Report

On

Industrial Summer Training

Undertaken at

TechTrunk Ventures Private Limited, Hyderabad

From 01st June, 2019 to 01st August, 2019

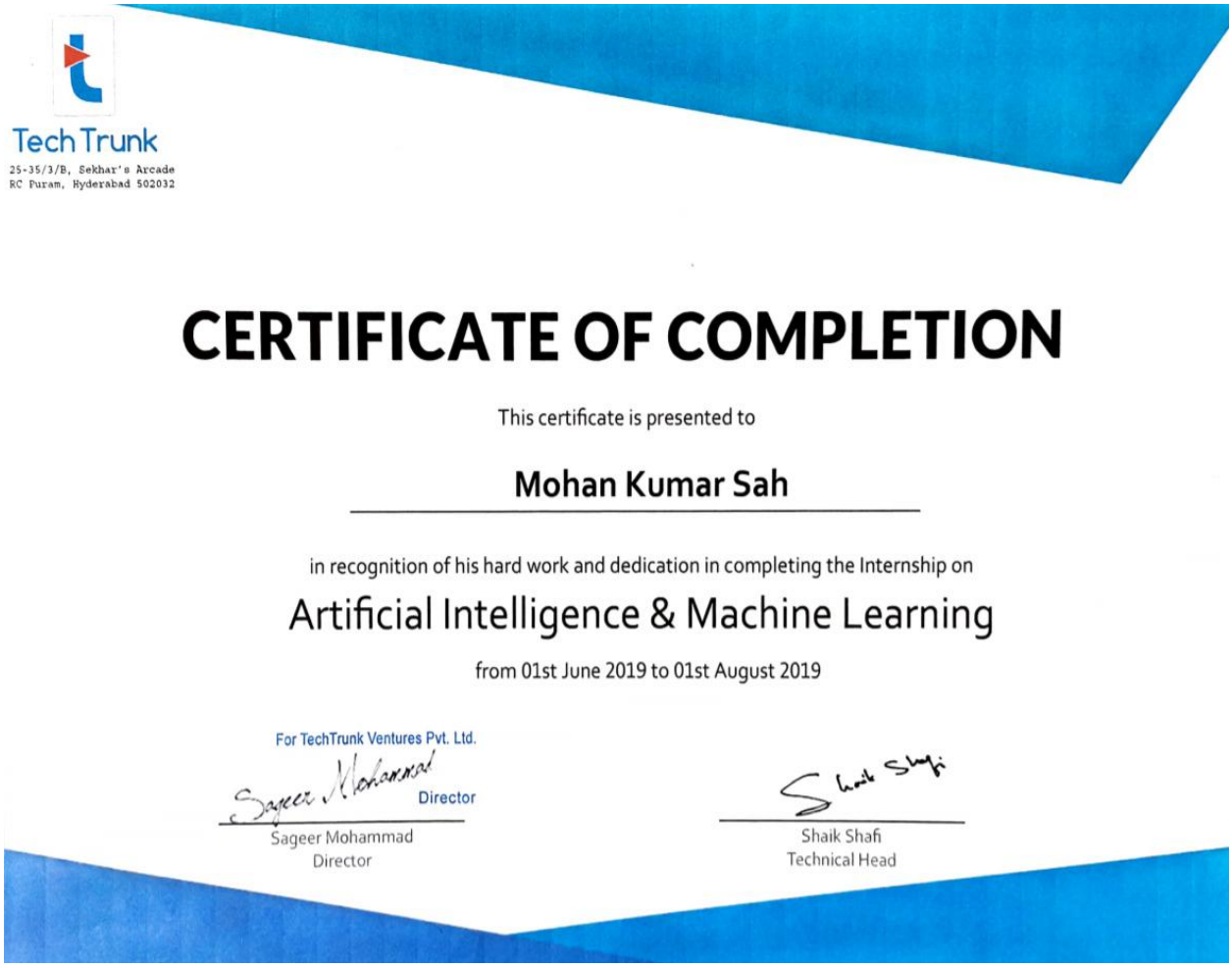
Submitted to

Dept. of Computer Science & Engineering, Tezpur University, Assam

Submitted by:

MOHAN KUMAR SAH
B.Tech (CSE dept. , 7th Sem.)
Roll No: CSB16071
Tezpur Central University, Tezpur, Assam

CERTIFICATE



Tech Trunk
25-35/3/B, Sekhar's Arcade
RC Puram, Hyderabad 502032

CERTIFICATE OF COMPLETION

This certificate is presented to

Mohan Kumar Sah

in recognition of his hard work and dedication in completing the Internship on

Artificial Intelligence & Machine Learning

from 01st June 2019 to 01st August 2019

For TechTrunk Ventures Pvt. Ltd.
Director

Sageer Mohammad
Director

Shaik Shafi
Technical Head

Recommendation Letter

Tech Trunk

www.techtrunk.in
contact@techtrunk.in
CIN NO: U72200TG2015PTC100095

Recommendation Letter

TO WHOM IT MAY CONCERN

Date: August 1st, 2019

We are pleased to write this letter to inform you that the **Mohan Kumar Sah** had been working for us, **TechTrunk Ventures Private Limited**, in his capacity as **AI & Machine Learning Intern** for **2 months** (from 01st June to 1st August 2019).

During his tenure in the office with our Organization, he participated in performing the work with determination and sincerity. As we observed, he was an active and qualified person and he could perform all of assigned tasks effectively. Besides, in my opinion, he was a motivated, devoted, professional and hard-working person. He contributed much to our organizational goals and targets and his performance was proven to be among the most effective in our organization.

Moreover, **Mr. Mohan Kumar Sah** demonstrated excellent behavior and attitude during his service with us. We found him be sincere, truthful, reliable sociable. He was also a pleasant person to talk and work within a team.

We wish his all the success in any path of career.

TechTrunk Ventures Pvt. Ltd.

Director

For TechTrunk Ventures Pvt. Ltd.

Director

Acknowledgement

There are many people who have added to the fruitful culmination of this venture and I seek this opportunity to offer my thanks to each and every one of them.

Above all else I thank the god-like to favor me with the open doors, quality, good faith and support. I thank my folks who have been a consistent wellspring of motivation and consolation.

With incredible joy and thankfulness, I augment my profound feeling of appreciation to **Mr. Sageer Mohammad**, Co-Founder and Director, TechTrunk Venture Private Limited, Hyderabad for giving me the chance to undertake training at this laboratory.

I feel privileged to offer thanks and gratitude to **Mr. Sobin Sunny**, Trainer, TechTrunk Venture Private Limited, Hyderabad for his help and guidance during training.

Lastly I thank all those people who are directly or indirectly related with the completion of this project.

Declaration

I hereby declare that the report on internship as "**Artificial & Machine Learning Intern**" is my original work. This written submission represents my ideas in my own words and wherever others ideas and words have been included, I have adequately cited and provided references to the original sources. I also declare that I have adhered to all the principles of the academic honesty and integrity and have not misrepresented or falsified any idea/data/fact/source in my submission.

MOHAN KUMAR SAH
B.Tech (CSE dept. , 7th Sem.)
Roll No: CSB16071
Tezpur Central University, Tezpur, Assam

Abstract

I carried out my internship at TechTrunk Venture Private Limited, Hyderabad under the guidance of Sunny Sobin, Trainer at TechTrunk Venture Private Limited.

During my internship period a number of approaches and exposure methods were used which included: hands on, through reading relevant materials, and also questions and answer approaches.

I was assigned to different types of works during my internship period which includes:

1. Exploring about the concepts like Feature Engineering, Feature Selection, and Exploratory Data Analysis.
2. Analyzing 10 different data sets using the mentioned concepts and create machine learning model on these data sets.
3. To assign one of the above data set for analysis and model creation to students who came for training in the company and guide them in their work.
4. Building a Talking Chatbot using Chatbot Corpus.

In conclusion, this was an opportunity to develop and enhance skills and competencies in my carrier field which I actually achieved.

CONTENTS

SI No Chapter **Page No.**

CHAPTER 1

About TechTrunk

TechTrunk Venture Private Limited is a company which provides online courses on Artificial Intelligence, Machine Learning and Deep Learning. Its courses cover demo and webinars on how to use AI-based, cloud based AI services that includes Microsoft Azure ML studio, AWS machine learning, AWS sage maker, Azure bot frame work and other cloud platform. Tech trunk also provides online classes through live LED classes and regular class rooms by working professionals and consultants from renowned organizations processing over a decade of experiences. It also provides workshops on University level and delivers workshops across the country. Tech trunk provides better understanding of the concepts through TechTrunk leverages technologies like go to training , portal by LogMeIn digital writing board by Wacom, Google smart digital class room, LM by Moodle , office intelligent cloud lab with AWS and more working with real time case studies like AI based security systems , AI to optimize IT operations chat bots and accessing the requirements for deploying an AI model in terms of infrastructural investment, time and economic benefits etc.

Tech trunk also enables it trainers to work on different IT projects ensuring they gain real time industry experience that helps it to deliver the high quality training.

Message of Director

Our team has a goal to reach out more than 1 billion students and working professionals across the globe by coming 7 years We have done more than 50 Research labs and CoE setups and we have a goal to make this figure 100 times more than the current in next 5 years. Our enthusiastic and passionate team is backed by a team of advisors who have more than 20 years of experience of working as industry leaders, which makes us an expert team to provide best services to our customers. Our vision is to become World's Largest Training service provider and help youngster across the world to build a bright career.

CHAPTER 2

Internship Description

2.1 Internship Profile:

Post	Artificial Intelligence & Machine Learning Intern
Organization	TechTrunk Ventures Private Limited Hyderabad, Telangana, India
Platform	Windows 10
Guide	Sobin Sunny

2.2 Job Assigned:

1. Asked to explore about the concepts like Feature Engineering, Feature Selection, and Exploratory Data Analysis.
2. Asked to analyze 10 different data sets using the mentioned concepts and create machine learning model on these data sets.
3. Asked to assign one of the above data set for analysis and model creation to students who came for training in the company and guide them in their work.
4. Asked to build a Talking Chat bot using corpus data set.

2.3 Hardware Required:

1. CPU - Intel **2.4 GHz** Core i5
2. System Memory – Minimum **1GB RAM and 512 GB Space**
3. Storage - **1 TB** SATA SSD for OS + **3 TB** 7200 rpm HDD for Long-term Data Storage

2.4 Software Required:

1. Anaconda, Jupyter Notebooks

2.5 Language & Libraries Required:

1. **Python** programming language.
2. **Libraries:** numpy, pandas, sklearn, matplotlib, seaborn, keras, speech_recognition, chatterbot, pyttsx3, PyAudio

CHAPTER 3

Introduction to Datasets

A collection of instances is a **dataset** and when working with machine learning methods we typically need a few datasets for different purposes.

1. **Training Dataset**: A dataset that we feed into our machine learning algorithm to train our model.
2. **Testing Dataset**: A dataset that we use to validate the accuracy of our model but is not used to train the model. It may be called the validation dataset.

Datasets on which I worked during the Internship .

1. **Chicago Crime Dataset**

   This dataset reflects reported incidents of crime (with the exception of murders where data exists for each victim) that occurred in the City of Chicago from 2001 to present, minus the most recent days.

   Objective: Explore the data, and provide insights and forecasts about crimes in Chicago.

2. **Sales-Win-Loss Dataset**

   Objective: Explore, analyze data and build a predictive model to tell us which sales campaign will result in a loss and which will result in a win.

3. **Marketing Campaign Dataset**

   Objective: Analyze test market campaigns based on responses, revenue and other key metrics. Predict who will respond to which campaign by which channel and why.

4. **Accounts Receivable Dataset**

   Objective: Understand the factors of successful collection efforts. You can Predict which customers will pay fastest and recover more money and improve collections efficiency.

5. **HR Employee Attrition Dataset**

   Objective: Measure the attrition of the Employees at specific time-gaps. The factors on which the Employee Attrition depends upon are: Age of the Employee, Monthly Income, Overtime, Monthly Rate, Distance from Home, Years at Company and more.

6. **Automobile Dataset**

   This data set consists of three types of entities: (a) the specification of an auto in terms of various characteristics, (b) its assigned insurance risk rating, (c) its normalized losses in use as compared to other cars. The second rating corresponds to the degree to which the auto is more risky than its price indicates. Cars are initially assigned a risk factor symbol associated with its price. Then, if it is more risky (or less), this symbol is adjusted by moving it up (or down) the scale. Actuarians call this process "symboling". A value of +3 indicates that the auto is risky, -3 that it is probably pretty safe.

The third factor is the relative average loss payment per insured vehicle year. This value is normalized for all autos within a particular size classification (two-door small, station wagons, sports/speciality, etc...), and represents the average loss per car per year.
Note: Several of the attributes in the database could be used as a "class" attribute.

Objective: Explore dataset, do feature selection process and predict price of vehicle or car.

7. **Pima Indians diabetes Dataset**

The datasets consist of several medical predictor (independent) variables and one target (dependent) variable, Outcome. Independent variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

Objective: Explore dataset and diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.

8. **Rain in Australia Dataset**

This dataset contains daily weather observations from numerous Australian weather stations.
The target Rain Tomorrow means: Did it rain the next day? Yes or No.
Note: You should exclude the variable Risk-MM when training your binary classification model. If you don't exclude it, you will leak the answers to your model and reduce its predictability.

Objective: Explore data, extract information and Predict whether or not it will rain tomorrow by training a binary classification model on target RainTomorrow.

9. **Financial Dataset**

Paysim synthetic dataset of mobile money transactions . Each step represents an hour of simulation. This dataset is scaled down 1/4 of the original dataset which is presented in the paper "PaySim: A financial mobile money simulator for fraud detection".

Objective: Explore data, extract information and do fraud detection in finance.

10. **IPL Dataset**

The file contains ball by ball details for all matches for all seasons.

Objective: Analyze the IPL data and Predict the winner of the next matches of IPL based on past data, Visualizations, Perspectives, etc.

CHAPTER 4

Introduction to Feature Engineering, Feature Selection & Exploratory Data Analysis

---

4.1 Feature Engineering

Most of the machine learning algorithms works on numerical data (features) so we have to process and covert our data into numeric values.

Feature engineering involves some form of transformation of the categorical values into numeric values and then applying some encoding scheme on these values.

1. Transforming Categorical Attributes

i. Transforming Nominal Attributes

Nominal attributes consist of discrete categorical values with no notion or sense of order amongst them. The idea here is to transform these attributes into a more representative numerical format which can be easily understood by downstream code and pipelines.

ii. Transforming Ordinal Attributes

Ordinal attributes are categorical attributes with a sense of order amongst the values. In general, there is no generic module or function to map and transform these features into numeric representations based on order automatically. Hence we can use a custom encoding\mapping scheme.

2. Encoding Numerical Attributes

i. One-hot Encoding Scheme

Considering we have the numeric representation of any categorical attribute with m labels (after transformation), the one-hot encoding scheme, encodes or transforms the attribute into m binary features which can only contain a value of 1 or 0. Each observation in the categorical feature is thus converted into a vector of size m with only one of the values as 1 (indicating it as active).

ii. Dummy Coding Scheme

The dummy coding scheme is similar to the one-hot encoding scheme, except in the case of dummy coding scheme, when applied on a categorical feature with m distinct labels, we get m - 1 binary features. Thus each value of the categorical variable gets converted into a vector of size m - 1. The extra feature is completely disregarded and thus if the category values range from {0, 1, …, m-1} the 0th or the m - 1th feature column is dropped and corresponding category values are usually represented by a vector of all zeros (0).

iii. Effect Coding Scheme

The effect coding scheme is actually very similar to the dummy coding scheme, except during the encoding process, the encoded features or feature vector, for the category values which represent all 0 in the dummy coding scheme, is replaced by -1 in the effect coding scheme.

Note: The encoding schemes which is mentioned above, work quite well on categorical data in general, but they start causing problems when the number of distinct categories in any feature becomes very large. Essential for any categorical feature of m distinct labels, you get m separate features. This can easily increase the size of the feature set causing problems like storage issues, model training problems with regard to time, space and memory.

Hence we need to look towards other categorical data feature engineering schemes for features having a large number of possible categories (like IP addresses). The bin-counting scheme, feature hashing scheme are some useful schemes for dealing with categorical variables having many categories.

iv. Bin-counting Scheme

In this scheme, instead of using the actual label values for encoding, we use probability based statistical information about the value and the actual target or response value which we aim to predict in our modeling efforts.

v. Feature Hashing Scheme

In this scheme, a hash function is typically used with the number of encoded features pre-set (as a vector of pre-defined length) such that the hashed values of the features are used as indices in this pre-defined vector and values are updated accordingly.

Since a hash function maps a large number of values into a small finite set of values, multiple different values might create the same hash which is termed as collisions. Typically, a signed hash function is used so that the sign of the value obtained from the hash is used as the sign of the value which is stored in the final feature vector at the appropriate index. This should ensure lesser collisions and lesser accumulation of error due to collisions.

Hashing schemes work on strings, numbers and other structures like vectors. You can think of hashed outputs as a finite set of b bins such that when hash function is applied on the same values\categories, they get assigned to the same bin (or subset of bins) out of the b bins based on the hash value. We can pre-define the value of b which becomes the final size of the encoded feature vector for each categorical attribute that we encode using the feature hashing scheme.

Thus even if we have over 1000 distinct categories in a feature and we set b=10 as the final feature vector size, the output feature set will still have only 10 features as compared to 1000 binary features if we used a one-hot encoding scheme.

4.2 Feature Selection

Feature Selection is a technique which selects the most relevant features from the original dataset.

Importance of Feature Selection in Machine Learning

Machine learning works on a simple rule – if you put garbage in, you will only get garbage to come out. By garbage here, I mean noise in data.

This becomes even more important when the number of features is very large. You need not use every feature at your disposal for creating an algorithm. You can assist your algorithm by feeding in only those features that are really important.

Top reasons to use feature selection are:

- It enables the machine learning algorithm to train faster.
- It reduces the complexity of a model and makes it easier to interpret.
- It improves the accuracy of a model if the right subset is chosen.
- It reduces overfitting.

Various methodologies and techniques that you can use to subset your feature space and help your models perform better and efficiently.

1. Filter Methods

Filter methods are generally used as a preprocessing step. The selection of features is independent of any machine learning algorithms.

Feature\Response	Continuous	Categorical
Continuous	Pearson's Correlation	LDA
Categorical	Anova	Chi-Square

- **Pearson's Correlation:** It is used as a measure for quantifying linear dependence between two continuous variables X and Y. Its value varies from -1 to +1. Pearson's correlation is given as:

$$\rho_{X,Y} = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y}$$

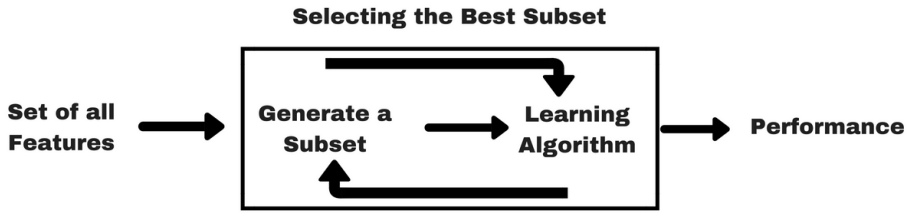
- **LDA:** Linear discriminant analysis is used to find a linear combination of features that characterizes or separates two or more classes (or levels) of a categorical variable.

- **ANOVA:** ANOVA stands for Analysis of variance. It is similar to LDA except for the fact that it is operated using one or more categorical independent features and one continuous dependent feature. It provides a statistical test of whether the means of several groups are equal or not.

- **Chi-Square:** It is a is a statistical test applied to the groups of categorical features to evaluate the likelihood of correlation or association between them using their frequency distribution.

Note: Filter methods do not remove multicollinearity. So, you must deal with multicollinearity of features as well before training models for your data.

2. Wrapper Methods



In wrapper methods, we try to use a subset of features and train a model using them. Based on the inferences that we draw from the previous model, we decide to add or remove features from your subset. The problem is essentially reduced to a search problem. These methods are usually computationally very expensive.

Some common examples of wrapper methods are forward feature selection, backward feature elimination, recursive feature elimination, etc.

- **Forward Selection:** Forward selection is an iterative method in which we start with having no feature in the model. In each iteration, we keep adding the feature which best improves our model till an addition of a new variable does not improve the performance of the model.
- **Backward Elimination:** In backward elimination, we start with all the features and removes the least significant feature at each iteration which improves the performance of the model. We repeat this until no improvement is observed on removal of features.
- **Recursive Feature elimination:** It is a greedy optimization algorithm which aims to find the best performing feature subset. It repeatedly creates models and keeps aside the best or the worst performing feature at each iteration. It constructs the next model with the left features until all the features are exhausted. It then ranks the features based on the order of their elimination.

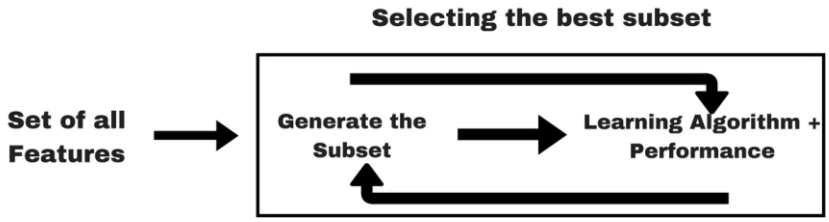
One of the best ways for implementing feature selection with wrapper methods is to use Boruta package that finds the importance of a feature by creating shadow features.

It works in the following steps:

1. Firstly, it adds randomness to the given data set by creating shuffled copies of all features (which are called shadow features).
2. Then, it trains a random forest classifier on the extended data set and applies a feature importance measure (the default is Mean Decrease Accuracy) to evaluate the importance of each feature where higher means more important.
3. At every iteration , it checks whether a real feature has a higher importance than the best of its shadow features (i.e. whether the feature has a higher Z-score than the maximum Z-score of its shadow features) and constantly removes features which are deemed highly unimportant.

4. Finally, the algorithm stops either when all features get confirmed or rejected or it reaches a specified limit of random forest runs.

3. Embedded Methods



Embedded methods combine the qualities' of filter and wrapper methods. It's implemented by algorithms that have their own built-in feature selection methods.

Some of the most popular examples of these methods are LASSO and RIDGE regression which have inbuilt penalization functions to reduce over fitting.

- **Lasso regression** performs L1 regularization which adds penalty equivalent to absolute value of the magnitude of coefficients.
- **Ridge regression** performs L2 regularization which adds penalty equivalent to square of the magnitude of coefficients.

Other examples of embedded methods are Regularized trees, Memetic algorithm, Random multinomial logit.

Difference between Filter and Wrapper methods

The main differences between the filter and wrapper methods for feature selection are:

- Filter methods measure the relevance of features by their correlation with dependent variable while wrapper methods measure the usefulness of a subset of feature by actually training a model on it.
- Filter methods are much faster compared to wrapper methods as they do not involve training the models. On the other hand, wrapper methods are computationally very expensive as well.
- Filter methods use statistical methods for evaluation of a subset of features while wrapper methods use cross validation.
- Filter methods might fail to find the best subset of features in many occasions but wrapper methods can always provide the best subset of features.
- Using the subset of features from the wrapper methods make the model more prone to overfitting as compared to using subset of features from the filter methods.

4.3 Exploratory Data Analysis

Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

It is a good practice to understand the data first and try to gather as many insights from it. EDA is all about making sense of data in hand, before getting them dirty with it.

Different types of analysis:

1. **Univariate** : In univariate analysis we will be using a single feature to analyze almost of its properties.
2. **Bivariate** : When we compare the data between exactly 2 features then it's called bivariate analysis.
3. **Multivariate**: Comparing more than 2 variables is called as Multivariate analysis.

Some of the most important exploratory data analysis methods are **Scatter plot, Pair plot, Box plot, Violin plot, Joint plot, Line plot, Kerner Density Estimate (KDE) plot, Point plot** etc.

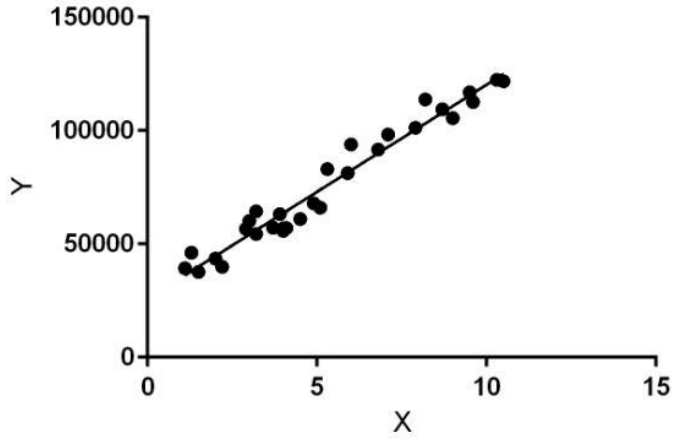
All the mentioned analysis methods are explained with examples in chapter 6.

CHAPTER 5

Introduction to Machine Learning Algorithms

5.1. Linear Regression

Linear Regression is a machine learning algorithm based on supervised learning. It performs a regression task. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.



Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.

In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

Hypothesis function for Linear Regression :

$$y = \theta_1 + \theta_2.x$$

While training the model we are given :
x: input training data (univariate – one input variable(parameter))
y: labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.
θ1: intercept
θ2: coefficient of x

Once we find the best θ1 and θ2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

How to update θ1 and θ2 values to get the best fit line ?

Cost Function (J):
By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ1 and θ2 values, to reach the best value that minimize the error between predicted y value (pred) and true y value (y).

$$minimize \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2 \qquad J = \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

Cost function(J) of Linear Regression is the Root Mean Squared Error (RMSE) between predicted y value (pred) and true y value (y).

Gradient Descent:
To update θ1 and θ2 values in order to reduce Cost function (minimizing RMSE value) and achieving the best fit line the model uses Gradient Descent. The idea is to start with random θ1 and θ2 values and then iteratively updating the values, reaching minimum cost.

How to use gradient descent to update θ1 and θ2 . To update θ1 and θ2 , we take gradients from the cost function. To find these gradients, we take partial derivatives with respect to θ1 and θ2 .

$$J = \frac{1}{n}\sum_{i=1}^{n}(pred_i - y_i)^2$$

$$J = \frac{1}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i)^2$$

$$\frac{\partial J}{\partial a_0} = \frac{2}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i) \implies \frac{\partial J}{\partial a_0} = \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i)$$

$$\frac{\partial J}{\partial a_1} = \frac{2}{n}\sum_{i=1}^{n}(a_0 + a_1 \cdot x_i - y_i) \cdot x_i \implies \frac{\partial J}{\partial a_1} = \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i) \cdot x_i$$

$$a_0 = a_0 - \alpha \cdot \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i)$$

$$a_1 = a_1 - \alpha \cdot \frac{2}{n}\sum_{i=1}^{n}(pred_i - y_i) \cdot x_i$$

The partial derivates are the gradients and they are used to update the values of θ1 and θ2 . Alpha is the learning rate which is a hyper parameter that you must specify. A smaller learning rate could get you closer to the minima but takes more time to reach the minima, a larger learning rate converges sooner but there is a chance that you could overshoot the minima.

5.2. Logistic Regression

Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability.

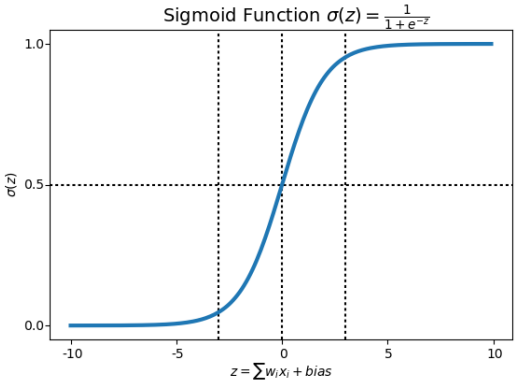
We can call a Logistic Regression a Linear Regression model but the Logistic Regression uses a more complex cost function, this cost function can be defined as the 'Sigmoid function' or also known as the 'logistic function' instead of a linear function.

The hypothesis of logistic regression tends it to limit the cost function between 0 and 1. Therefore linear functions fail to represent it as it can have a value greater than 1 or less than 0 which is not possible as per the hypothesis of logistic regression.

$$0 \leq h_\theta(x) \leq 1$$

What is the Sigmoid Function?

In order to map predicted values to probabilities, we use the Sigmoid function. The function maps any real value into another value between 0 and 1.



Sigmoid Function Graph

$$f(x) = \frac{1}{1+e^{-(x)}}$$

Hypothesis Representation

$\sigma(Z) = \sigma(\beta_o + \beta_1 X)$

We have expected that our hypothesis will give values between 0 and 1.

$Z = \beta_o + \beta_1 X$
$h\Theta(x) = sigmoid(Z)$ i.e. $h\Theta(x) = 1/(1 + e^{-(\beta_o + \beta_1 X)})$

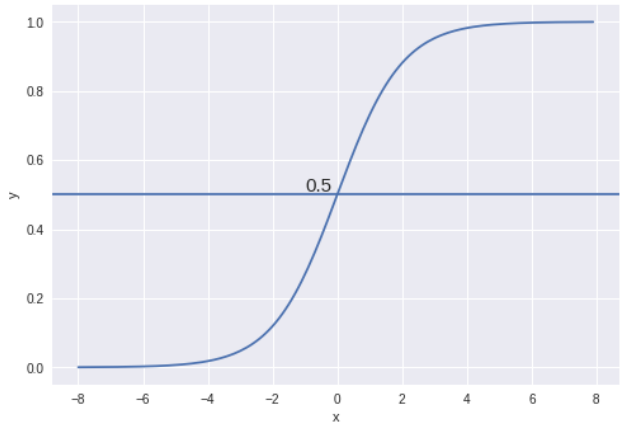
$$h\theta(X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

The Hypothesis of logistic regression

Decision Boundary

We expect our classifier to give us a set of outputs or classes based on probability when we pass the inputs through a prediction function and return a probability score between 0 and 1.

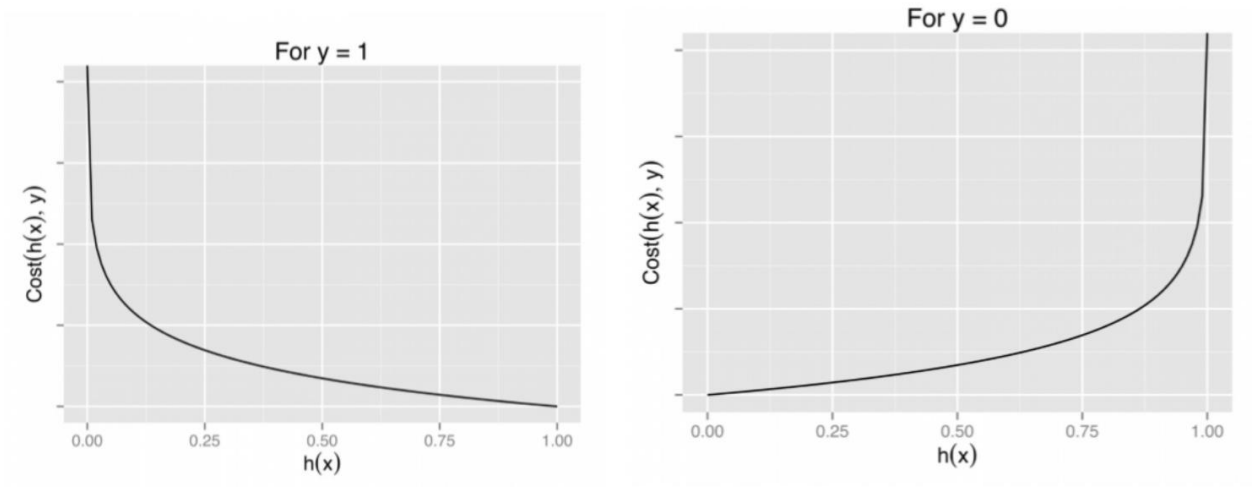
For Example, We have 2 classes, let's take them like cats and dogs(1 — dog , 0 — cats). We basically decide with a threshold value above which we classify values into Class 1 and of the value goes below the threshold then we classify it in Class 2.



As shown in the above graph we have chosen the threshold as 0.5, if the prediction function returned a value of 0.7 then we would classify this observation as Class 1(DOG). If our prediction returned a value of 0.2 then we would classify the observation as Class 2(CAT).

For logistic regression, the Cost function is defined as:

$$Cost(h_\theta(x), y) = \begin{cases} -log(h_\theta(x)) & \text{if } y = 1 \\ -log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



Graph of logistic regression

The above two functions can be compressed into a single function i.e.

$$J(\theta) = -\frac{1}{m}\sum\left[y^{(i)}\log(h\theta(x(i))) + \left(1-y^{(i)}\right)\log(1-h\theta(x(i)))\right]$$

Gradient Descent:
Now the question arises, **how do we reduce the cost value?** Well, this can be done by using Gradient Descent. The main goal of Gradient descent is to minimize the cost value. i.e. min J(θ).

Now to minimize our cost function we need to run the gradient descent function on each parameter i.e.

$$\theta j := \theta j - \alpha \frac{\partial}{\partial \theta j} J(\theta)$$

Objective: To minimize the cost function we have to run the gradient descent function on each parameter.

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

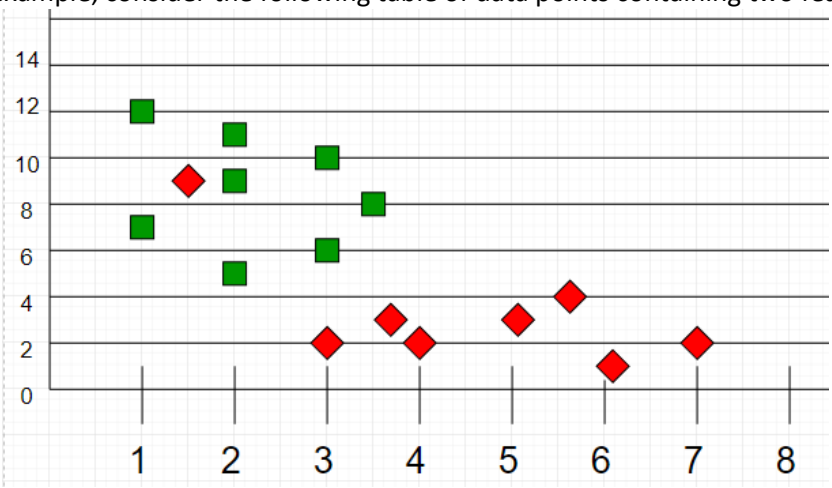
$\}$

5.3. K-Nearest Neighbours

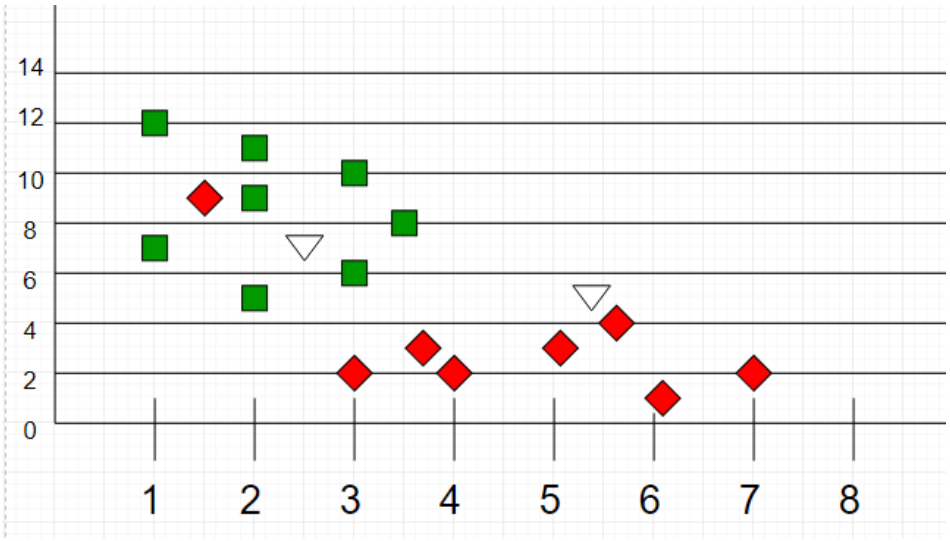
K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.

We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

As an example, consider the following table of data points containing two features:



Now, given another set of data points (also called testing data), allocate these points a group by analyzing the training set. Note that the unclassified points are marked as 'White'.



Intuition

If we plot these points on a graph, we may be able to locate some clusters or groups. Now, given an unclassified point, we can assign it to a group by observing what group its nearest neighbors belong to. This means a point close to a cluster of points classified as 'Red' has a higher probability of getting classified as 'Red'.

Intuitively, we can see that the first point (2.5, 7) should be classified as 'Green' and the second point (5.5, 4.5) should be classified as 'Red'.

Algorithm

Let m be the number of training data samples. Let p be an unknown point.

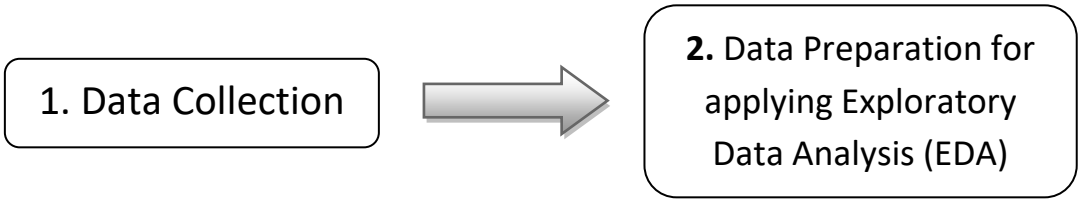
1. Store the training samples in an array of data points arr[]. This means each element of this array represents a tuple (x, y).
2. for i=0 to m:
3. Calculate Euclidean distance d(arr[i], p).
4. Make set S of K smallest distances obtained. Each of these distances corresponds to an already classified data point.
5. Return the majority label among S.

CHAPTER 6

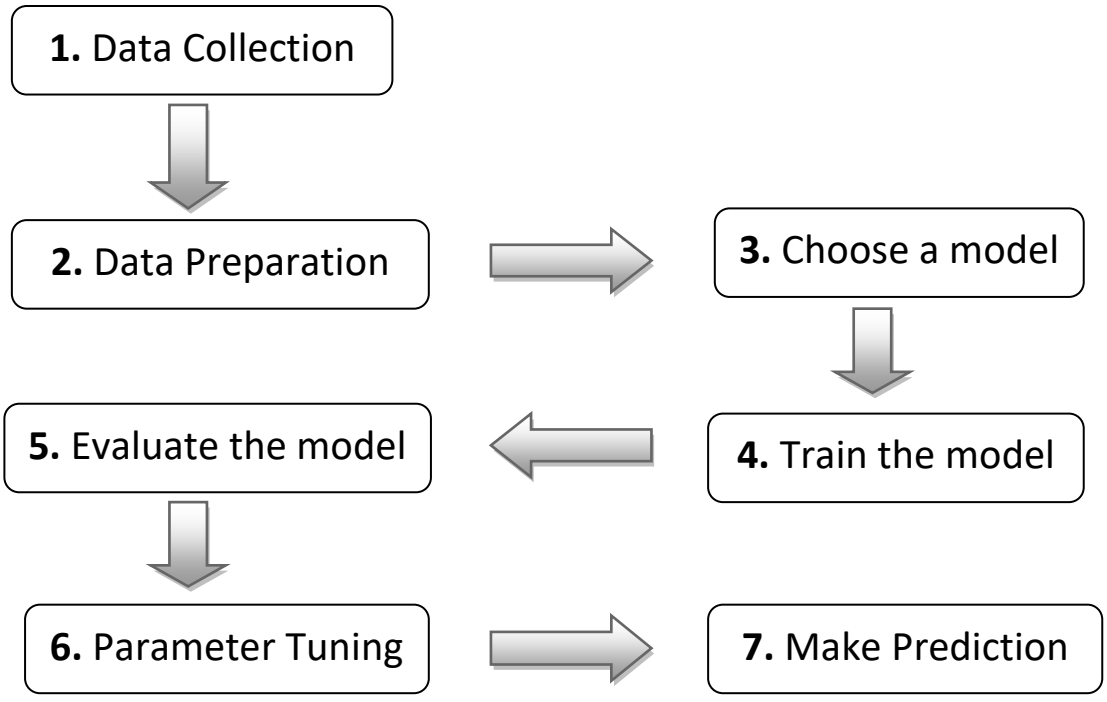
Implementation of ML algorithms on different datasets

The company assigned me two types of work:
Type 1: I have to apply Exploratory Data Analysis (EDA) on some datasets.
Type 2: I have to explore, analyze and build Machine Learning models on some datasets.

Methodology for work "Type 1"



Methodology for work "Type 2"



Explanation of steps:

1. **Data Collection**
   - The quantity & quality of your data dictate how accurate our model is

- The outcome of this step is generally a representation of data (Guo simplifies to specifying a table) which we will use for training
- Using pre-collected data, by way of datasets from Kaggle, UCI, etc., still fits into this step

2. Data Preparation

- Wrangle data and prepare it for training
- Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, and data type conversions, etc.)
- Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data
- Visualize data to help detect relevant relationships between variables or class imbalances (bias alert!), or perform other exploratory analysis
- Split into training and evaluation sets

3. Choose a Model

- Different algorithms are for different tasks; choose the right one

4. Train the Model

- The goal of training is to answer a question or make a prediction correctly as often as possible
- Linear regression example: algorithm would need to learn values for *m* (or *W*) and *b* (*x* is input, *y* is output)
- Each iteration of process is a training step

5. Evaluate the Model

- Uses some metric or combination of metrics to "measure" objective performance of model
- Test the model against previously unseen data
- This unseen data is meant to be somewhat representative of model performance in the real world, but still helps tune the model (as opposed to test data, which does not)
- Good train/eval split? 80/20, 70/30, or similar, depending on domain, data availability, dataset particulars, etc.
-

6. Parameter Tuning

- This step refers to *hyperparameter* tuning, which is an "artform" as opposed to a science
- Tune model parameters for improved performance
- Simple model hyperparameters may include: number of training steps, learning rate, initialization values and distribution, etc.

7. Make Predictions

- Using further (test set) data which have, until this point, been withheld from the model (and for which class labels are known), are used to test the model; a better approximation of how the model will perform in the real world

Since I have worked on several datasets, now I am showing the screen shots of my code and corresponding outputs for two datasets, one dataset belongs to work "Type1" and other belongs to work "Type2".

Note: All the codes are written in Jupyter Notebook Software and all the screen shots are taken from the same.

Exploratory Data Analysis Seaborn Tips Dataset

In [1]:
```
# Importing necessary libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Loading the dataset
tips = sns.load_dataset("tips")
# Printing the first five rows
tips.head(5)
```

Out[2]:

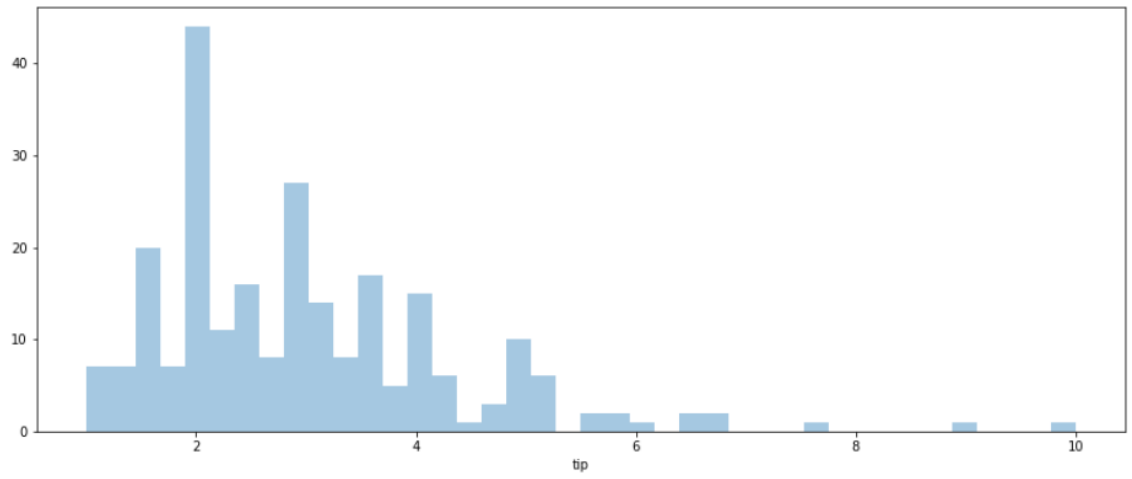
	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

Univariate plots

1. Histogram

A **histogram** is a graphical representation that organizes a group of data points into user-specified ranges.

In [7]:
```
# Using seaborn
plt.figure(figsize=(15,6))
sns.distplot(tips['tip'], kde=False, bins=40);
```

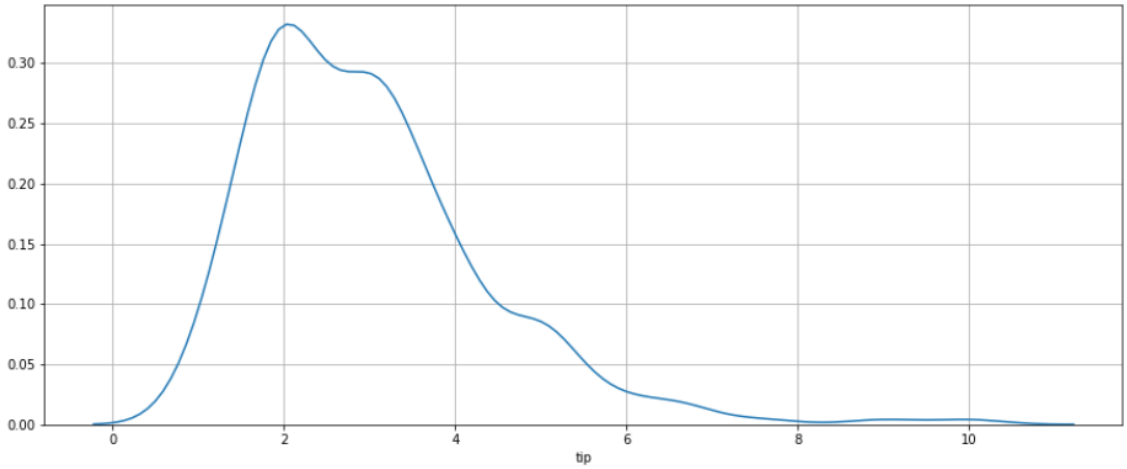


We can see that the count of different tip value present in the dataset and infer that most of the tips are between 2 and 4.

2. Kerner Density Estimate (KDE)

KDE is a way to estimate the probability density function of a continuous random variable. It is used when you need to know the distribution of the variable.

```
In [8]: plt.figure(figsize=(15,6))
        plt.grid()
        sns.distplot(tips['tip'],hist=False, bins=40);
```



We can see the the probabilty for tip value 2 is maximum approx 0.35

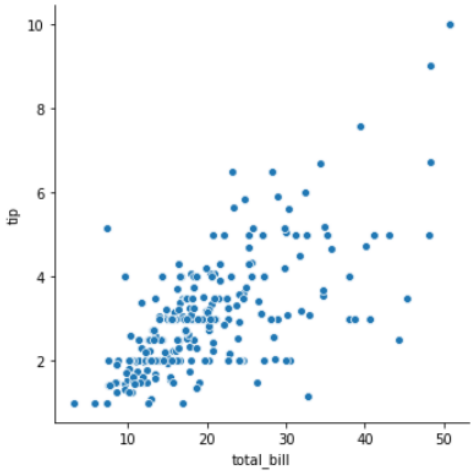
Bivariate Plots

1. Scatterplot

It shows the relationship between two variables.

```
In [10]: # Using seaborn
         sns.relplot(x="total_bill", y="tip", data=tips)
```

```
Out[10]: <seaborn.axisgrid.FacetGrid at 0x25d4b5a30f0>
```



we can see that if total_bill is between 10–30 then the tip will be mostly above 2

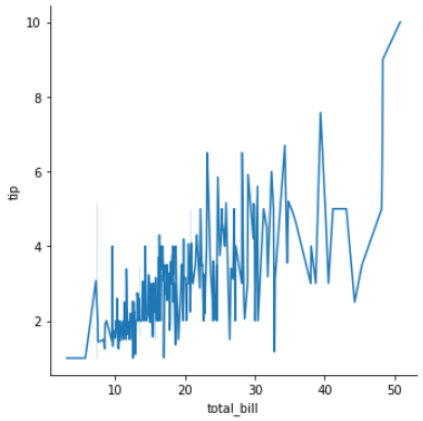
We can add the third variable also in scatterplot using different colors or shape of dots.For this use seaborn library.

2. Lineplot

This plot is similar to the scatterplot but instead of dots, it displays the line joining all the dots by arranging the variable value represented on the x-axis.

```
In [12]: # Using seaborn
         sns.relplot(x="total_bill", y="tip", kind="line", data=tips)
```

```
Out[12]: <seaborn.axisgrid.FacetGrid at 0x25d4b95eb38>
```



Line plot arranges the rows as per total_bill and then joins the points

3. Boxplot

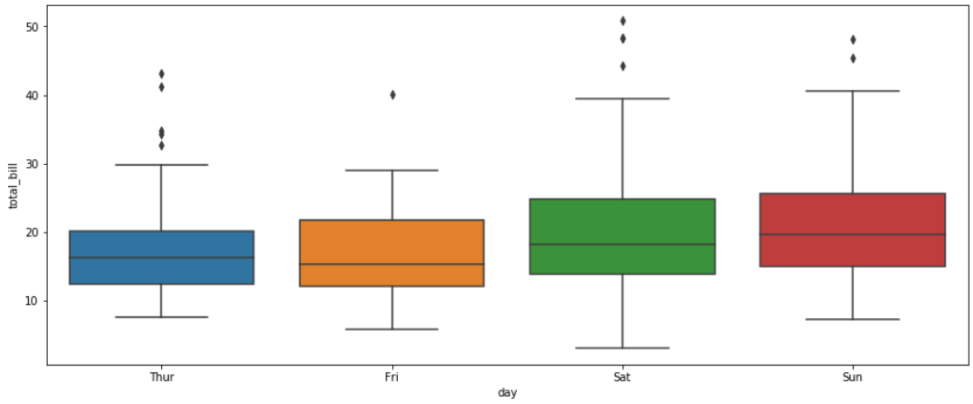
Boxplot can give all the stats provided in dataset .describe in a single plot. If the dataset is too large and the range of value is too big then it shows some values as outliers based on an inter-quartile function.

A box plot consists of **5 things**: **Minimum**, **First Quartile or 25%**, **Median (Second Quartile) or 50%**, **Third Quartile or 75%** and **Maximum**.

The first quartile (Q1) is defined as the middle number between the smallest number and the median of the data set. The second quartile (Q2) is the median of the data. The third quartile (Q3) is the middle value between the median and the highest value of the data set.

```
In [15]: plt.figure(figsize=(15,6))
         sns.boxplot(x="day", y="total_bill", data=tips)
```

```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x25d4ba15c50>
```



Let's take the first box plot i.e, blue box plot of the figure and understand these statistical things:

1. Bottom black horizontal line of blue box plot is minimum value
2. First black horizontal line of rectangle shape of blue box plot is First quartile or 25%
3. Second black horizontal line of rectangle shape of blue box plot is Second quartile or 50% or median.
4. Third black horizontal line of rectangle shape of blue box plot is third quartile or 75%
5. Top black horizontal line of rectangle shape of blue box plot is maximum value.
6. Small diamond shape of blue box plot is outlier data or erroneous data.

Note: From First quartile to Third quartile is called Interquartile Range

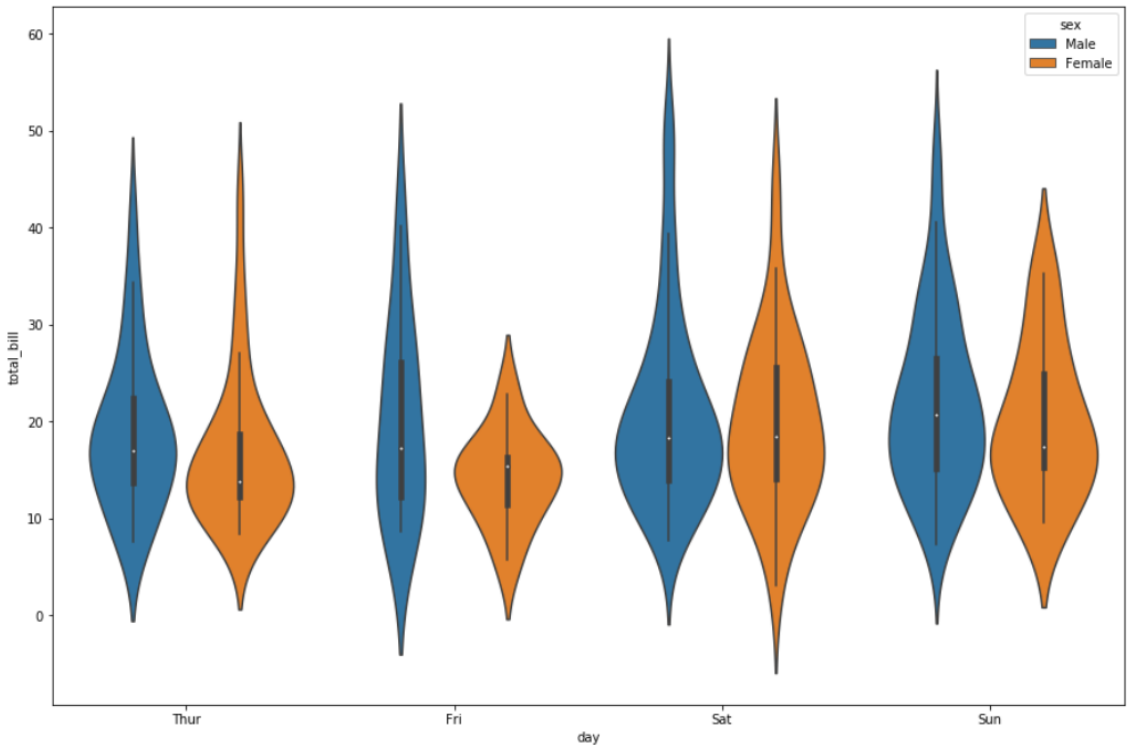
About 25% of data is lower than 13 and about 75% of data is above 13.

4. Violin Plot

This plot is used to visualize the distribution of the data and its probability density. This chart is a combination of a Box Plot and a Density Plot. So if you need to find the frequency distribution along with box plot use violin plot.

```
In [17]: plt.figure(figsize=(15,10))
         sns.violinplot(x="day", y="total_bill", hue="sex", data=tips)
```

```
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x25d4b52c828>
```



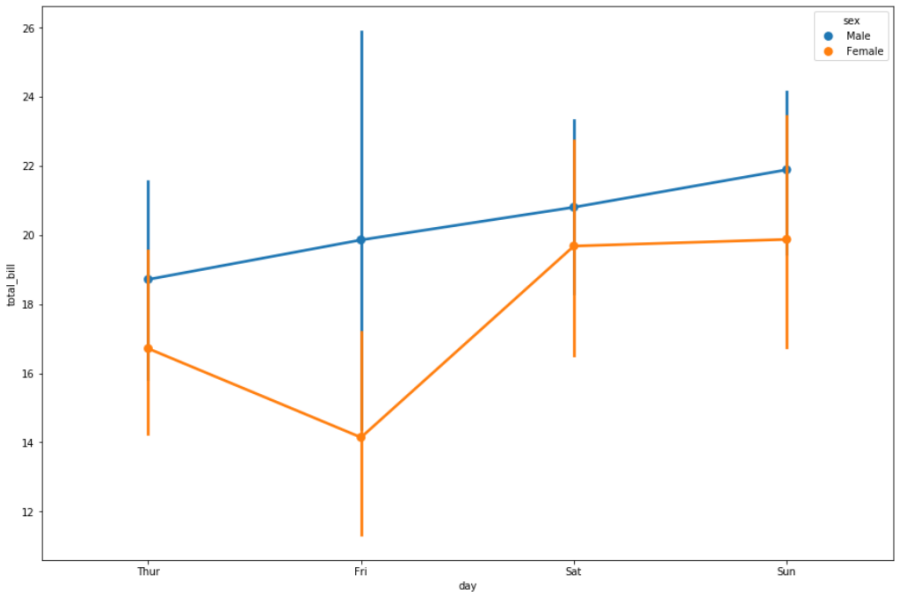
Consider for Thursday, there are more number of female is in between median to first quartile and there are almost equal number of male is in between first quartile to median and median to 3rd quartile.

On Friday we can see that female's total_bill is much less than male's total_bill.

5. Point plots

A point plot represents an estimate of central tendency for a numeric variable by the position of scatter plot points and provides some indication of the uncertainty around that estimate using error bars. Point plot shows only mean values and error rate surrounding those mean values. They are not very much informative but are easy to find the change in a variable based on different categories.

```
In [18]: plt.figure(figsize=(15,10))
         sns.pointplot(x="day", y="total_bill", hue="sex", data=tips)
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x25d4b518dd8>
```



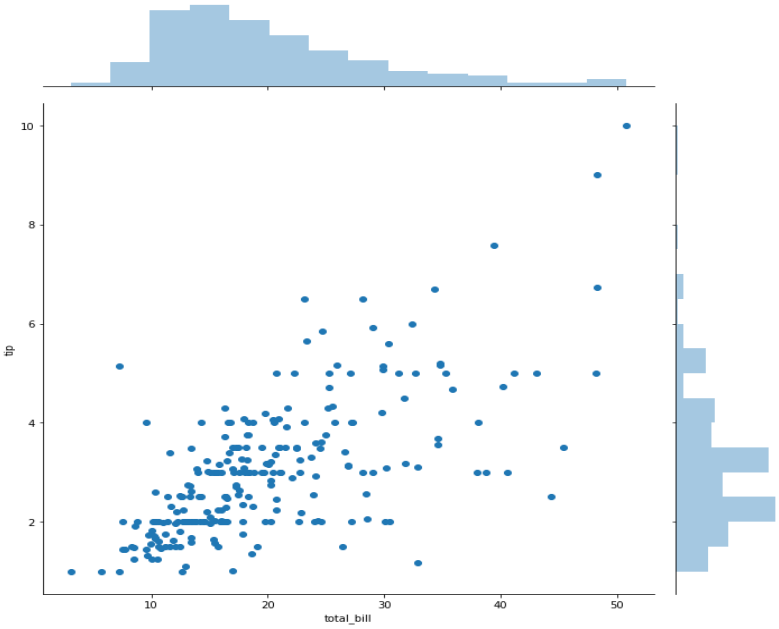
Using this plot it's so simple to find changes in total_bill according to days.

The total_bill is rising for male's as the weekend arises while it decreases on Friday for females and jumps on Saturday and remains mostly constant on Sunday.

6. Joint plot

In a single figure we can do both univariate as well as bivariate analysis. The main plot will give us a bivariate analysis, whereas on the top and right side we will get univariate plots of both the variables that were considered.

```
In [19]: sns.jointplot(y="tip", x="total_bill", data=tips, size=10)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\axisgrid.py:2262: UserWarning: The `size` paramter has
been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
Out[19]: <seaborn.axisgrid.JointGrid at 0x25d4b838828>
```



There are variety of option you can choose from, which can be tuned using kind parameter in seaborn's jointplot function.

Multivariate Plot

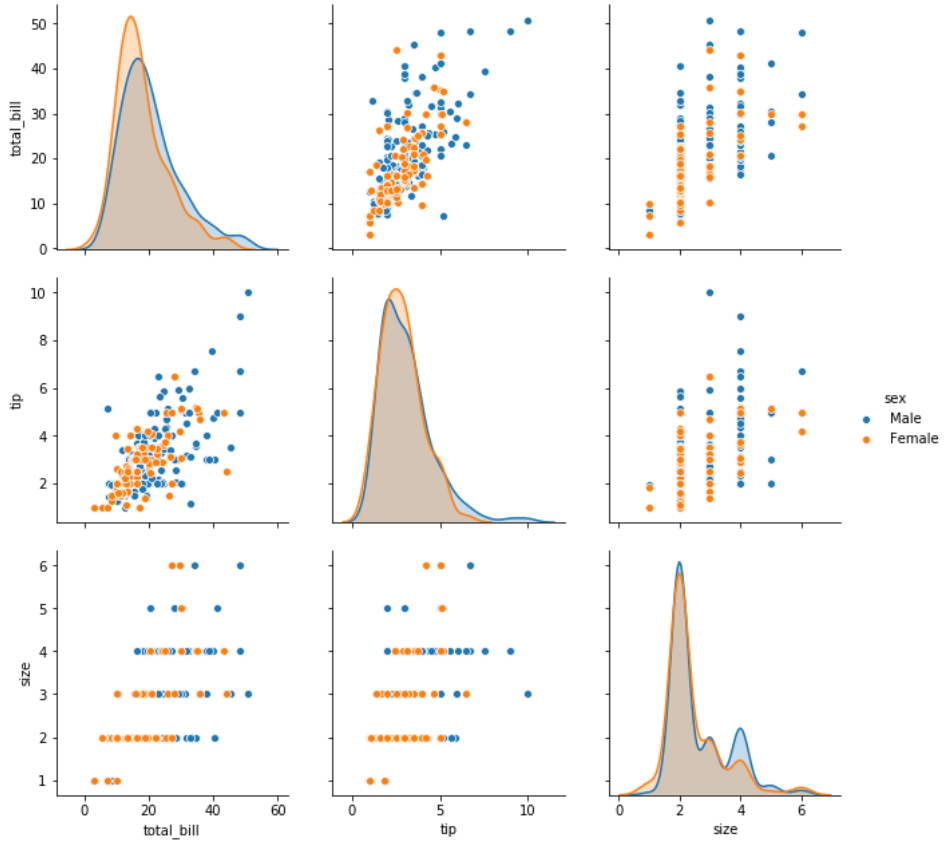
1. Pair Plot

Pair plot will create us a (n x n) figure where the diagonal plots will be histogram plot of the feature corresponding to that row and rest of the plots are the combination of feature from each row in y axis and feature from each column in x axis.

```
In [23]: sns.pairplot(tips, hue="sex", size=3)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\axisgrid.py:2065: UserWarning: The `size` parameter has
been renamed to `height`; pleaes update your code.
  warnings.warn(msg, UserWarning)
```

```
Out[23]: <seaborn.axisgrid.PairGrid at 0x25d4ba260b8>
```



By getting a high level overview of plots from pair plot, we can see which two features can well explain/separate the data and then we can use scatter plot between those 2 features to explore further.

From the above plot, we are not able to find variables which seperate the data.

Since we will be getting n x n plots for n features, pairplot may become complex when we have more number of feature say like 10 or so on. So in such cases, the best bet will be using a dimensionality reduction technique to map data into 2d plane and visualizing it using a 2d scatter plot.

Model Creation on Pima Indians diabetes Dataset

In [1]:
```
# importing necessary packages

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# Loading the csv file
df=pd.read_csv("diabetes.csv")
# Printing first 5 rows
df.head(5)
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [3]:
```
# Printing dimension of dataset (Rows, Columns)
df.shape
```

Out[3]:
```
(768, 9)
```

In [4]:
```
# Information about Null Values and data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
Pregnancies                 768 non-null int64
Glucose                     768 non-null int64
BloodPressure               768 non-null int64
SkinThickness               768 non-null int64
Insulin                     768 non-null int64
BMI                         768 non-null float64
DiabetesPedigreeFunction    768 non-null float64
Age                         768 non-null int64
Outcome                     768 non-null int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

We can see that all columns have 768 values means there are no Null value in any columns.

Creating Model

In [11]:
```
# Splitting the Dataset into training and testing part

x=df.drop("Outcome", axis=1)
y=df['Outcome']

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=5)
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(614, 8)
(614,)
(154, 8)
(154,)
```

In [12]:
```
# Standardising the data
# Before making any actual predictions, it is always a good practice to scale the features so that all of them can be uniformly evaluated

scaler = StandardScaler()
scaler.fit(x_train)

x_train = scaler.transform(x_train)
x_test = scaler.transform(x_test)
```

In [13]:
```
# Logistic Regression model
from sklearn.linear_model import LogisticRegression

# Creating object
model=LogisticRegression()

# Training the model
model.fit(x_train,y_train)

# Prediction
y_pred=model.predict(x_test)
y_pred
```

Out[13]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1],
      dtype=int64)
```

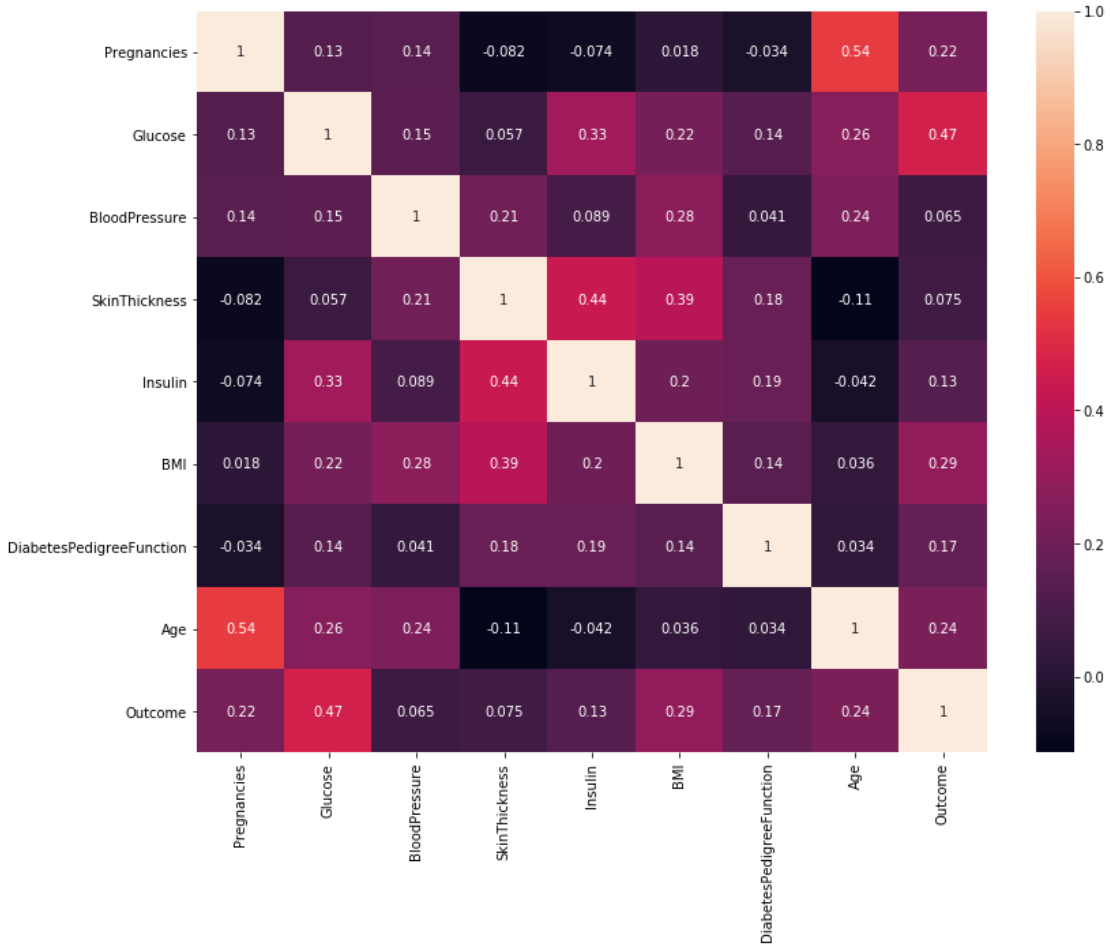
In [14]:
```
print("Accuracy of Logistic Regression is {:.2f} %" .format(model.score(x_test,y_test)*100))
```

```
Accuracy of Logistic Regression is 79.87 %
```

Feature Selection

In [15]:
```
plt.figure(figsize=(13,10))
sns.heatmap(df.corr(), annot=True)
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x1f88edacba8>



From the Heatmap we can say that BloodPressure and SkinThickness are very less correlated with Outcome so we can remove them.

In [16]:
```
df.drop(["BloodPressure","SkinThickness"], axis=1, inplace=True)
df.head(5)
```

Out[16]:

	Pregnancies	Glucose	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	0	33.6	0.627	50	1
1	1	85	0	26.6	0.351	31	0
2	8	183	0	23.3	0.672	32	1
3	1	89	94	28.1	0.167	21	0
4	0	137	168	43.1	2.288	33	1

Creating Model With Selected Features

In [17]:
```
# Splitting the Dataset into training and testing part

x1=df.drop("Outcome", axis=1)
y1=df['Outcome']

x1_train,x1_test,y1_train,y1_test=train_test_split(x1,y1,test_size=0.2,random_state=5)
print(x1_train.shape)
print(y1_train.shape)
print(x1_test.shape)
print(y1_test.shape)

# Standardising the data
# Before making any actual predictions, it is always a good practice to scale the features so that all of
them can be uniformly evaluated

scaler = StandardScaler()
scaler.fit(x1_train)

x1_train = scaler.transform(x1_train)
x1_test = scaler.transform(x1_test)

# Logistic Regression model
from sklearn.linear_model import LogisticRegression

# Creating object
model1=LogisticRegression()

# Training the model
model1.fit(x1_train, y1_train)

# Prediction
y1_pred=model1.predict(x1_test)
y1_pred
```

```
(614, 6)
(614,)
(154, 6)
(154,)
```

Out[17]:
```
array([0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1],
      dtype=int64)
```

In [18]:
```
print("Accuracy of Logistic Regression is {:.2f} %" .format(model1.score(x1_test,y1_test)*100))
```

```
Accuracy of Logistic Regression is 81.17 %
```

We can see that by doing feature selection accuracy increases from 79.87 % to 81.17 %.

CHAPTER 7

Introduction to Talking Chatbot & Chatbot Corpus

What is a Chatbot?

A chatbot is artificial intelligence (AI) software that can simulate a conversation (or a chat) with a user in natural language through messaging applications, websites, and mobile apps or through the telephone.

Why Chatbots are important?

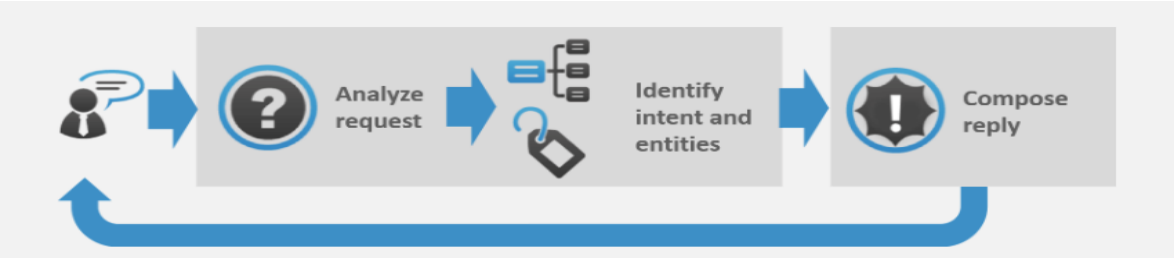
Chatbot applications streamline interactions between people and services, enhancing customer experience. At the same time, they offer companies new opportunities to improve the customer's engagement process and operational efficiency by reducing the typical cost of customer service.

To be successful, a chatbot solution should be able to effectively perform both of these tasks. Human support plays a key role here: Regardless of the kind of approach and the platform, human intervention is crucial in configuring, training and optimizing the chatbot system.

Behind the scenes: How a Chatbot works

There are two different tasks at the core of a chatbot:

1. User request analysis
2. Returning the response



How a Chatbot Works: *As you can see in this graphic, a chatbot returns a response based on input from a user. This process may look simple; in practice, things are quite complex.*

User request analysis: this is the first task that a chatbot performs. It analyzes the user's request to **identify the user intent** and to **extract relevant entities**.

Which chatbot application is right for you?

There are different approaches and tools that you can use to develop a chatbot. Depending on the use case you want to address, some chatbot technologies are more appropriate than others. In order to achieve the desired results, the combination of different AI forms such as natural language processing, machine learning and semantic understanding may be the best option.

What is a Talking chatbot?

It is similar to the chatbot only difference is that it takes voice as input and also gives voice as output.

What is Chatbot Corpus ?

This is a corpus of dialog data that is included in the chatterbot module.

CHAPTER 8

Implementation of Talking Chatbot

Logic used to build the Talking Chatbot

I developed my own logic to build the Talking Chatbot which mainly consist of three steps :
First we have to develop a chatbot which can take the text and reply with text (Used in step 2).

1. Converting the speech (user voice) into text
   - ✓ Here first we recognize the voice of the user which is the input of the Talking Chatbot and after we convert the voice into string (text) for the next step.
2. Feed the text to chatbot which will give a reply to that text
3. Converting the reply into speech (chatbot voice)
   - ✓ Here we convert the text again to voice which is the output of the Talking Chatbot.

In python code I put these three steps in a infinite while loop and set a condition that if user say “bye” then loop breaks and program terminates.

Python Code

Importing necessary packages

```
In [5]: from chatterbot import ChatBot
        from chatterbot.trainers import ChatterBotCorpusTrainer
        import speech_recognition as sr
        import pyttsx3
```

Chatbot which can take the text and reply with text

```
# Making an object of Chatbot and Setting the name of chatbot Mohan
chatbot = ChatBot('Mohan')

# Set the ChatterBotCorpusTrainer as trainer which allow to train the chatbot
chatbot.set_trainer(ChatterBotCorpusTrainer)

# Train the chatbot using data from the ChatterBot dialog corpus
chatbot.train("chatterbot.corpus.english")
```

```
ai.yml Training: [####################] 100%
botprofile.yml Training: [####################] 100%
computers.yml Training: [####################] 100%
conversations.yml Training: [####################] 100%
emotion.yml Training: [####################] 100%
food.yml Training: [####################] 100%
gossip.yml Training: [####################] 100%
greetings.yml Training: [####################] 100%
history.yml Training: [####################] 100%
humor.yml Training: [####################] 100%
literature.yml Training: [####################] 100%
money.yml Training: [####################] 100%
movies.yml Training: [####################] 100%
politics.yml Training: [####################] 100%
psychology.yml Training: [####################] 100%
science.yml Training: [####################] 100%
sports.yml Training: [####################] 100%
trivia.yml Training: [####################] 100%
```

Talking Chatbot

In [6]:

```
while True:
    #Initialize the recognizer
    r = sr.Recognizer()
    #use the microphone as source for input
    with sr.Microphone() as source:
        print("Speak:")
        #listens for the user's input
        audio = r.listen(source)


    try:
        #Recognize the user's input
        text=r.recognize_google(audio)
        print("You said:",text)
        if(text=="bye"):
            reply="Thankyou for talking, Bye"
            break
        # getting the response for the text
        reply=str(chatbot.get_response(text))
    #error occurs when google could not understand what was said
    except:
        reply ="Could not understand the audio "


    print("Reply:",reply)

    # object creation
    engine = pyttsx3.init()
    # saying the text assign to the variable reply
    engine.say(reply)
    # waiting till the saying process is completed
    engine.runAndWait()
```

Step 1

Step 2

Step3

```
Speak:
You said: hello
Reply: Hi
Speak:
You said: hi
Reply: How are you doing?
Speak:
You said: I am doing well
Reply: That is good to hear
Speak:
You said: who are you
Reply: I am just an artificial intelligence.
Speak:
You said: bye
```

CHAPTER 9

Conclusion

During the internship period of two months (01st June, 2019 to 01st Aug, 2019), I learned a lot in the field of Machine Learning, Deep Learning and Data Science. I learned something which I never learned before. Apart from knowledge, I earned good industrial experiences. I learned how to deal with pressure that we are normally facing in office. I learned how to deal with clients and how to talk with clients on any projects. Since I was asked to guide and help some students in their projects, I also gained mentoring experiences. By talking to students or clearing doubts of students, I learned how to teach or guide someone or help someone in their projects. I also developed my communication skills.

At last I would like to inform you that I was successfully delivering my work whether it was about working on projects or mentoring the students. I successfully completed all the work assigned by company on time. Also the Director of the company provides me the **Certificate** and **Recommendation Letter** for my work and said I did well and he is happy with my work.

CHAPTER 10

Bibliography

- ✓ https://botpress.io/learn/what-and-why
- ✓ https://expertsystem.com/chatbot/
- ✓ https://towardsdatascience.com/understanding-feature-engineering-part-2-categorical-data-f54324193e63
- ✓ https://blog.myyellowroad.com/using-categorical-data-in-machine-learning-with-python-from-dummy-variables-to-deep-category-66041f734512
- ✓ https://www.analyticsvidhya.com/blog/2016/12/introduction-to-feature-selection-methods-with-an-example-or-how-to-select-the-right-variables/
- ✓ http://www.statgraphics.com/exploratory-data-analysis
- ✓ https://medium.com/datadriveninvestor/build-your-first-chatbot-in-10-lines-of-code-88c4f15e39c9
- ✓ https://chatterbot.readthedocs.io/en/stable/corpus.html
- ✓ https://www.geeksforgeeks.org/ml-linear-regression/
- ✓ https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-regression-14c4e325882a
- ✓ https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148
- ✓ https://www.geeksforgeeks.org/k-nearest-neighbours/
- ✓ https://www.geeksforgeeks.org/k-nearest-neighbours/
- ✓ https://www.kdnuggets.com/2018/05/general-approaches-machine-learning-process.html