This will return some duplicate sample in set1 and set2. As best practice, we should use K-Fold that will not repeat the sample in sets.

##### Model evaluation parameters –

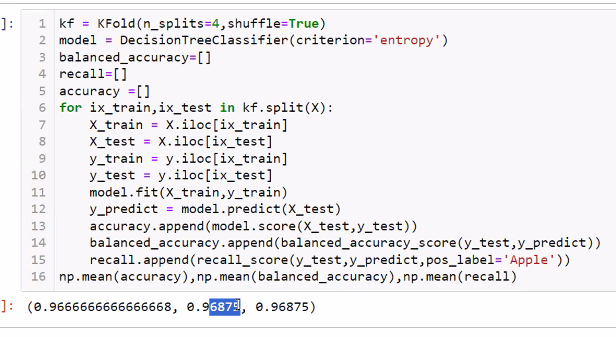
* Model degree of freedom
* Residual degrees of freedom(DF)
* SSE(Sum Squared Error)
* MSE(Mean Squared Error)
* R-Squared – It shows how data fit into model. It has value between 0 to 1. Higher value are good.
* Standard error
* p-value – Pvalue is the significance between 0 to 1. Usually pvalue cutoff is 0.05. If it is lesser than cutoff then performance is good. More pvalue means data has errors.

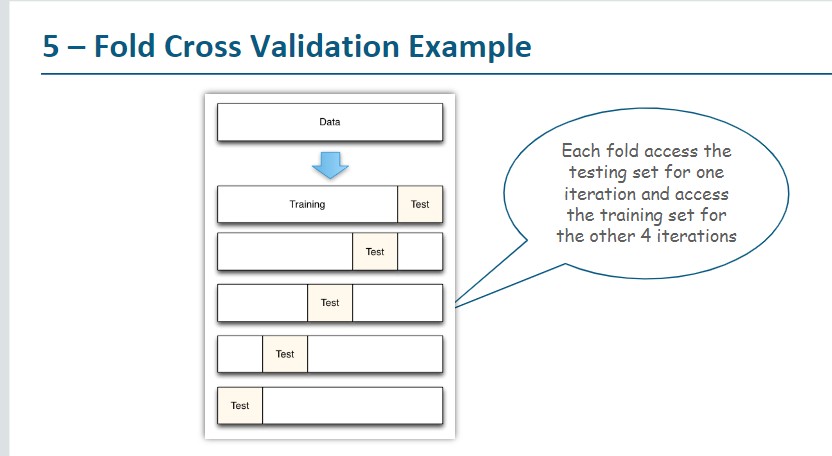
Model evaluation is the process to finding best accuracy by doing different kind of testing or applying different models. There are below ways to get model accuracy-

* Train and Test on same data
* Train and Test on different data – It can lead to huge variance.
* Cross Validation – Test subset data from Train and Test both and average the result together.
  + KFold validation – It is one type of cross validation. K(eg. ) is the number of subsets/folds. It takes one subset of data from testing and cross validates it with one by one 4 subsets from training.
  + It will not repeat the sample among the folds. Below are the multiple splitted sets-



We need to pass all sets then get accuracy for each set and at then end average them-



* + It split samples in multiple folds. One fold will go for training and remaining will be used for testing.

|  |  |
| --- | --- |
| **Cross Validation** | **Train/Test Split** |
| More accurate estimate of out-of-sample accuracy. | Run K times faster than K fold cross-validation. |
| More efficient use of data(every observation used for both training and testing. | Simpler to examine the detailed result of testing process. |
|  |  |
|  |  |

Note: Usually we run both evaluation models on dataset and whoever have more accuracy will be selected. When we run multiple evaluation algorithms, there are chances the best model also providing lesser accuracy or error for particular subset. These subset needs booster.

**Leave one out Method**– It is same as K-Fold method where K = No. Of samples. In this method, one sample will be reserved for testing and rest all will used for training. This process will keep repeating for all samples. The loop number or folds would be equal to number of samples. This method is used when the number of samples are less such as 20. As we have only one sample to test in each loop, it will always give accuracy as 1 or 0. If we apply the method on medium or large sample size, it will be too costly due to huge train-test sets.



**Leave POut method:** It needs P as parameter that is number of samples in pivot. If p=2 then it will create all unique combinations of two sample. In other words, it will find all distinct samples in size of P for testing and remaining samples(Total samples-P) will consider for training.

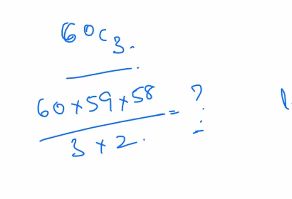
For example if we have 60 samples and P=2 then it will create

(60 \* (60-1))/2 = 1770 iterations

So total, 1770 unique combination iteration will happen where each time combination samples will go for testing and remaining will go for training.

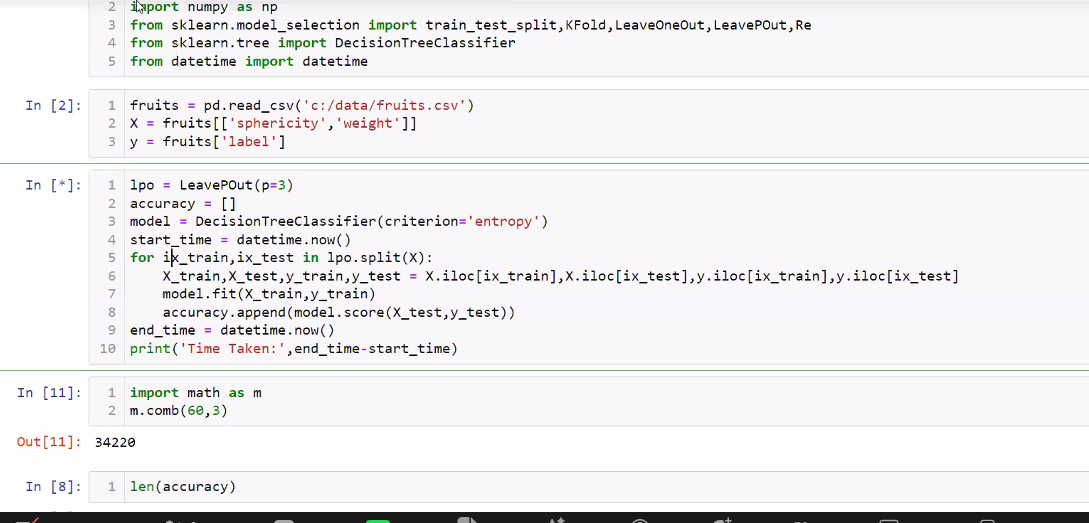
If p=3 and 60 samples then

(60\*59\*58)/(3\*2) = 34220 iterations



If we pass p=1, then it will be like Leave one out as each sample is considered as unique combination.

It is method is recommended if the samples are less like 10 or 20.



Repeated KFold: It will repeat the samples.

Shuffled KFold : It takes K as number of folds with test size, the remaining will be considered for training. Each set will be shuffled data.

**Boot Strap-** It is sampling method that allows repeating the samples. In other sampling methods, we get each sample once only in each set but in this method we can have same sample more than once(repeated). This enables us to have more samples for testing than the available sample size.

* Some percentage of samples will be there in training and testing both.
* Good sampling method when we want to test the model with few samples used in training with new samples in testing.

------------------------------------Boosting ---------------------------------

##### Boosting –

It is process to convert week learners to strong learners. 3 types-

1. AdaBoost(Adaptive Boosting)
2. Gradient Tree Boosting
3. XGBoost

* Decision Tree Stumping

import pandas as pd

from sklearn import model\_selection

from sklearn.ensemble import AdaBoostClassifier

df = pd.read\_csv('Diabetes.txt', sep=",", header=None)

df.columns = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

array = df.values

X = array[:,0:8]

Y = array[:,8]

kfold = model\_selection.KFold(n\_splits=10, random\_state=7, shuffle=True)

model = AdaBoostClassifier(n\_estimators=30, random\_state=7)

results = model\_selection.cross\_val\_score(model, X, Y, cv=kfold)

print(results.mean())

##### [AdaBoost Mechanism](#_AdaBoost_Mechanism)

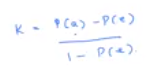
------------------------------------Boosting END---------------------------------

------------------------------------------Model Evaluation Metrics-----------------------------

**Simple Accuracy** =This is metrics to evaluate overall model . Correct predicted samples/Total samples

**Balance Accuracy Score** : This is metrics to evaluate overall model . It meaning correctly answered accuracy score for each class. It calculates score for each class individually then sum it and divide by no. Of classes. For example first class calculated 100% means all predicted correctly, 2nd 60% and then 3rd 50% then (100+60+50)/3=70%. If BalanceAccuracy Score is greater than accuracy score then it means model learned. If Balance is lesser than accuracy score that means, model didn't learn or data is imbalanced.

**Cohen Kappa Score** = This is metrics to evaluate overall model . It is agreement between predicted and actual values. If it is more than 0.8 then it’s good.

The formula is – 

Here P(a) stands for actual predicted(or Simple Accuracy) and P(e) stands for expected predicted.

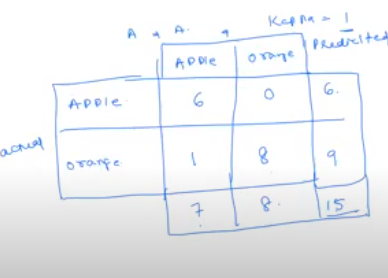
(Actual correct class1/total sample)\* total predicted class1/total samples)

/

(total samples of class2/total samples\*total correct prediction/total samples).

The formula is K = p(a) - P(e)/1-P(e)

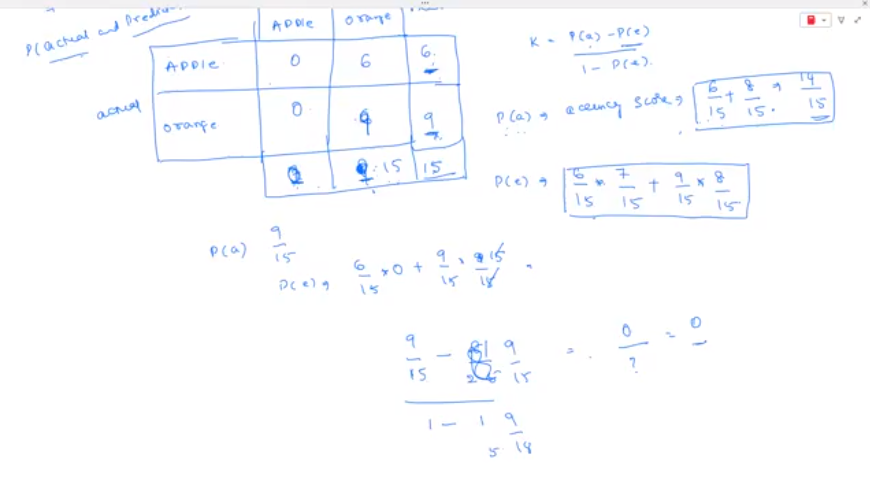
It focuses not only on the correct values predicted but consider the correct value predicted in wrong class as well.

Example - 

In above, P(a) = 6/15+8/15

P(e) = Apple(Class1) : (6/15\*7/15)

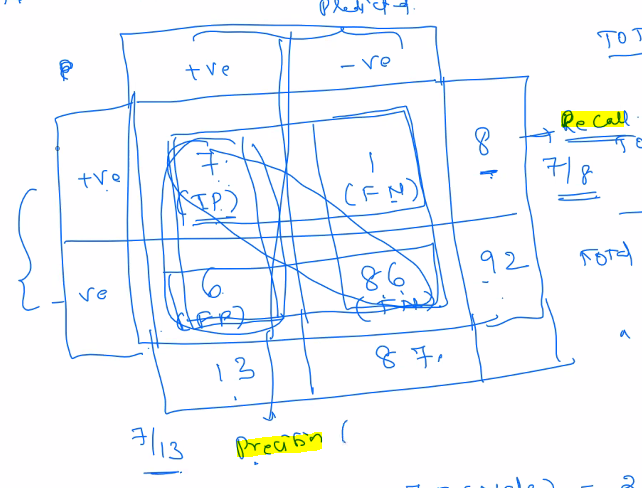
Orange(Class2) : (9/15\*8/15)



**Precision:** This is metrics to evaluate focused or class of interest model; such as from predicted Apple and Oranges, we are interested in Apple only. So the Apple is Positive.

In Confusion matrix, the Precision is - Predicted correct positive values/Predicted Column total. TP/TP+FP.

Precision is useful in like detecting valid email out of spam emails. While calling Precision score function, we have to provide class of interest label name.

****

**Recall :** This is metrics to evaluate focused or class of interest model; such as from predicted Apple and Oranges, we are interested in Apple only. So the Apple is Positive. In Confusion matrix, the recall is- Predicted correct positive values/actuall row total. TP/TP+FN

It mostly used in Medical diagnosis where False Negative has more emphasis. Higher Recall is good for medical diagnosis. While calling Recall score function, we have to provide class of interest label name.

Recall is also called as **True Positivity Rate(TPR) or Sensitivity or Hit Ratio.**

**FNR(False Negativity Rate)** = FN/TP+FN

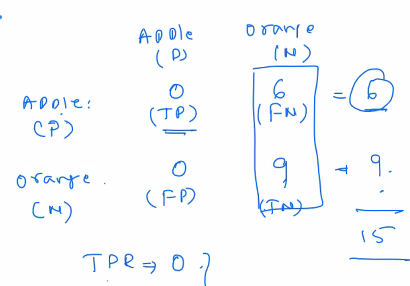
Note: TPR+FNR = 1

**FPR(False Positivity Rate)**: Ratio of false positive vs FP/FP+TN

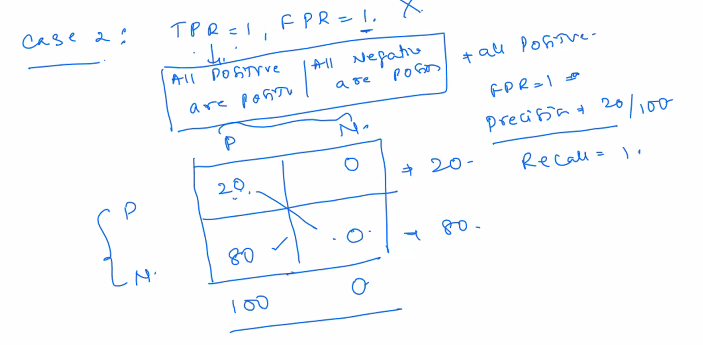
**TNR(True Negativity Rate)**: TN/TN+FP. It is also called as Specifity.

Note: TNR+FPR = 1

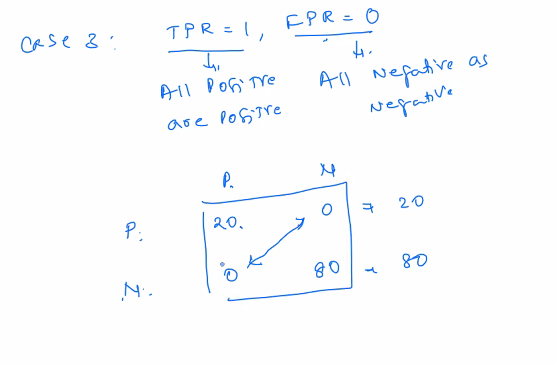
**Case1** - If TPR and FPR both are 0 for model that means, all samples are predicted as Negative. So we can’t accept the model.



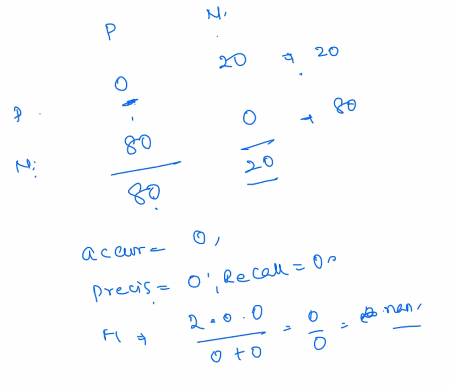
**Case2** - Similarly, if TPR and FPR both are 0 then all samples are wrongly predicted as Positive and model has not learnt anything.



**Case3** – If TPR=1 or close to 1 and FPR=0 or close to 0 then model is good. Means, mostly positive predicted as positive and negative as negative. Or in other words, the positive diagonal should have more values than negative diagonal.



**Case4 –** If TPR = 0 and FPR = 1 then it is worst model means all negatives predicted as positive. So, the diagonal will be reverse.

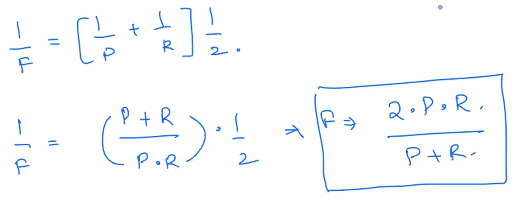


**F1 Score :** F1 measure is combination or harmonic mean of Precision and Recall. It is focused or class of interest model evaluation. If F1 measure is more than 0.8 than model is good.

While calling F1 score function, we have to provide class of interest label name.

f1\_score(y\_test,y\_predict,pos\_label=’M’)

Note: pos\_label is the class of interest.



<https://en.wikipedia.org/wiki/Receiver_operating_characteristic>

**Other meaning -**

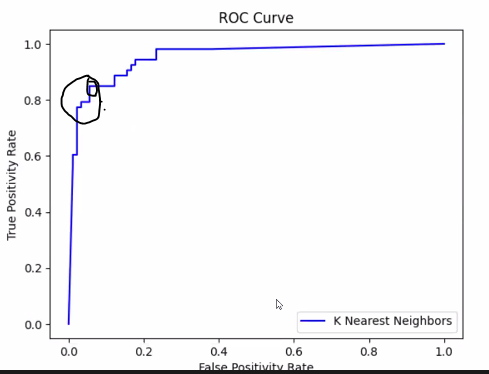
Total Positive Predicted = TP+FP

Total Negative Predicted = TN+FN

Total Actual Positive = TP+FN

Total Actual Negative = TN+FP

**ROC(Receiver Operating Characteristic or ROC Curve):** it has TPR on column and FPR on rows and it look for the threshold point for each sample. It takes class of interest sample probabilities as input and return threshold i.e. (TPR and FPR) at each level to decide which threshold is best. The threshold point where highest TPR and lowest FPR is the perfect threshold that we should consider. The area under these points for threshold means samples will predict correct at this threshold is called AUC(Area under ). The highest area is considered good.



**Conclusion** : To evaluate the model, we shoud list all the evaluation metrics to check model performance with all perspective and the model having higher TPR and less FPR would be the best model.

------------------------------------------Model Evaluation Metrics END-----------------------------

-----------------------------------------Model Evaluation END-----------------------------

-------------------------------- Visualizations tools--------------------------------

#### Visualization tools -

Power BI

---------------------------Charts begin ----------------------------

##### Charts –

##### Compare values across categories

* Column Chart – Compare values of diff. categories
* Clustered Chart - show values of multiple category groups
* Trellis Chart – Multiple categories across multiple dimensions are compared
* Bar Chart – compare categories
* Bullet Chart – highlighting KPI

##### Part to whole relationship

* Stacked Bar Chart – How category made up of another element
* Stacked Bar Chart – Part to whole relationship
* Pie/ Ring/ Doughnut Chart – Distribution
* Funnel Chart – Used to define bottlenecks in workflow by classifying
* Waterfall – show how values increased/decreased to reach final stage.

##### Relationship among variables

* Scatter – Correlation between two variables
* Heat map – Highlight relationship across two dimensions and a metric with colors
* Circle Grid – Highlight relationship across two dimensions and two metrics with color and size
* Event Chart – Aligns the occurrence of events against the values of a numeric data set over time
* Bubble Chart – Correlation between three variables

##### Compare over time

* Stacked Bar chart- Metric over a period of time
* Trellis Line chart – Time across multiple categories
* Step Chart – Sudden changes in timeseries data is highlighted
* Area Chart – Trend with Timeseries as volume
* Week Density – Assistance decision-making regarding in resource planning

##### Frequency (Count) of values

* Histogram – Count by bin
* Box And Whisker Plot – Distribution by group. Whisker means maximum and minimum. Plot contains median and first and third quartiles.

##### Maps

* Filled Geographical Map – fill color and border areas within map
* Symbol Geographical map – highlight location by color and size. Eg. Usage by location
* Density Geographical Map – Show location of data points using bubble size.

##### KPI

* Dial
* Big Number chart
* Gauge chart
* Bullet Chart

Parend Child Hierarical data –

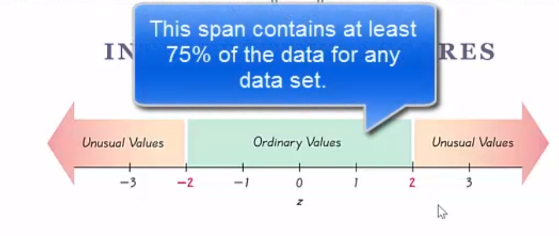
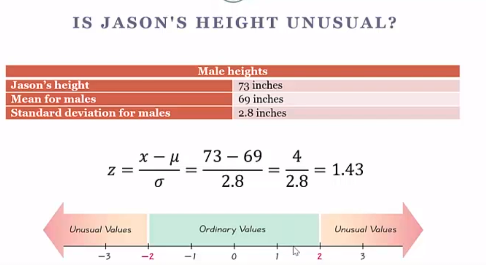
Sunburst chart

---------------------------Charts begin END ----------------------------

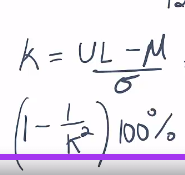
-------------------------------- Visualizations tools END--------------------------------

----------------------------------Statistics begin--------------

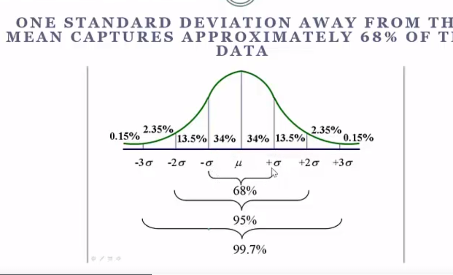
* **ZScore –** 
  + It is used to calculate score the distance from Mean(µ).
  + It can return negative ZScore as well.
  + 2 STD(σ) span from mean(µ) contain 75% on data and that called Ordinary values. Remaining values called un-usual data.
  + In below example, we calculate Zscore i.e. 1.43, from given Json’s heigh. This zscore fall within two STD span hence his height is usual.

* **Chevyshevs Theorom –** 
  + It is used to calculate Minimum percentage from given range.
  + We need to subtract mean from upper limit from range. Then Divide by STD(σ))
  + It don’t care about data dispersion. Whether it is symmetry or not
  + It always returns positive value only.
  + If k=2, 75% of data population will fall within Mean(µ)+2 and Mean(µ)-2



* **Empirial Theorom** –
  + Data must in bell curve and symmetric
  + The 68% values within 1 span of deviation from mean.
  + 95% data values are within 2 span of deviation from mean.
  + 99.7% data is within 3 span of deviation from mean.
  + It returns approximate result.



It needs symmetry data only

---------------------------------Statistics end------------------

------------------------------Case Study-----------------------------------------

#### Case Studies-

------------------------------Case Study END-----------------------------------------

-------------------------------Other Practices----------------------------------

#### Other practices

----------------------------Requirement Gathering---------------------

##### Requirement Analysis, Elicitation and case understanding

1. Understand domain
2. Overview of requirement
3. Context – Context around the requirement
4. Objective – Expected result like sales prediction etc.
5. Key issues
6. Considerations - like age should not differentiate etc.
7. Data Description and Volume
8. Data source
9. Data cleaning required
10. Business Benefits
11. Finalize Approach. Like solution steps, Libraries algorithm, platform etc.
12. What is accuracy expected as per case and model. It is required to test overfitting and under fitting

----------------------- Requirement Gathering End ---------------------------------

----------------------Hyper parameters for model selection------------------

##### Hyperparameters

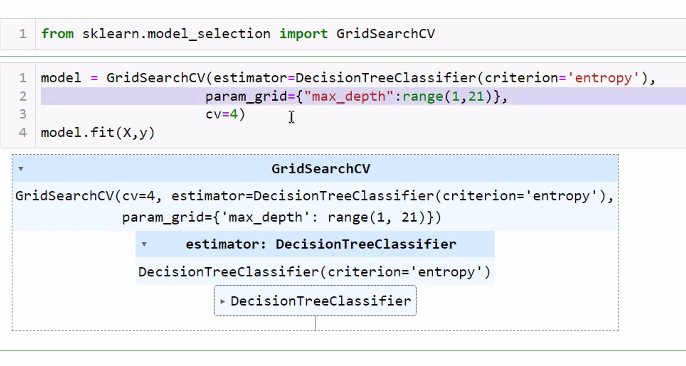
Hyperparameters are the parameters to tune the algorithm to get best fit from it.

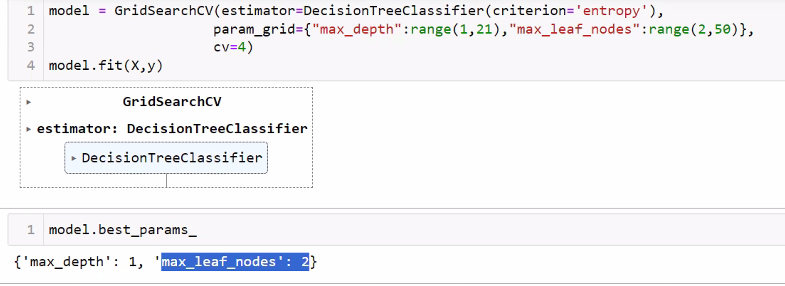
Hyperparameter search can be done in two ways-

1. Grid wise search –Grid search is utility to apply all parameters combinations given. The parameter values are equally distributed within the parameter range specified by user. End value and it’s output are checked before dividing the final value.

For eg. If given range is -5 to 5 then the value would be -5, -4, -3, -2, -1 , 0 ,1 ,2 ,3, 4, 5

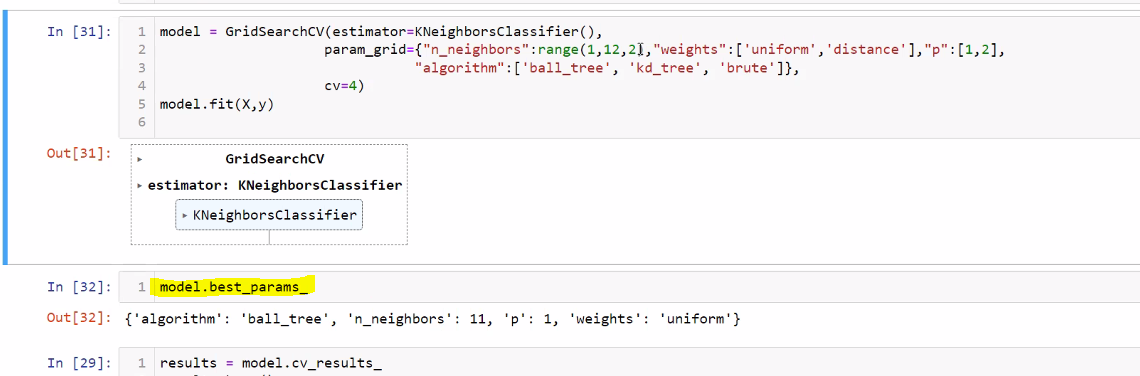
It accepts model estimator, number of sample folds i.e. Cross validation etc has parameter and will execute it and return the best combination for model.

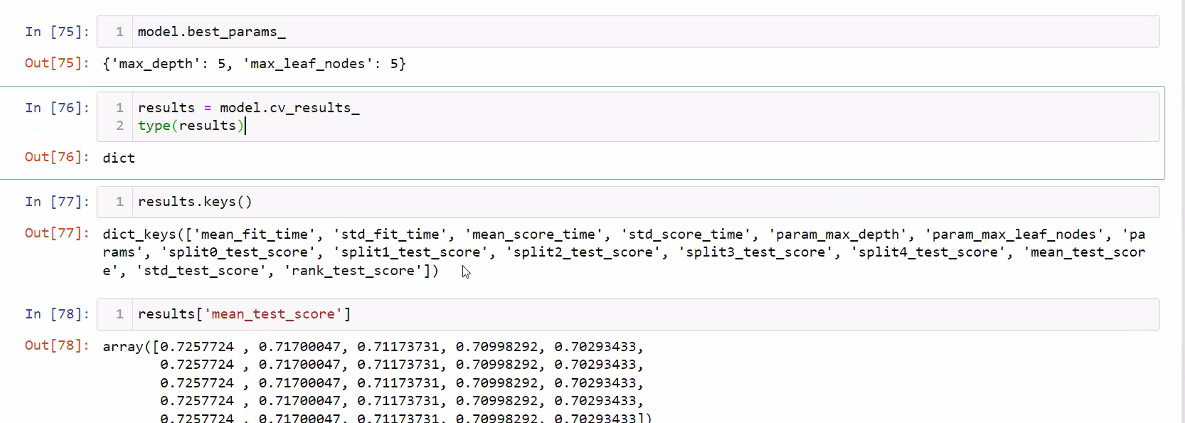




It returns dictionary as key value pair for each applied scoring method like ranking, mean\_test\_score, split0\_test\_score etc. Where scoring method is the key and values would be result set for that method

Model.best\_params\_ will return the best algorithm.





1. Random Search – The hyper parameter value randomly chosen within the range specified by the user.

This is same as Grid wise search but it allows to set number of iterations so it will be quick. So in results[‘Params’] we will get parameter combination same as given number of iterations.

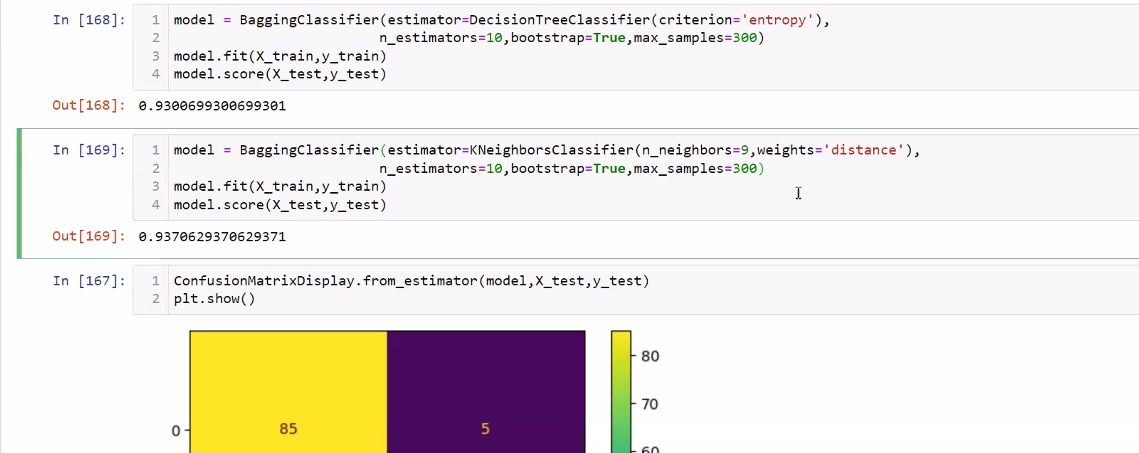


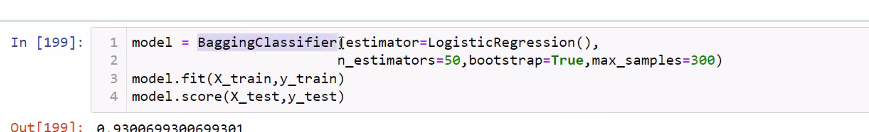
**Ensemble Techniques** – It is kind of technique to learn from mistakes. It will create multiple estimators based on given hper-parameters and will compare and learn from each estimator and decide the best estimator by combining them.

* **Random forest Classifier**– it is extension for decision tree. It will create multiple decision trees based on different condition like different weight and will collect learning from each tree and will decide the what returned by most of the estimators(majority). It support Decision tree only.

It accepts boot\_strap as hyper-parameter to replace(duplicate) or unique samples.

* **Bagging Classifier** : It is like RandomForest that allow other algorithms with hyperparameter but it support KNN, Regression and Decision tree all compare to Random Forest who supports only Decision tree. It does same process like multiple estimators then come up with the best prediction. For example, if there are 5 estimators, out which 3 suggested prediction A and 2 suggested prediction B then the result would be A.





**Boosting technique** – In this technique, it will give preference for errors from first estimator.

**AdaBoostClassifier** – It will focus on incorrect predicted samples in subsequent estimations.

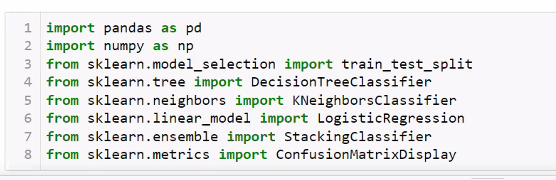
We need to provide estimation algorithm like Decision Tree and number of estimator( n\_estimator). If first estimator itself is 100% in training then it will stop right there. But if first estimator is less than 100% then it will try incorrect predicted samples in second estimator. In this way it will keep goes.

**Gradient Boost**

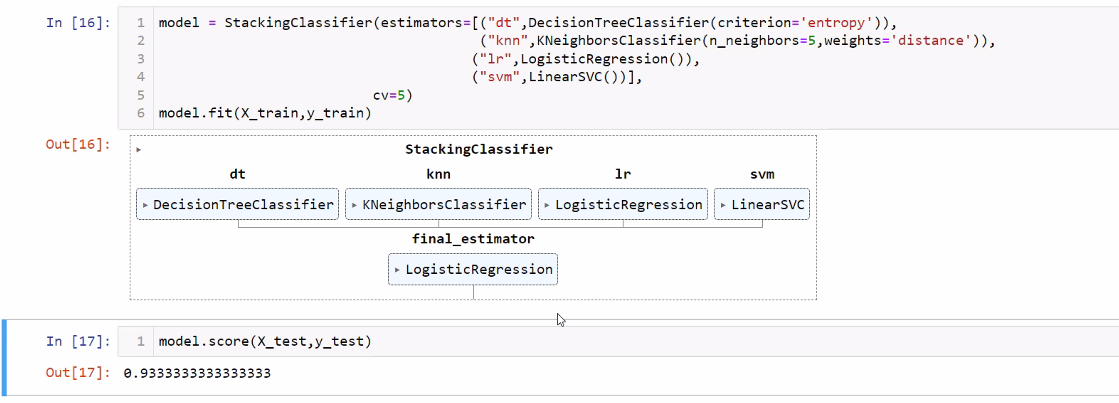
**XGBoost**

**Stacking Classifier**: This is technique as other ensemble classifer but it allows to apply multiple algorighm like Decision Tree and KNN together. That means, predict sample with multiple classifiers then based on majority the sample will be predicted.

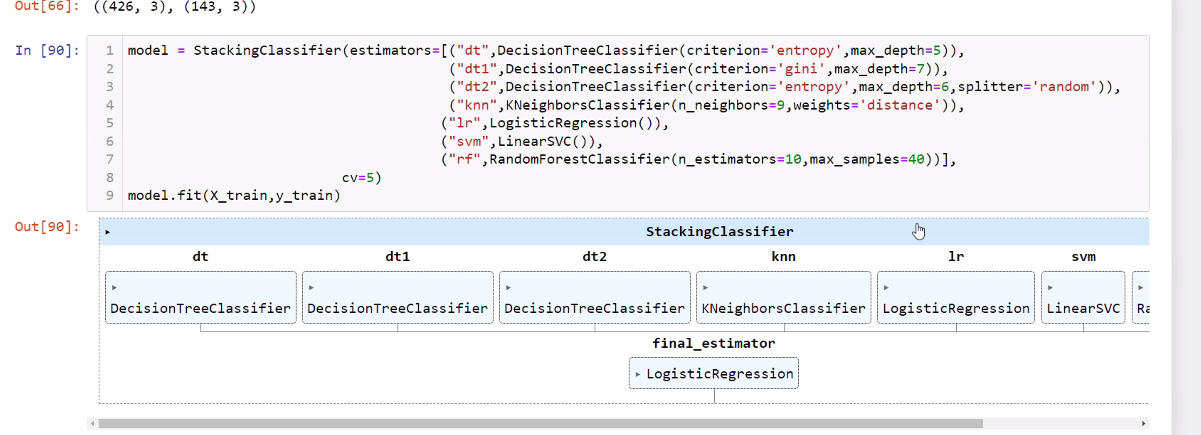
If we don’t provide estimator then the default estimator is Logistic Regression. After applying all given Classifier, the final result will be passed to Logistic regression that will apply probability based on accuracy predicted by previous classifier.







We can apply classifier with hyper-parameter –



----------------------Hyper parameters END------------------

-------------------------------Other Practices END----------------------------------

----------------------Libraries----------------------

#### Libraries

* Pandas
* Numpy

##### ScikitLearn –

It contains datasets as well.

*from sklearn.model\_selection import train\_test\_split – to train and split data*

*X=iris.data*

*y=iris.target*

*X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.2)*

* Seaborn – Visuals for statistical model like heatmap
* Matplotlib – Charts like Bar, Line, Scatter, Pie, Quiver Plots, Contour Plots
* Scipy
* Plotly

----------------------END of Libraries----------------------

Resources

Khan Academy

3Blue1Brow youtube channel for Neural Networks

Medium

----------------------------Certification -----------------

#### Certifications

-----------------------------Certification End----------------------

---------------------------Deep Machine Learning-----------------

**Color image:**

RGB – Red, Green and Blue

All images are in 2 dimensional arrays where each element holds pixels three RGB values so it is 3d array.

**Black and White images:** These are called as Gray-Scale images. Each pixel holds only one value hence it is 2D array. White = all 255 and Black = all 0.

While plot black and white image, we need to mention cmap=’gray’

Px1 = plt.imread(“c:/image.jpg”)

Plt.imshow(px\_gray,cmap=’gray’)

Sometimes, RGB value could be divided by 255 and presented as between 0 to 1.

In order to convert color image to B&W, just set Red and Green array position to 1 using px1[:,:,0]=px1[:,:,1]

---------------------Open CV library ----------------

It is used for real-time computer vision. It is freeware but not directly available in Anaconda setup. We need to install in manually.

Matplotlib.imread read image as RGB order pixels but OpenCV.imread read as BGR.

!pip install opencv-python

import cv2 as cv

import matplotlib.pyplot as plt

px = cv.imread("d:/cat.jpg")

plt.imshow(px)

px\_gray=cv.cvtColor(px,cv.COLOR\_BGR2GRAY) #to change BGR to RGB

plt.imshow(px\_gray,cmap='gray') #to deplay image in gray mode

px\_rect = cv.rectangle(px\_rgb,[125,25],[220,125],[255,0,0],3) #to draw rectangle on image(eg. On face)

#to crop the face

*face\_crop = px\_rect[25:125,125:220]*

*plt.imshow(face\_crop)*

*plt.show()*

#resize image

*Px\_resize = cv.resize(px\_rgb,[100,100])*

*Plt.figure(figsize=[3,2])*

*Plt.show(px\_resize)*

*--------------------------------Face detection---------------------------------*

*Fer – Facial expression recognition. pip install fer*

*From fer import FER*

*Import cv as cv2*

*Face\_detect = FER()*

*Face\_detect.find\_faces(px)*

*Face\_detect.find\_Faces(px)*

*Sample face images available at –*

*The first number in image name is age of person then second is gender then race.*

[*https://susanqq.github.io/UTKFace/*](https://susanqq.github.io/UTKFace/)

*Fashion, Hand written and various datasets -* [*https://www.tensorflow.org/datasets/catalog/mnist*](https://www.tensorflow.org/datasets/catalog/mnist)

------------------------------------------------------------------------CV library---------------------------

Deep Learn ML uses Neuron that accepts each sample with random weight then sum it up. It uses Linear Regression algorithm with activation function like sigmoid, Relu etc. Basically, it tried to find and apply the non-linear in data. It apply random weight and see the impact, if error decreased then it will increase the weight but if error increase then it will decrease the weight based on weight and error pattern.

If data is normal linear data then ML should be use but if it is non-linear then Deep Learn should use.

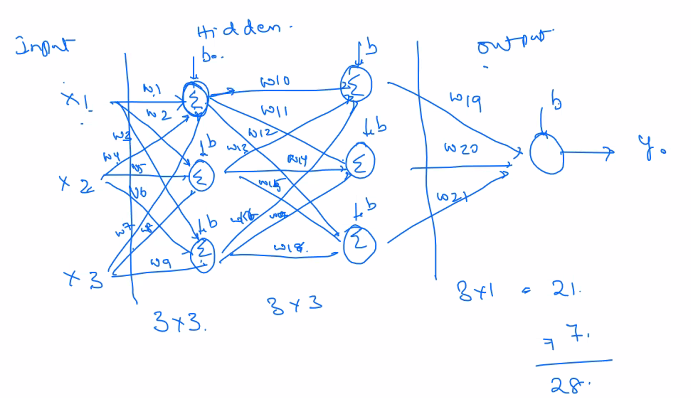
Neuron calculation = Sigmoid(x)+b

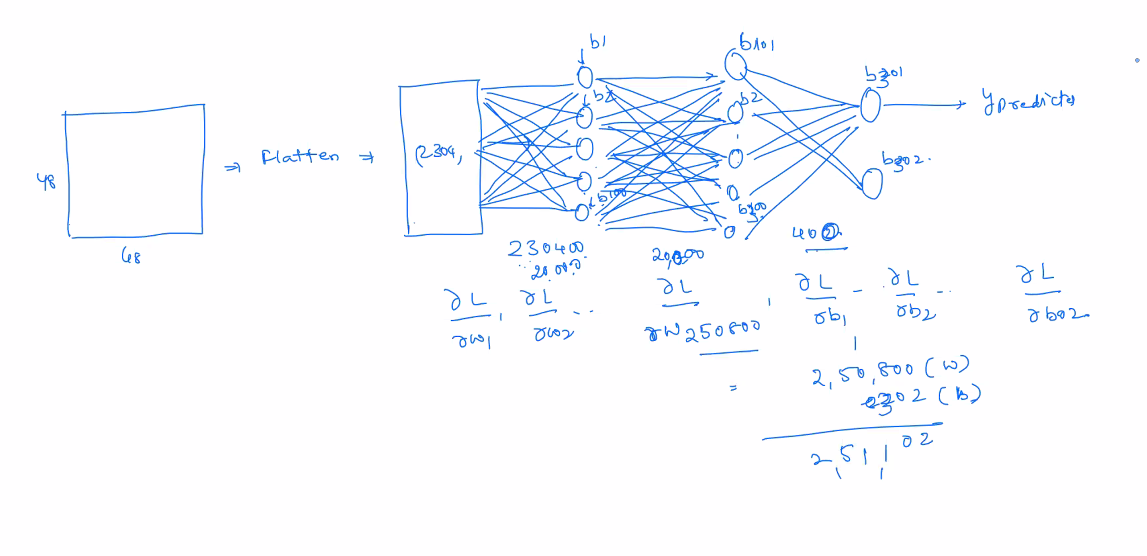
Neural Network is network of neurons.

Types of Neural Networks

* ANN – Artificial Neural Network or Multi Layer Perception or Fully Connected Neural Network
* CNN
* RNN
* GAN (Encoders and Decoders)

----------------------------------------------ANN--------------------------

In ANN, it tries to learn W and Bs to minimize the loss. 



X is input feature sample or each pixle.

W is the random weight value added into formula to calculated Y.

Weight value calculation is based on loss gradient.

Next W = Old W=dL/dW\*Learning rate --note dL/dW called as partial dervicative.

*Derivative of Loss and Derivative of W.*

*Where Loss = 1/n(yActual-yPredicted)Square*

Learning Rate – There is possibility that dL/dW return huge number but we need to keep it minimal to keep to avoid big differences in W, hence we multiple each dL/dW with some constant rate i.e. Learning Rate usually 0.01. If dL/dW returns big value like 1000 then the 1000 will multiply with 0.01.

b is bias added into x for each neuron.

Activation function - Once one hidden layer completed it will generate linear output and then the activation function will make it non-linear and will pass it to next layer to add further W and B. Linear means straight line and non-linear means curvy or daggle line. If Activation function is not passed then the it will become simple linear only.

The Y or the output would be number of classes. If it is binary like for gender then two dimensional will have (0,1) or (1,0).

The sum of Ws+Bs should equal trainable parameters.

Percepron – If we have only one neuron with input and output.

MLP – Multiple layer percepron – it is when we have multiple neurons.

The main purpose of neuron is to reduce the loss.

Loss Error calculation methods are MSE(Mean Squared Error) and MAE(Mimimal Average Error)

In Deep learning, we need to decide the network for that we need to provide-

1. Hidden Layer,
2. Neurons,
3. Activation Function,
4. Loss function

Calculus – Calculus is used for Grandient(trend or slope of learning learning). There are two types of calculus-

1. Differential Calculus – Cut the values in small pieces to see how it changes.(ref. www.Mathisfun.com)
2. Integration/Integral Calculus – Join the small pieces together to find how much there is.

Epoch is the process(Feed Forward + Back Propogation) that keeps repeating and each epoch has its loss. Epoch steps are-

1. Initialize Random Weights
2. Feed-Forwarded - Calculate the loss for Y Predicted and update the Ws for next layer. In first epoch, it will take random Ws but thereafter, it will take Ws from previous epoch Back-propogation.
3. Calculate the Loss - (1/n(Y-actual-Y-predicted) square
4. Back Propagation – Update the loss for next epoch to calculate the Ws.

<https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd>

**Drop out layer** – To avoid over fitting, this layer will randomly make few neurons as 0 to remove their weight- age. It will not contribute in forward- propagation or back propagation. We need to provide percentage of neurons needs to drop. Drop out layer doesn’t learn anything.

*Model.add(dropout=0.5)*

**Activation functions –**

* Sigmoid- It returns values 0 or 1 based on probability. If the expected output is binary then Sigmoid is good. Hence it is used in last layer mostly. Sofmax is better than Sigmoid as it support multiclass as well.
* Tanh- It return value between -1 and 1.
* Relu(rectified linear unit activation function) – It is maximum of x and y. It will replace negative value to 0. It used for regression models hence mostly used in hidden layers. If it returns 0 then neuron will deactivated.
* Leaky\_Relu -maximum of (0.01,y). It used when we don’t want to get 0 (as in Relu) as that will totally deactivate the neuron. And we want to keep neuron activated with small values.
* Lazy,
* SoftMax – It used for multi-class classifier hence it mostly used in last layer in model.

If model is not learning anything that means the loss is constant then we need to increase the number of layers i.e. Dense and if learning speed is less then we need increase the neurons.

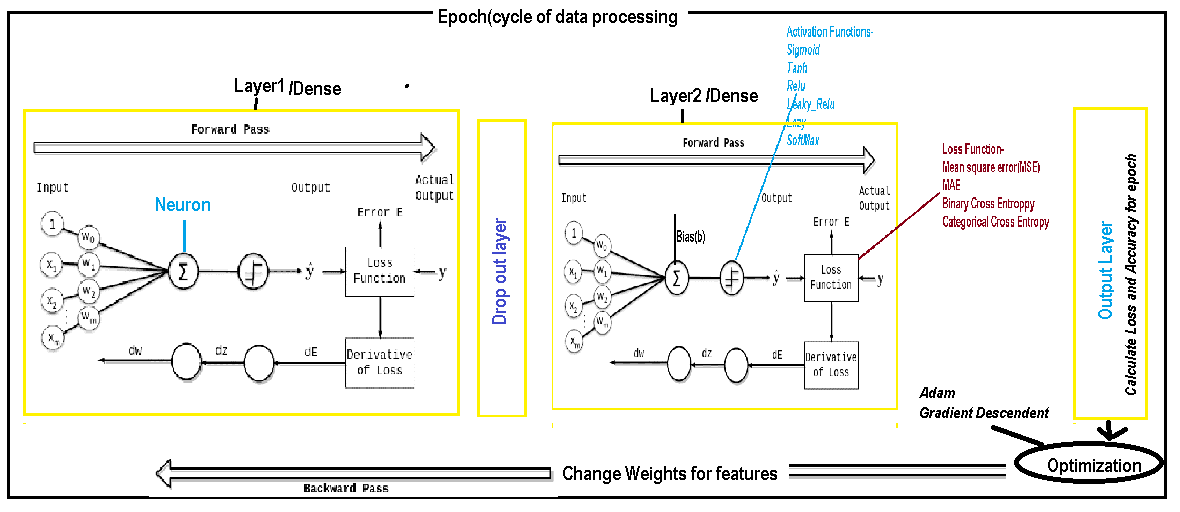
Adding validatin\_split in model in while train the model will also increase the accuracy.

**Loss Functions-**

* Mean square error(MSE)-
* MAE
* Binary Cross Entroppy – It is used for binary classification.
* Categorical Cross Entropy – It used multiclass classification.

**Optimizers** –

* Adam - Adam is most popular and used. And It has better performance.
* Gradient Descend(sgd) – It is stoghastic gradient descendent. It is poor compare to Adam hence it avoid.



--------------------------------------------ANN END-----------------------------------

-----------------------------------CNN(Convolutional Neural Network--------------------

It gives more accuracy.

It has extra two layers Convolution and Pooling layer. We can add more than once both the layers in network.

It is more focused on higher value or color pixel. It ignores pixel without any value. Basically it detects edges, color, gradient orientation, etc. in the object. Hence it reduce the number of samples.

In Convolutional method, it will identify the changes pixel series. Wherever there is change in pixel value it will capture change or edge. This process called filter or kernel.

For example we have pixel array [0,0,1,0,0,0,1,1,0,0] then from first array to next array below change calculation will happen-

* If first element is 0 and next is 1 then 1
* If both are 0 then 0
* If 1 to 0 then -1

**Kernel/Filter/K** size that is kind of group of pixel that needs to calculate in samples.

After applying the Convolutional Layer, it will convert to convelt features means only the change features.

In case of 2D data, filter size need to provide like [2,2] etc. so the how the it will move to calculate the change.

If we run Convolutional Layer multiple times, each time the samples will decrease.

It is mostly used with Visions.

After convolutional and pooling layer and reduce it to less size, that will process in fully connected neural network.

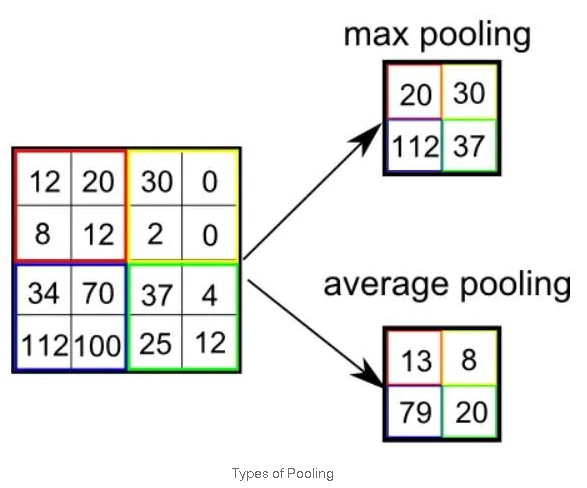
**Stride** means the step or jump for each pixel selection while traverse. If stride = 1 then it called non-stride.

**Channel –** It means number of dimensions of pixel for example, RGB image will have 3 channels.

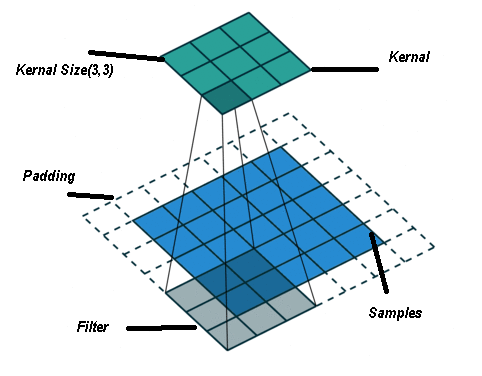
**Padding** – if it require to add extra pixels layer at all sides of image to cover all pixels while kernel move as per stride, is called padding. All extra padding pixles would be 0 only.

1. Valid Padding – No Padding
2. Same Padding

**Pooling layer** is the layer applied on convoluated output values. Its job is to extract only dominant features. There are two methods. Pooling purpose is to get only most important features.

* Max pooling – It will take max value from each kernel filter. It remove noise as well, hence it has better performance.
* Avg pooling – It will take average value of each kernel filter.
* 

<https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>



-----------------------Audio files librosa libarary--------------------

Sr = sampling rate. Values stored per second.

Mono = audio will come from one speaker/channel or two. If true means one speaker.

Example if sr=100 and it is 4 seconds file then they array shape would be 400(100x4)

------------------------audio library end-----------------------------

--------------------------------------------NLP------------------------------------------

Ngram – Sequence of words.

* Unigram – one word
* Bi-gram – two words
* Tri-gram – three words

Stop words – common used words like a, the, is etc. These needs to remove.

Tokenize words = split string into words. It takes care of special characters like % etc.

Stemming – It stem the word like convert beautiful to beauty, going to go, thanks to thank etc.

Lemmatization – It convert plural to singular etc.

Bag of words = It is processing to create bags of words on X axis and finding words in each document(string/row of text/array) and mark it true/false(0,1) if it exists. This is required to create data axis to feed model.

NLTK – Natural Language ToolKit (Text processing)

------------------------------------Naive Byes----------------

This robust algorithm compare to Decision Tree or KNN.

It consider incremental average between samples hence it has more accuracy.

With Decision tree and KNN, if sample change the accuracy will change especially if noise added to samples. But this is not with Naive byes as it calculate conditional probability that means it will consider the condition between difference of samples.

----------------------Naive Byes end--------------

RNN – Recurrent Neural Network – It repeats the process. It can overfitting. It can memorize short term information.

LSTM – Long short-Term memory – It can memorize short and long term information hence it is good for long series of data. During training, it will retain the relevant information over the time and ignore the irrelevant information. It has three gates to control-

Input Gate – It receive the new information in the cell.

Forget Gate – It removes the un-important information from the cell.

Output gate – It will extract the information from the cell and decide what to pass to next hidden layer.

------------------------------------------GENAI------------------------------------------

* Palm
* Lamda
* GPT

------------------------------------------GENAI END------------------------------------------