Predicting the Winning Football Team

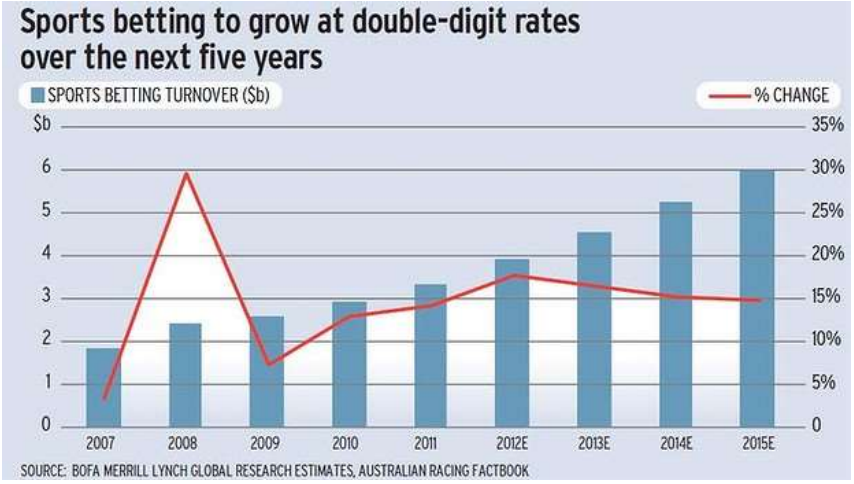
Can we design a predictive model capable of accurately predicting if the home team will win a football match?

Steps

1. We will clean our dataset
2. Split it into training and testing data (12 features & 1 target (winning team (Home/Away/Draw))
3. Train 3 different classifiers on the data -Logistic Regression -Support Vector Machine -XGBoost
4. Use the best Classifer to predict who will win given an away team and a home team

History

Sports betting is a 500 billion dollar market (Sydney Herald)



Kaggle hosts a yearly competiton called March Madness
https://www.kaggle.com/c/march-machine-learning-mania-2017/kernels

Several Papers on this
https://arxiv.org/pdf/1511.05837.pdf

"It is possible to predict the winner of English county twenty twenty cricket games in almost two thirds of instances."
https://arxiv.org/pdf/1411.1243.pdf

"Something that becomes clear from the results is that Twitter contains enough information to be useful for predicting outcomes in the Premier League"
https://qz.com/233830/world-cup-germany-argentina-predictions-microsoft/

For the 2014 World Cup, Bing correctly predicted the outcomes for all of the 15 games in the knockout round.

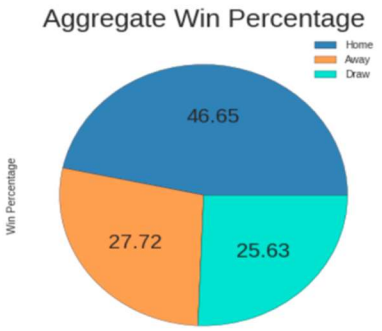
So the right questions to ask are

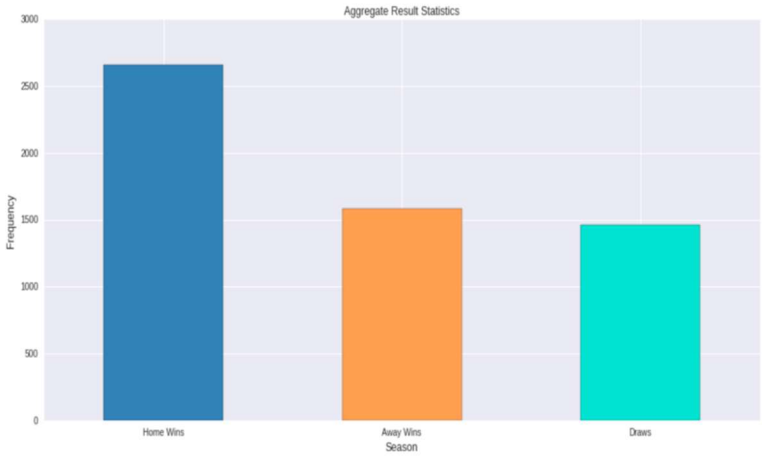
-What model should we use? -What are the features (the aspects of a game) that matter the most to predicting a team win? Does being the home team give a team the advantage?

Dataset

- Football is played by 250 million players in over 200 countries (most popular sport globally)
- The English Premier League is the most popular domestic team in the world
- Retrived dataset from http://football-data.co.uk/data.php

	Home Wins	Away Wins	Draws	Total
Overall	2659	1580	1461	5700



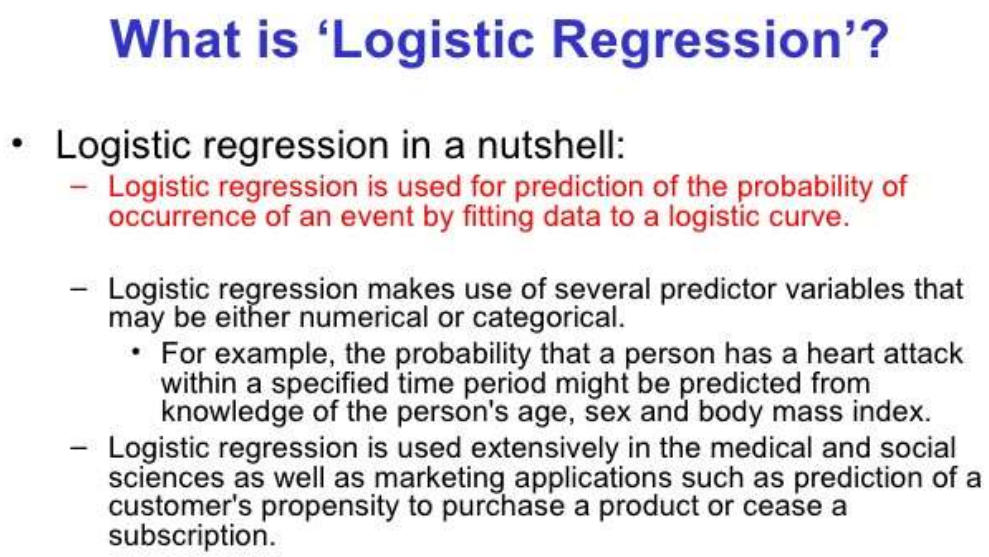


- Football is a team sport, a cheering crowd helps morale
- Familarity with pitch and weather conditions helps
- No need to travel (less fatigue)

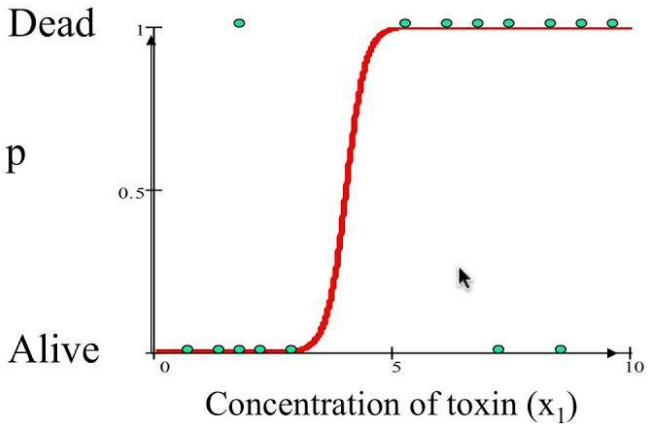
Acrononyms- https://rstudio-pubs-static.s3.amazonaws.com/179121_70eb412bbe6c4a55837f2439e5ae6d4e.html

Other repositories

- https://github.com/rsibi/epl-prediction-2017 (EPL prediction)
- https://github.com/adeshpande3/March-Madness-2017 (NCAA prediction)



$$p = \frac{e^{\{\beta_0 + \beta_1 x\}}}{1 + e^{\{\beta_0 + \beta_1 x_1\}}}$$

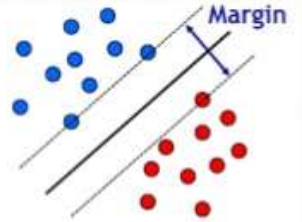


Support Vector Machine (SVM):

- **Basic idea:**

- The SVM tries to find a classifier which maximizes the margin between pos. and neg. data points.
- Up to now: consider linear classifiers

$$\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$$



- Formulation as a convex optimization problem

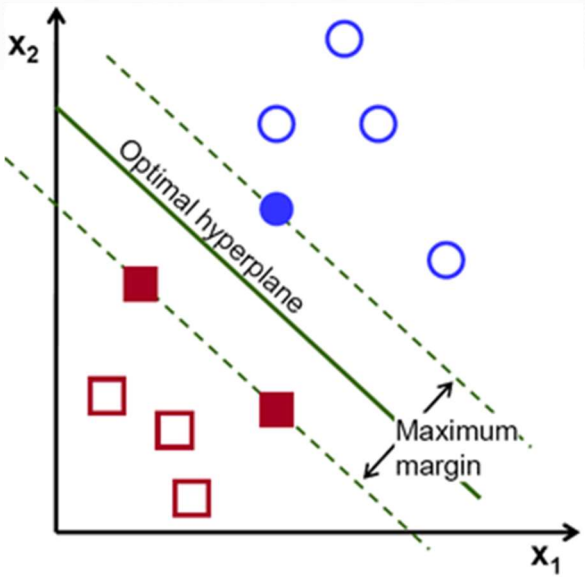
Find the hyperplane satisfying

$$\arg\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

under the constraints

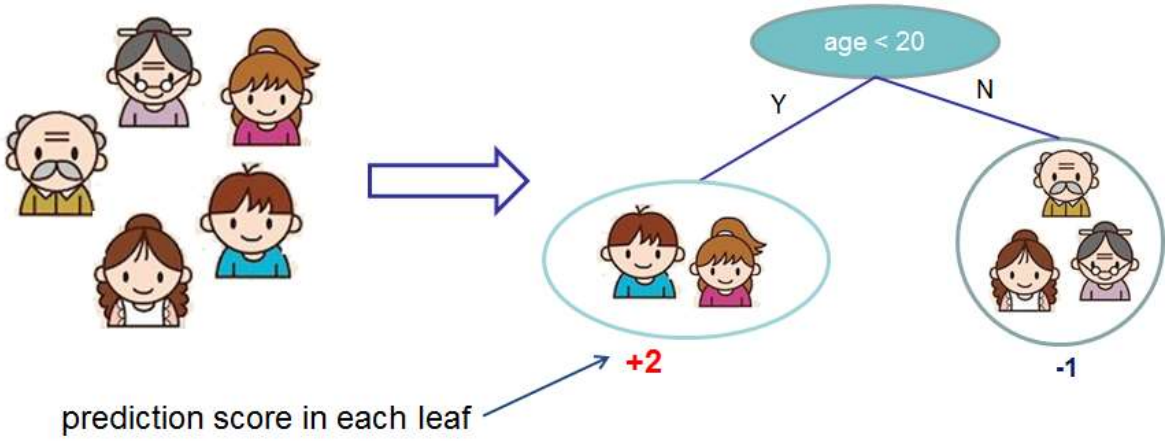
$$t_n(\mathbf{w}^{\mathrm{T}}\mathbf{x}_n + b) \geq 1 \quad \forall n$$

based on training data points $\mathbf{x}_n$ and target values $t_n \in \{-1, 1\}$.

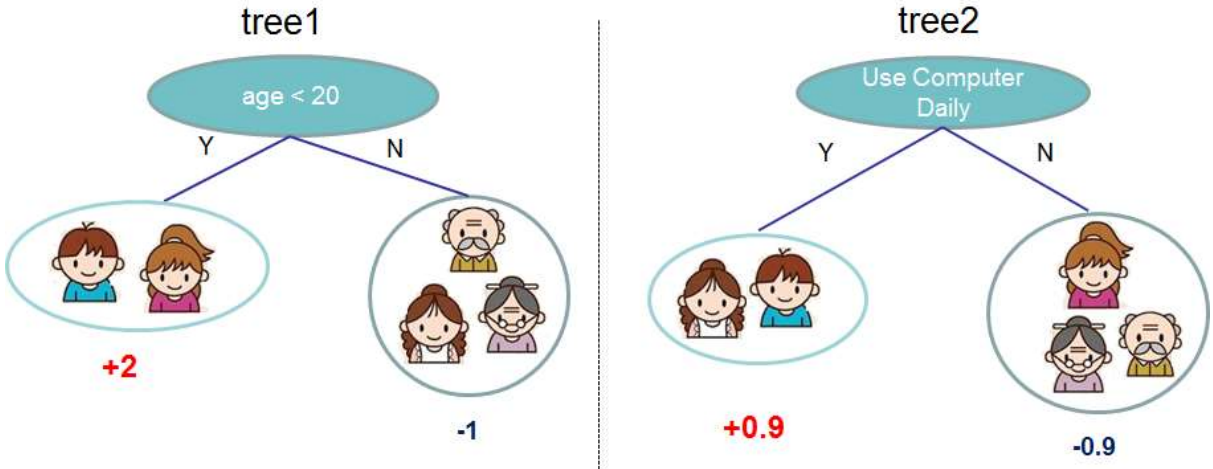


Input: age, gender, occupation, …

Like the computer game X



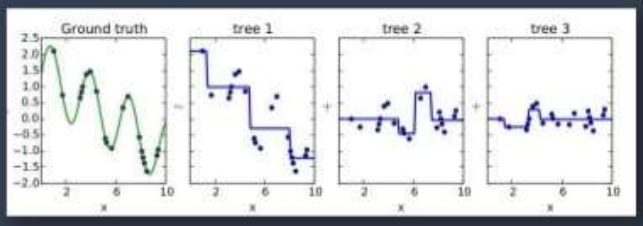
prediction score in each leaf



f() = 2 + 0.9= 2.9 f()= -1 - 0.9= -1.9

XGBoost explained in 2 pics (2/2)

Gradient boosting on CART



- One more tree = loss mean decreases = more data explained
- Each tree captures some parts of the model
- Original data points in tree 1 are replaced by the loss points for tree 2 and 3

XGBoost explained in 2 pics (1/2)

Classification And Regression Tree (CART)

Decision tree is about learning a set of rules:

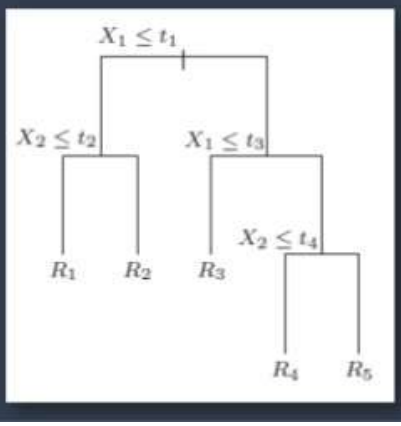
if$(X_1 \leq t_1)$ & if$(X_2 \leq t_2)$ then R_1
if$(X_1 \leq t_1)$ & if$(X_2 > t_2)$ then R_2
...

Advantages:

- Interpretable
- Robust
- Non linear link

Drawbacks:

- Weak Learner ☹
- High variance



Clearly XGBoost seems like the best model as it has the highest F1 score and accuracy score on the test set.

Tuning the parameters of XGBoost.

GBDT Hyper Parameter Tuning

Hyper Parameter	Tuning Approach	Range	Note
# of Trees	Fixed value	100-1000	Depending on datasize
Learning Rate	Fixed => Fine Tune	[2 - 10] / # of Trees	Depending on # trees
Row Sampling	Grid Search	[.5, .75, 1.0]	
Column Sampling	Grid Search	[.4, .6, .8, 1.0]	
Min Leaf Weight	Fixed => Fine Tune	3/(% of rare events)	Rule of thumb
Max Tree Depth	Grid Search	[4, 6, 8, 10]	
Min Split Gain	Fixed	0	Keep it 0

Best GBDT implementation today: https://github.com/tqchen/xgboost
by **Tianqi Chen** (U of Washington)

DataRobot

With Germany's win Microsoft perfectly predicted the World Cup's knockout round

Update 5:40pm ET: With Germany's 1-0 win over Argentina in the final, Microsoft's prediction engine had a perfect record in predicting the knockout stages of the World Cup.

The most accurate predictor of the knockout stages, Microsoft's predictions, have been perfect for every match of the final stage of the World Cup. Generated using a mathematical model, Microsoft predicted an Argentina-Germany final. And it has been predicting a German victory today.

The predictions, delivered in its Bing search engine results and through its Cortana virtual assistant on Windows phones, are part of a broader effort to give people not just relevant search results, but information about the things that interest them that keeps them coming back to Bing, according to Craig Beilinson, Microsoft's director of consumer communications.

"The elimination round has been magical for us here at as we watch all of these predictions come true," Beilinson says. "It's been a fun computer science experience here and we think a lot about where we can take it. We hope it's fun for people to see more than just search results, to see more information that they can follow."

The Microsoft team started out making predictions for things where people determine the outcome, specifically reality TV shows where people vote including The Voice, American Idol, and Dancing With the Stars, using things like search data and analysis of social sentiment from data partnerships with Facebook and Twitter.

“Really for the first time they weren’t going blind watching their favorite shows—they had some inkling of an idea whether the person they were rooting for was in trouble,” Beilinson says. “We had incredible results from an accuracy perspective across these shows and that gave us confidence to try other things.

The team then tried their hand at the NBA draft, with impressive if not perfect results, which led it to the World Cup.

“You’d imagine this is a pretty complex algorithm to get right, right? And anything can happen, hopefully you’ve been watching these matches and seeing how close they are, there’s no perfect algorithm,” Beilinson says. “But the team has spent a lot of time looking into what would make an accurate predictions, the previous record of these players playing together, the kind of turf are they playing on and what are they used to, how far away are they from Brazil, does this really feel like a home or away game. Teaching a PC if you beat up on a weak team, how valuable is that compared to a close match against a really good team? Taking all of the data and having it come up with a set of predictions was our goal.”

More detail on the model, which draws on the work of Microsoft researcher David Rothschild, is available here.

After a rough opening round where the engine got about 60% of the answers right, Microsoft is thrilled that the knockout stages are going so well.

As to why the knockout rounds have been better, part of it is that the favorites have tended to win at a much higher rate. But others have still managed to make high-profile misses, like FiveThirtyEight’s prediction that Brazil would beat Germany and win the tournament.

The team plans to do more predictions in the future, on everything from this November’s US midterm elections to possibly the Emmy awards or movie box office totals.

Microsoft’s latest World Cup run has prompted comparisons to the success of Paul the octopus in predicting Germany’s 2010 World Cup games.

INTRODUCTION

1.1. PURPOSE OF AI IN DESEASE DIGONISIS

AI can use historical data, which in **sports** is well documented, to predict the future potential of players before investing in them. It can also be used to estimate players' market values to make the right offers while acquiring new talent.

1.2. What is AI?

AI is a technique that enables machines to mimic human behavior artificial intelligence is the theory and development of computer systems able to perform tasks normally requiring human intelligence such as visual perception speech recognition decision-making and translation between languages.

Now the term artificial intelligence was actually coined way back in 1956 by John McCarthy or professor at Dartmouth for years it was thought that computers would never match the power of the human brain but it has proven to not be the case well back then we did not have enough data and reputation per hour but now with big data coming into existence with the great advent of cheap use artificial intelligence is much possible.

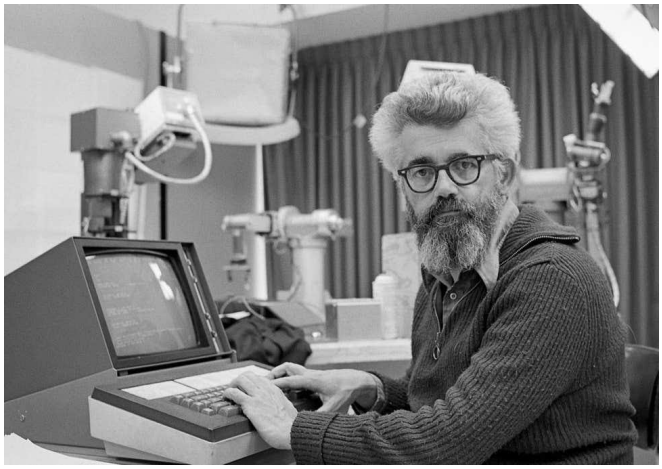


Figure 1: John McCarthy

Now this artificial intelligence machine learning and deep end they all come under the roof of data science well data science is something that has been there for ages and data science is the extraction of knowledge from beta by using different techniques and algorithms.

Now artificial intelligence is the technique which enables machine to mimic human behavior and the idea behind here is fairly simple yet fascinating which is to make intelligent machines that can take decisions on its own now machine learning is a subset of artificial intelligence technique which uses statistical methods to enable machines to improve with experience.
Now deep learning as we know is a subset of machine learning which makes the same computation of multi-layer neural network feasible and uses neural networks to stimulate brain like decision making.

The importance of artificial intelligence it automates repetitive learning and discovery through data paper forms frequent high-volume computerized us reliably and without fatigue.

Ai adapts through progressive learning algorithms to let the data do the programming the algorithm becomes a classifier or a predictor so just as the algorithm can teach itself how to play any game it can teach itself what product to recommend next online it analyzes more and deeper data using neural networks that have many hidden layers you need lots of data and to Train deep learning models because they learn directly from the data the more data you can feed them the more accurate they become now a achieves incredible accuracy its route to planning neural networks.

There are two types of artificial intelligence and the first one is the narrow AI and the second one is the broad Ai.

Narrow a narrow AI is an artificial intelligence system that is designed and trained for one particular task now virtual assistance such as Amazon's Alexi the APUs CV used in Aero AI known area is sometimes also referred to as weak Ai Howard that does not

mean that narrow area is inefficient or something of that sort on the contrary it is extremely good at routine jobs both physical and cognitive it is narrow AI that is threatening to replace many human jobs throughout the world.

wide AI is that why dia is a system with cognitive abilities so that when the system is presented with an unfamiliar task it is intelligent enough to find a solution now here the system is capable of having intelligent behavior across a variety of tasks from driving a car to telling a joke and the techniques aim at replicating and surpassing many capabilities of human intelligence such as risk analysis and other cognitive processes

the robots today are machine learning enabled tools that provide doctors with extended precision and control now these machines enables shortening the patient's hospital stay positive affecting the surgical experience and reducing the medical cost all at once

different domains of artificial intelligence first of all we have neural networks so neural networks are a class of models within the general machine learning literature and they are specific set of algorithms that have revolutionized machine learning and artificial intelligence

1.3. **What is Machine Learning?**

Well, Machine Learning is a concept which allows the machine to learn from examples and experience, and that too without being explicitly programmed. So

instead of you writing the code, what you do is you feed data to the generic algorithm, and the algorithm/ machine builds the logic based on the given data.

- **Types of Machine Learning**

Machine learning is sub-categorized to three types:

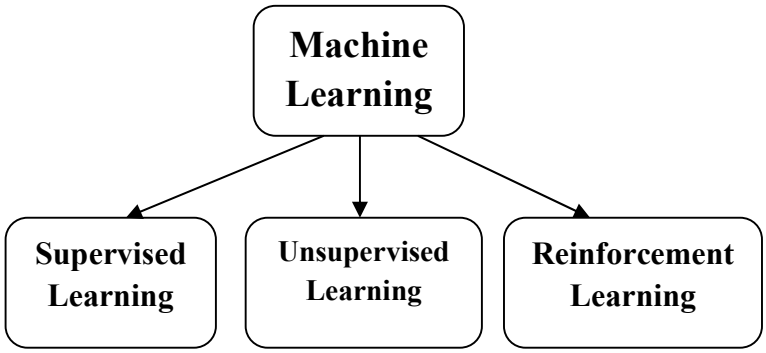


Fig:1.2 Types of Machine Learning

- **What is Supervised Learning?**

Supervised Learning is the one, where you can consider the learning is guided by a teacher. We have a dataset which acts as a teacher and its role is to train the model or the machine. Once the model gets trained it can start making a prediction or decision when new data is given to it.

- **What is Unsupervised Learning?**

The model learns through observation and finds structures in the data. Once the model is given a dataset, it automatically finds patterns and relationships in the dataset by creating clusters in it. What it cannot do is add labels to the cluster, like it cannot say this a group of apples or mangoes, but it will separate all the apples from mangoes.

- **What is Reinforcement Learning?**

It is the ability of an agent to interact with the environment and find out what is the best outcome. It follows the concept of hit and trial method. The agent is rewarded or penalized with a point for a correct or a wrong answer, and on the basis of the positive reward points gained the model trains itself. And again once trained it gets ready to predict the new data presented to it.

1.4. **What is Deep Learning?**

Deep learning is one of the only methods by which we can overcome the challenges of feature extraction. This is because deep learning models are capable of learning to focus on the right features by themselves, requiring little guidance from the programmer. Basically, deep learning mimics the way our brain functions i.e. it learns from experience. As you know, our brain is made up of billions of neurons that allows us to do amazing things. Even the brain of a one year old kid can solve complex problems which are very difficult to solve even using super-computers. For example:

Recognize the face of their parents and different objects as well.

Discriminate different voices and can even recognize a particular person based on his/her voice.

Draw inference from facial gestures of other persons and many more.

Actually, our brain has sub-consciously trained itself to do such things over the years. Now, the question comes, how deep learning mimics the functionality of a brain? Well, deep learning uses the concept of artificial neurons that functions in a similar manner as the biological neurons present in our brain. Therefore, we can say that Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.

- **How Deep Learning Works?**

In an attempt to re-engineer a human brain, Deep Learning studies the basic unit of a brain called a brain cell or a neuron. Inspired from a neuron an artificial neuron or a perceptron was developed. Now, let us understand the functionality of biological neurons and how we mimic this functionality in the perceptron or an artificial neuron:

If we focus on the structure of a biological neuron, it has dendrites which is used to receive inputs. These inputs are summed in the cell body and using the Axon it is passed on to the next biological neuron as shown in the above image.

Similarly, a perceptron receives multiple inputs, applies various transformations and functions and provides an output.

As we know that our brain consists of multiple connected neurons called neural network, we can also have a network of artificial neurons called perceptrons to form a Deep neural network. So, let's move ahead in this Deep Learning Tutorial to understand how a Deep neural network looks like.

Any Deep neural network will consist of three types of layers:

- The Input Layer

the first layer is the input layer which receives all the inputs and the last layer is the output layer which provides the desired output.

- The Hidden Layer

All the layers in between these layers are called hidden layers. There can be n number of hidden layers thanks to the high end resources available these days.

- The Output Layer

The number of hidden layers and the number of perceptrons in each layer will entirely depend on the use-case you are trying to solve.

1.5 Relation between Ai, Machine learning and Deep learning

Artificial Intelligence makes it possible for the machines to learn from their experience. The machines adjust their response based on new inputs thereby performing human-like tasks by processing large amounts of data and recognizing patterns in them.

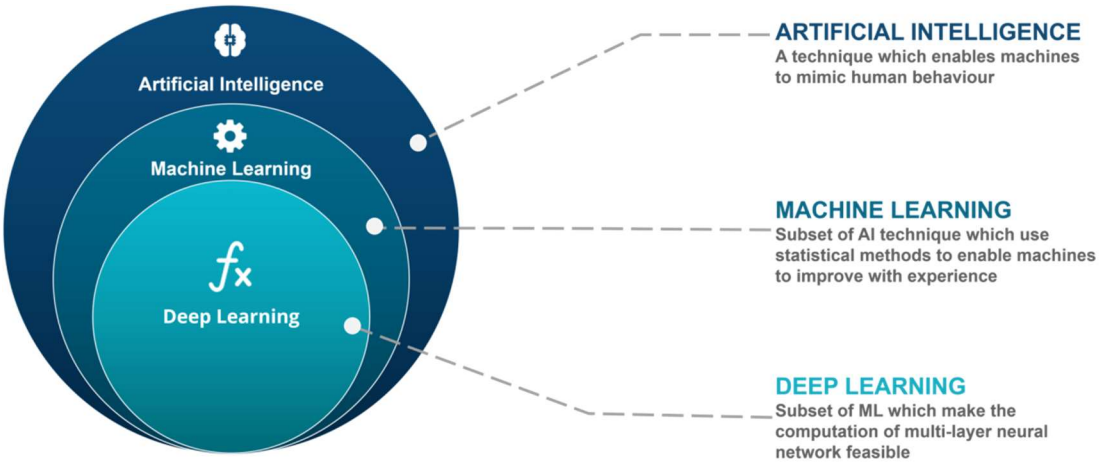


Figure -1.3 the relation between AI, Machine learning and Deep learning

The first church took generations to finish, so most of the workers working on it never saw the final outcome. Those working on it took pride in their craft, building bricks and chiselling stones that were to be placed into the Great Structure. So, as AI researchers, we should think ourselves as humble brick makers, whose job it is to study how to build components (e.g. parsers, planners, learning algorithms, etc) that someday someone, somewhere, will integrate into intelligent systems.

Some of the examples of Artificial Intelligence from our day to day life are Apple's Siri, the chess-playing computer, tesla's self-driving car and many more. These examples are based on deep learning and natural language processing.

Machine Learning is a subset of artificial intelligence. It allows the machines to learn and make predictions based on its experience (data).

Deep learning is a particular kind of machine learning that achieves great power and flexibility by learning to represent the world as nested hierarchy of concepts or abstraction.

The concept of deep learning is not new. But recently its hype has increased, and deep learning is getting more attention. This field is a special kind of machine learning which is inspired by the functionality of our brain cells called artificial neural network. It simply takes data connections between all artificial neurons and adjusts them according to the data pattern. More neurons are needed if the size of the data is large. It automatically features learning at multiple levels of abstraction thereby allowing a system to learn complex functions mapping without depending on any specific algorithm.

1.6 Advantage of Deep learning over Machine learning

- ### Reduction in Human Error

The phrase "**human error**" was born because humans make mistakes from time to time. Computers however, do not make these mistakes if they are programmed properly. With Artificial intelligence, the decisions are taken from the previously gathered information applying certain set of algorithms. So errors are reduced and the chance of reaching accuracy with a greater degree of precision is a possibility.

Example: In Weather Forecasting using AI they have reduced majority of human error.

- ## Takes risks instead of Humans

This is one of the biggest advantage of Artificial intelligence. We can overcome many risky limitations of human by developing an AI Robot which in turn can do the risky things for us. Let it be going to mars, defuse a bomb, explore the deepest parts of oceans, minning for coal and oil, it can be used effectively in any kinds of nautural or man made disasters.

Example: Have you heard about the **Chernoby**l nuclear power plant explosion in Ukraine? At that time there were no AI powered robots which can help us to minimise the affect of radiation by controlling the fire in early stages, as any human went close to the core was dead in matter of minutes. They eventually poured sand and boron from helichopters from a mere distance.

AI Robots can be used in such situations where an human intervention can be hazardous.

- ## Available 24×7

An Average human will work for 4-6 hours a day excluding the breaks. Humans are built in such a way to get some time out for refereshing themselves and get ready for a new day of work and they even have weekly off's to stay intact with their work life and personal life. But using AI we can make machines work 24×7 without any breaks and they don't even get bored unlike humans.

Example: Educational Institutes and Helpline centers are getting many queries and issues which can be handled effecively using AI.

- ## Helping in Repetitive Jobs

In our day-to-day work, we will be performing many repetitive works like sending a thanking mail, verifying certain documents for errors and many more things. Using artificial intelligence we can productively automate these mundane tasks and can even remove "**boring**" tasks for humans and free them up to be increasingly creative.

Example: In banks, we often see many verifications of documents in order to get a loan which is a repetitive task for the owner of the bank. Using AI Cognitive

Automation the owner can speed up the process of verifying the documents by which both the customers and the owner will be benefited.

- ## Digital Assistance

Some of the highly advanced organizations use digital assistants to interact with users which saves the need of human resource. The digital assistant also used in many websites to provide things that user want. We can chat with them about what we are looking for. Some chatbots are designed in such a way that it becomes hard to determine that we're chatting with a chatbot or a human being.

Example: We all know that organizations have a customer support team which needs to clarify the doubts and queries of the customers. Using AI the organizations can set up a Voicebot or Chatbot which can help customers with all their queries. We can see many organizations already started using them in their websites and mobile applications.

- ## Faster Decisions:

Using AI alongside other technologies we can make machines take decisions faster than a human and carry out actions quicker. While taking a decision human will analyze many factors both emotionally and practically but AI-powered machine works on what it is programmed and delivers the results in a faster way.

Example: We all have played a Chess game in Windows. It is nearly impossible to beat CPU in the hard mode because of the AI behind that game. It will take the best possible step in a very short time according to the algorithms used behind it.

- ## Daily Applications

Daily applications such as Apple's **Siri**, Window's **Cortana**, Google's **OK Google** are frequently used in our daily routine whether it is for searching a location, taking a selfie, making a phone call, replying to a mail and many more.

Example: Around 20 years ago, when we are planning to go somewhere we used to ask a person who already went there for the directions. But now all we have to do is say "**OK Google** where is Visakhapatnam". It will show you Visakhapatnam's location on google map and best path between you and Visakhapatnam.

- **New Inventions:**

AI is powering many inventions in almost every domain which will help humans solve the majority of complex problems.

1.7 Limitation of Deep learning

In deep learning, everything is a vector, i.e. everything is a point in a geometric space. Model inputs (it could be text, images, etc) and targets are first "vectorized", i.e. turned into some initial input vector space and target vector space. Each layer in a deep learning model operates one simple geometric transformation on the data that goes through it. Together, the chain of layers of the model forms one very complex geometric transformation, broken down into a series of simple ones. This complex transformation attempts to maps the input space to the target space, one point at a time. This transformation is parametrized by the weights of the layers, which are iteratively updated based on how well the model is currently performing. A key characteristic of this geometric transformation is that it must be differentiable, which is required in order for us to be able to learn its parameters via gradient descent. Intuitively, this means that the geometric morphing from inputs to outputs must be smooth and continuous—a significant constraint.

The whole process of applying this complex geometric transformation to the input data can be visualized in 3D by imagining a person trying to uncrumple a paper ball: the crumpled paper ball is the manifold of the input data that the model starts with. Each movement operated by the person on the paper ball is similar to a simple geometric transformation operated by one layer. The full uncrumpling gesture sequence is the complex transformation of the entire model. Deep learning models are mathematical machines for uncrumpling complicated manifolds of high-dimensional data.

That's the magic of deep learning: turning meaning into vectors, into geometric spaces, then incrementally learning complex geometric transformations that map one space to another. All you need are spaces of sufficiently high dimensionality in order to capture the full scope of the relationships found in the original data.

- ## The limitations of deep learning

The space of applications that can be implemented with this simple strategy is nearly infinite. And yet, many more applications are completely out of reach for current deep learning techniques—even given vast amounts of human-annotated data. Say, for instance, that you could assemble a dataset of hundreds of thousands—even millions—of English language descriptions of the features of a software product, as written by a product manager, as well as the corresponding source code developed by a team of engineers to meet these requirements. Even with this data, you could not train a deep learning model to simply read a product description and generate the appropriate codebase. That's just one example among many. In general, anything that requires reasoning—like programming, or applying the scientific method—long-term planning, and algorithmic-like data manipulation, is out of reach for deep learning models, no matter how much data you throw at them. Even learning a sorting algorithm with a deep neural network is tremendously difficult.

This is because a deep learning model is "just" a chain of simple, continuous geometric transformations mapping one vector space into another. All it can do is map one data manifold X into another manifold Y, assuming the existence of a learnable continuous transform from X to Y, and the availability of a dense sampling of X:Y to use as training data. So even though a deep learning model can be interpreted as a kind of program, inversely most programs cannot be expressed as deep learning models—for most tasks, either there exists no corresponding practically-sized deep neural network that solves the task, or even if there exists one, it may not be learnable, i.e. the corresponding geometric transform may be far too complex, or there may not be appropriate data available to learn it.

Scaling up current deep learning techniques by stacking more layers and using more training data can only superficially palliate some of these issues. It will not solve the more fundamental problem that deep learning models are very limited in what they

can represent, and that most of the programs that one may wish to learn cannot be expressed as a continuous geometric morphing of a data manifold.

- **The risk of anthropomorphizing machine learning models**

One very real risk with contemporary AI is that of misinterpreting what deep learning models do, and overestimating their abilities. A fundamental feature of the human mind is our "theory of mind", our tendency to project intentions, beliefs and knowledge on the things around us. Drawing a smiley face on a rock suddenly makes it "happy"—in our minds. Applied to deep learning, this means that when we are able to somewhat successfully train a model to generate captions to describe pictures, for instance, we are led to believe that the model "understands" the contents of the pictures, as well as the captions it generates. We then proceed to be very surprised when any slight departure from the sort of images present in the training data causes the model to start generating completely absurd captions.

1.8 Python as a programming language

- Python In Artificial Intelligence

One of the key features of python language is its simplicity in code writing. It uses $1/5^{th}$ of the code when compared to other object oriented programs. This factor makes it the most sort after language used in trending domains like AI. AI has a wide horizon under which it deals with machine learning and deep learning.

Python has a variety of libraries that appeal to the needs of any programmer. It has some prebuilt libraries such as Numpy, SciPy, Pybrain etc., which are for advance and scientific computing. Python is platform independent, which makes it quite flexible in interfacing between other technologies. In addition, the current user base of the language is very diverse. Most python developers share queries and solutions on portals, which make it a comprehensive knowledge resource as well.

The language not only applies OOPs concepts but incorporates a scripting approach as well. There are numerous IDEs (Integrated Development Environment) like PyCharm, which allows users to carry out complex codes and algorithms of AI related projects. In an AI's SDLC (Software Development Life Cycle) phase like

testing, debugging and development, it becomes a cakewalk, when compared against other contemporary programming languages like Java, Javascript and Pearl.

These languages would definitely yield desired results but would make tasks cumbersome. Hence, looking at the numerous advantages of python, there is no doubt that it plays a crucial aspect in today's AI technologies.

- Deep Learning In Python

Deep learning is another trending domain in todays' world of Artificial Intelligence. Deep learning techniques are so powerful because they represent and learn of how to solve a problem in the best possible way. This is called as "Representation Learning". The deep learning programs are trained with numerous examples that make its predictions accurate. Deep learning models are extensively used in colorizing images and videos. It is used in identification of objects in photographs popularly termed as 'face-recognition'.

Python is the best platform to get started with deep learning models. Python is quick and easy to understand. It has a ton of features that make deep learning projects faster to operate and develop. The two most versatile libraries used by any deep learning expert is "Theano" and "Tensorflow". These are quiet technical and used exhaustively by research groups. The "Keras" library is written in pure python which provides an interface for the above two libraries.

- **Python As A Programming Language**

Now that we know how important Python is to the world and us. Let us deep dive into learning some of the technical aspects of the programming language. The below illustrated topics are rudimentary and would be easy to grasp.

1.9 Python use in machine learning

Machine Learning, as the name suggests, is the science of programming a computer by which they are able to learn from different kinds of data. A more general definition

given by Arthur Samuel is – “Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.” They are typically used to solve various types of life problems.

In the older days, people used to perform Machine Learning tasks by manually coding all the algorithms and mathematical and statistical formula. This made the process time consuming, tedious and inefficient. But in the modern days, it is become very much easy and efficient compared to the olden days by various python libraries, frameworks, and modules. Today, Python is one of the most popular programming languages for this task and it has replaced many languages in the industry, one of the reason is its vast collection of libraries. Python libraries that used in Machine Learning are:

- Numpy
- Scipy
- Scikit-learn
- Pandas
- Matplotlib

- NumPy is a very popular python library for large multi-dimensional array and matrix processing, with the help of a large collection of high-level mathematical functions. It is very useful for fundamental scientific computations in Machine Learning. It is particularly useful for linear algebra, Fourier transform, and random number capabilities. High-end libraries like TensorFlow uses NumPy internally for manipulation of Tensors.

```
# Python program using NumPy
# for some basic mathematical
# operations

import numpy as np

# Creating two arrays of rank 2
x = np.array([[1, 2], [3, 4]])
```

```
y = np.array([[5, 6], [7, 8]])

# Creating two arrays of rank 1
v = np.array([9, 10])
w = np.array([11, 12])

# Inner product of vectors
print(np.dot(v, w), "\n")

# Matrix and Vector product
print(np.dot(x, v), "\n")

# Matrix and matrix product
print(np.dot(x, y))
```

Output:

```
219


[29 67]


[[19 22]
 [43 50]]
```

Figure 1.4 :The output of above program

- SciPy is a very popular library among Machine Learning enthusiasts as it contains different modules for optimization, linear algebra, integration and statistics. There is a difference between the SciPy library and the SciPy stack. The SciPy is one of the core packages that make up the SciPy stack. SciPy is also very useful for image manipulation.

```
# Python script using Scipy
# for image manipulation

from scipy.misc import imread, imsave, imresize
```

```
# Read a JPEG image into a numpy array
img = imread('D:/Programs / cat.jpg') # path of the image
print(img.dtype, img.shape)

# Tinting the image
img_tint = img * [1, 0.45, 0.3]

# Saving the tinted image
imsave('D:/Programs / cat_tinted.jpg', img_tint)

# Resizing the tinted image to be 300 x 300 pixels
img_tint_resize = imresize(img_tint, (300, 300))

# Saving the resized tinted image
imsave('D:/Programs / cat_tinted_resized.jpg', img_tint_resize)
```

output:

Original image:

Tinted image:

Resized tinted image:

Figure 1.5 output of above program

- Skikit-learn is one of the most popular ML libraries for classical ML algorithms. It is built on top of two basic Python libraries, viz., NumPy and SciPy. Scikit-learn supports most of the supervised and unsupervised learning algorithms. Scikit-learn can also be used for data-mining and data-analysis, which makes it a great tool who is starting out with ML.

```
# Python script using Scikit-learn
# for Decision Tree Clasifier

# Sample Decision Tree Classifier
from sklearn import datasets
from sklearn import metrics
from sklearn.tree import DecisionTreeClassifier

# load the iris datasets
dataset = datasets.load_iris()

# fit a CART model to the data
model = DecisionTreeClassifier()
model.fit(dataset.data, dataset.target)
print(model)

# make predictions
expected = dataset.target
predicted = model.predict(dataset.data)

# summarize the fit of the model
print(metrics.classification_report(expected, predicted))
print(metrics.confusion_matrix(expected, predicted))
```

Output:

```
DecisionTreeClassifier(class_weight=None, criterion='gini',
max_depth=None,
            max_features=None, max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, presort=False,
random_state=None,
            splitter='best')
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       1.00      1.00      1.00        50
           2       1.00      1.00      1.00        50

   micro avg       1.00      1.00      1.00       150
   macro avg       1.00      1.00      1.00       150
weighted avg       1.00      1.00      1.00       150


[[50  0  0]
 [ 0 50  0]
 [ 0  0 50]]
```

Figure 1.6 :output of the above program

- Pandas is a popular Python library for data analysis. It is not directly related to Machine Learning. As we know that the dataset must be prepared before training. In this case, Pandas comes handy as it was developed specifically for data extraction and preparation. It provides high-level data structures and wide variety tools for data analysis. It provides many inbuilt methods for groping, combining and filtering data.

```
# Python program using Pandas for
# arranging a given set of data
# into a  table

# importing pandas as pd
import pandas as pd

data = {"country": ["Brazil", "Russia", "India", "China", "South
Africa"],
       "capital": ["Brasilia", "Moscow", "New Dehli", "Beijing",
"Pretoria"],
       "area": [8.516, 17.10, 3.286, 9.597, 1.221],
       "population": [200.4, 143.5, 1252, 1357, 52.98] }

data_table = pd.DataFrame(data)
print(data_table)
```

Output:

	country	capital	area	population
0	Brazil	Brasilia	8.516	200.40
1	Russia	Moscow	17.100	143.50
2	India	New Dehli	3.286	1252.00
3	China	Beijing	9.597	1357.00
4	South Africa	Pretoria	1.221	52.98

Figure 1.7: output of above program

- Matpoltlib is a very popular Python library for data visualization. Like Pandas, it is not directly related to Machine Learning. It particularly comes in handy when a programmer wants to visualize the patterns in the data. It is a 2D plotting library used for creating 2D graphs and plots. A module named pyplot makes it easy for programmers for plotting as it provides features to control line styles, font properties, formatting axes, etc. It provides various kinds of graphs and plots for data visualization, viz., histogram, error charts, bar chats, etc.

```
# Python program using Matplotib
# for forming a linear plot

# importing the necessary packages and modules
import matplotlib.pyplot as plt
import numpy as np

# Prepare the data
x = np.linspace(0, 10, 100)

# Plot the data
plt.plot(x, x, label ='linear')

# Add a legend
plt.legend()

# Show the plot
plt.show()
```

Output:

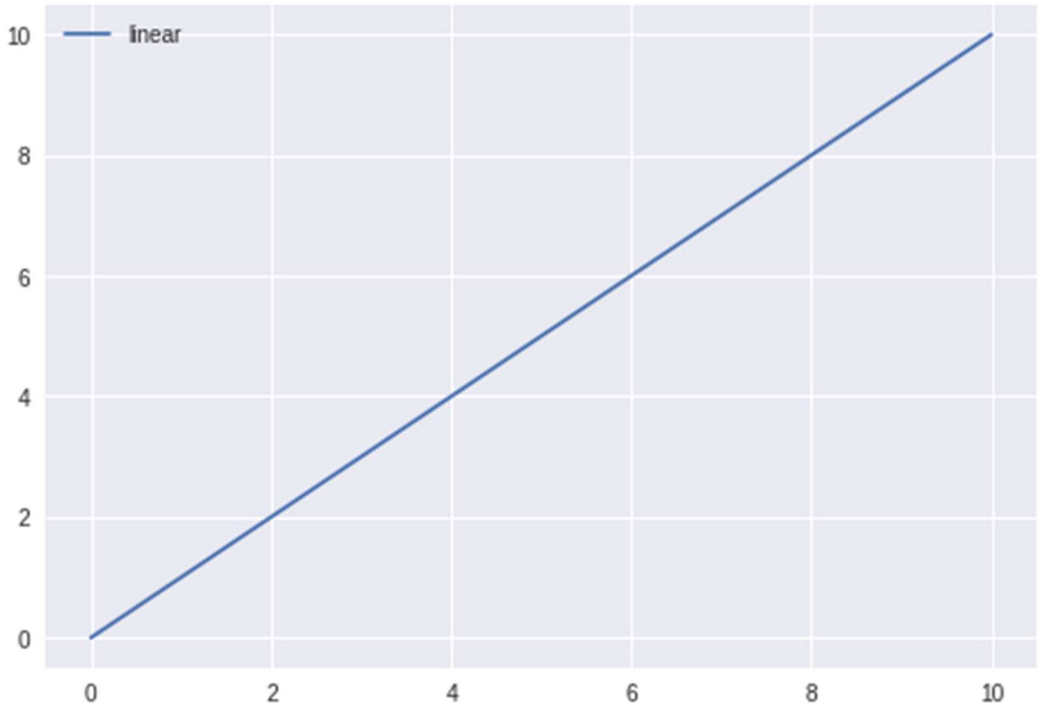


Figure 1.8: output of above prgram

1.10. Packages use in project:

1.10.1 NUMPY :

NumPy – which stands for numerical Python; It is a library consisting of multidimensional, array objects and collection of routines on processing those always. NumPy helps to perform mathematical and logical operation on array.

Operation using NumPy:

. Mathematical and logical on array.

. Furrier transforms and routines for shape manipulation.

. Operation related to linear algebra and in build functions for linear Algebra and random number generation.

NumPy is often used along with packages like spicy (scientific Python) and MatPlotlib (Plotting library)

Import NumPy as me the most import object define in NumPy is an n dimensional array type called nd array.

```
import NumPy a np
 a = np.array ([1, 2, 3])
 Print a
```

Output: -

[1, 2, 3]

```
import NumPy as np
a= np.array ([[12, 3],[4,5,6]])
 Print a.Shape
```

Output:

(2, 3)

1.10.2 : PANDAS

Pandas is an open source library that allows to you performs data manipulation. Pandas library is built on top of NumPy. Pandas need NumPy to operands, it's provided and easy way to create manipulates the data. It uses series on 1D data structure and DataFrame for multidimensional data structure. It provided slides the data, merge, concatenate, reshape the data.

- **What is DataFrame?**

A DataFrame is a 2D array with labels 2D access (row and Colum). It is standard way to data.

	Item	Price
0	A	2
1	B	3

- ## Create a data frame

  We can convert a numpy array to a pandas DataFrame with pd.DataFrame( ) . Here opposite is also possible, using np.array( ) that we can perfrom in a operation.

- ## Numpy to pandas

  Import numpy as np

  h = [[1, 2], [3,4]]

  table_h = pd.DataFrame(h)

  print('DataFrame : ' table_h)

- **Pandas to Numpy**

  array_h = np.array(table_h)

  print('Numpy_array : ' array_h)

- **output**

	0	1
0	1	2
1	3	4

DataFrame

Numpy array : [[1, 2] , [3, 4]]

- We have use directory to create a pandas DataFrame :

  Dic = { 'Name' : ["JOHN", "SMITH"], 'AGE' : [30, 40]}

  Pd.DataFrame (Data = Dic)

OUTPUT:

	AGE	NAME
0	30	JOHN
1	40	SMITH

- **Rangedata :**

- **Create data :**

Date_D = PD.Date_range (‘2030 01 01’, priodes = 6, freq = ‘D’
Print (‘Day : ‘. Dates_D)

OUTPUT:
Day : DatetimeIndex ([‘2030-01-01’, ‘2030-01-02’, ‘2030-01-03’, ‘2030-01-04’, ‘2030-01-05’, ‘2030-01-06’], Dtye = ‘datatime64 [ns]’, freq = ‘D’)

- **Inspecting data :-**

We can check the head or tail of the dataset

Step 1 : Craete rando, sequence with numpy. The sequence has 4 coloums and 6 rows.

Randon = np.random . rand n (6,4)

Setp 2 : Then create a data frame using pandas ; Use data_m as an index for the DataFrame. It means each row will be given an name or index corresponding to a date.

CRATE WITH DATE:

df = pd.DataFrame (random, index = date_m, coloms = list (‘ABCD’))

Step 3 : df .head (3)

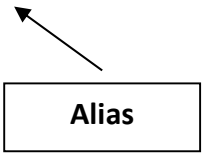
OUTPUT :

	A	B	C
2030-01-31	1.3	1.31	1.312
2030-02-28	2	4	6
2030-03-31	7	8	9

1.10.3 : Matplotlib :-

It is a ploting library for the python programming language and it's nemarical mathe,atics extension. Numpy using these package we can generates plots, histograms, power spectra, bar charts, error charts, scatter diagram etc.

Import matplotlib . pyplot as plt



- **The code will be: –**

```
X_values = [0, 1, 2, 3, 4, 5]

Squares = [0, 1, 4, 9, 16, 25]

Plt.plot (x_values, squares)

Plt.show ( )
```

- **Bar graph :-**

```
X_values = [50, 60, 10, 70, 30]
Y_values = [ "a", "b", "c", "d", "e" ]
Plt.bar (Y_values, X_values, color = "Red" )
Plt.show ( )
```

- **Different between Seaborn and matplotlib** :-

  Seaborn is complimentary to matplotlib and it specially tergates statistical data visualization. Seaborn extends matplotlib and thats why it cab address, limitation of matplotlib , seaborn work with different paremeter. The matplotlib defaults that usually don't speak to users are the colors, tick marks, styles.To load dataset in seaborn we need to use -

  load_dataset ( ) method.

- **HEATMAP IN PYTHON:**

  A heatmap is a 2D graphical representation of data where the individual values that are containing in are representated as columns. It represents different colour coding to represents different values the heatmaps can be use to show where users have clicked on a page, how far they have scroll down a page.

  A correlation heatmap uses colour cells, typically in a monochromatic scall, to show a 2D correlation matrix, the colour value of the cells is proportional to the number of measurement that match the dimension values.

- **Python Heatmap Code**

  We will create a seaborn heatmap for a group of 30 Pharmaceutical Company stocks listed on the National Stock Exchange of India Ltd (NSE). The seaborn heatmap will display the stock symbols and its respective single-day percentage price change.

  We collate the required market data on Pharma stocks and construct a comma-separated value (CSV) file comprising of the stock symbols and their respective percentage price change in the first two columns of the CSV file.

  Since we have 30 Pharma companies in our list, we will create a heatmap matrix of 6 rows and 5 columns. Further, we want our seaborn heatmap to display the percentage price change for the stocks in a descending order. To that effect, we arrange the stocks in a descending order in the CSV file and add two more columns which indicate the position of each stock on X & Y axis of our heatmap.

- Import the required Python packages
  We import the following Python packages:

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

- **Load the dataset**

We read the dataset using the read_csv function from pandas and visualize the first ten rows using the print statement.

Create a Python Numpy array

Since we want to construct a 6 x 5 matrix, we create an n-dimensional array of the same shape for “Symbol” and the “Change” columns.

Create a Pivot in Python

The pivot function is used to create a new derived table from the given data frame object “df”. The function takes three arguments; index, columns, and values. The cell values of the new table are taken from column given as the values parameter, which in our case is the “Change” column.

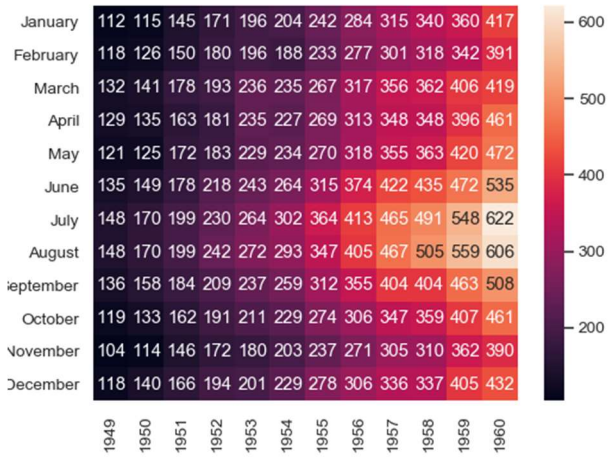
Create an Array to Annotate the Heatmap

In this step, we create an array which will be used to annotate the seaborn heatmap. We call the flatten method on the “symbol” and “percentage” arrays to flatten a Python list of lists in one line. The zip function which returns an iterator zips a list in Python. We run a Python For loop and by using the format function; we format the stock symbol and the percentage price change value as per our requirement.

Create the Matplotlib figure and define the plot

We create an empty Matplotlib plot and define the figure size. We also add the title to the plot and set the title’s font size, and its distance from the plot using the set_position method.

We wish to display only the stock symbols and their respective single-day percentage price change. Hence, we hide the ticks for the X & Y axis, and also remove both the axes from the heatmap plot.



1.10:4 Sk_learn :-

scikit – learn is an open source python library that implements a range of machine learning algorithms

- Simple tool for datamining and analysis :- It's features various classification, regration, clustering algorithms, including support vector machine, Random forests, fradient boosting, K-means algorithms.
- It works with NUMPY, SYPY the easiest way to install scikit learn.

```
pip install _  - U_scikit_learn
```

- Load the dataset, a dataset is nothing but a collection of data, a dataset has 2 main components -
  - Features
  - Response
- **Features** : They can simply variables of our data, they can be more than one and represented by a feature matrix
- **Response** : This is the output variable depending on the feature variable

// Load the iris dataset as an example.

```
from _sklearn . datasets_import  load_iris
```

```
Iris  =  load_iris (  )
```

// store the feature matrix and respose vector

```
X  =  iris . daya # we are  saving the feature to x
```

```
Y  =  iris . target # response
```

```
# Store the feature and target names,
feature_names  =  iris .feature_names
target _names  =  iris. target _names
print( type ( x ) )
print ( X [ : 5 ] )
```

1.11 PROBLEM DOMAIN

Assuming to the problem domain, we can see from the above discussion of the project (Scocer winning prediction using deep learning) that is the accuracy level of our machine learning model is to be 80% to 90% depending upon the strength of training and testing dataset.As much more accurate data we gather the result is to be much more accurate depending on our machine learning model.

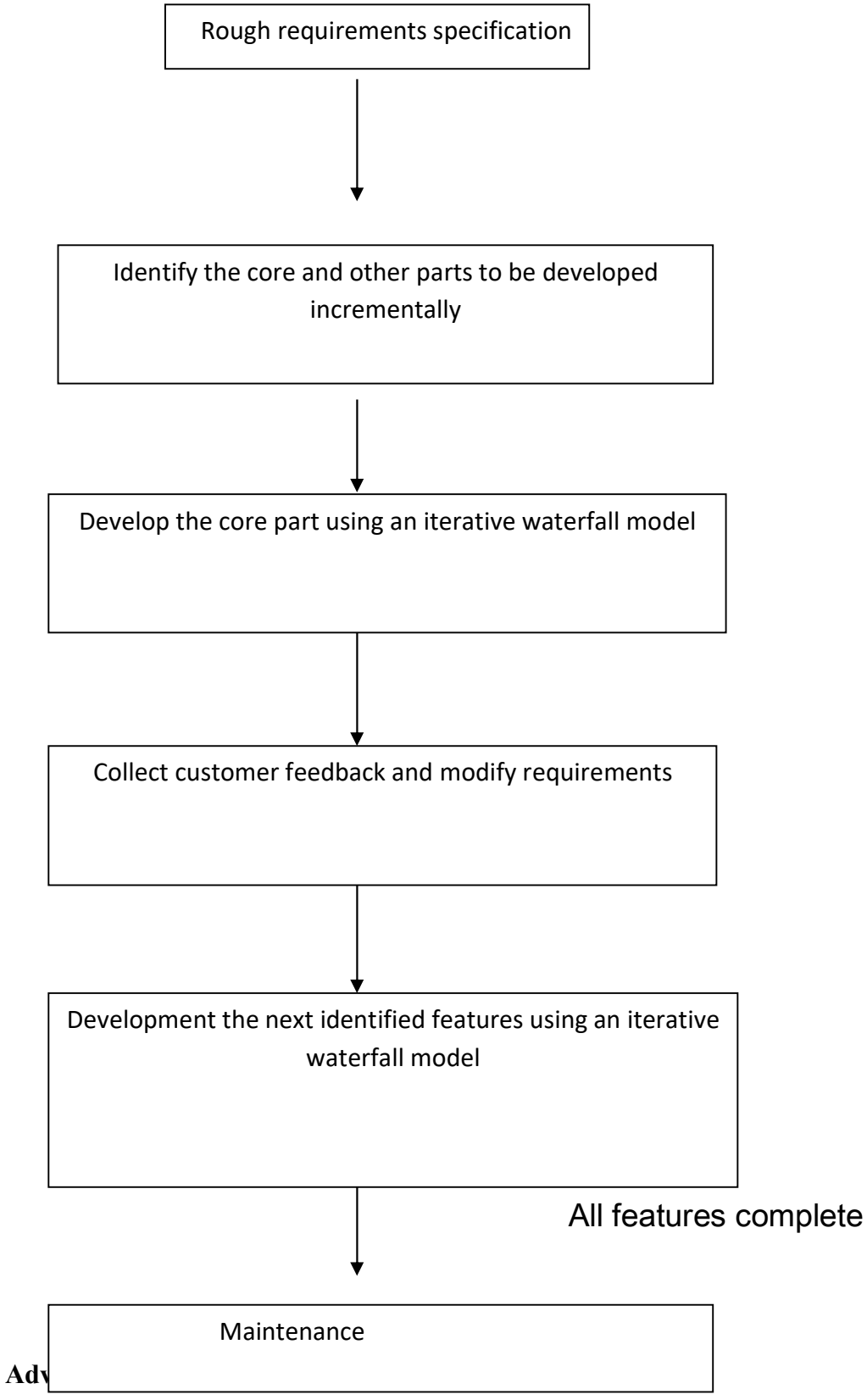
If we want to predict the accurate level of a winning scocer match then first of all we have to collect the match and players information that works as a feature of our dataset after train and test those data we get a accurate result that is 80% to 90% accurate and predict the winning team of the match.

1.12 SOLUTION DOMAIN

Evolutionary Model

This model has many of the features of the incremental model. As in case of the increment model, the software is developed over a number of increments. At each increment, a concept (feature) is implemented and is deployed at the client site. The software is successively refined and feature-enriched until the full software is realised. The principal idea behind the evolutionary life cycle model is conveyed by its name. In the incremental development model, complete requirements are first developed and the SRS document prepared. In contrast, in the evolutionary model, the requirements, plan, estimates, and solution evolve over the iterations, rather than fully defined and frozen in a major up-front specification effort before the development iterations begin. Such evolution is consistent with the pattern of unpredictable feature discovery and feature changes that take place in new product development.

Though the evolutionary model can also be viewed as an extension of the waterfall model, but it incorporates a major paradigm shift that has been widely adopted in many recent life cycle models. Due to obvious reasons, the evolutionary software development process is sometimes referred to as design a little, build a little, test a little deploy a little model. This means that after the requirements have been specified, the design, build, test, and deployment activities are iterated. A schematic representation of the evolutionary mode of development has been shown in Figure .


Rough requirements specification
Identify the core and other parts to be developed incrementally
Develop the core part using an iterative waterfall model
Collect customer feedback and modify requirements
Development the next identified features using an iterative waterfall model
All features complete
Maintenance

Adv

The evolutionary model of development has several advantages. Two important advantages using this model are the following:

- **Effective elicitation of actual customer requirements:**

In this model, the user gets a chance to experiment with a partially developed software much before the complete requirements are developed. Therefore, the evolutionary model helps to accurately elicit user requirements with the help of feedback obtained on the delivery of different versions of the software. As a result, the change requests after delivery of the complete software gets substantially reduced.

- **Easy handling change requests:**

In this model, handling change requests is easier as no long-term plans are made. Consequently, reworks required due to change requests are normally much smaller compared to the sequential models.

Disadvantages

The main disadvantages of the successive versions model are as follows:

- **Feature division into incremental parts can be non-trivial:**

For many development projects, especially for small-sized projects, it is difficult to divide the required features into several parts that can be incrementally implemented and delivered. Further, even for larger problems, often the features are so intertwined and dependent on each other that even an expert would need considerable effort to plan the incremental deliveries.

- **Ad hoc design:**

Since at a time design for only the current increment is done, the design can become ad hoc without specific attention being paid to maintainability and optimality. Obviously, for moderate sized problems and for those for which the customer requirements are clear, the iterative waterfall model can yield a better solution.

Evolutionary versus incremental model of developments

The evolutionary and incremental have several things in common, such as incremental development and deployment at the client site. However, in a purely incremental model, the requirement specification is completed before any development activities start. Once the requirement specification is completed, the requirements are split into requirements. In a purely evolutionary development, the development of the first version starts off after obtaining a rough understanding of what is required. As the development proceeds, more and more requirements emerge. The modern development models, such as the agile mode e neither purely incremental, nor purely evolutionary, but is somewhat in between and is referred to as incremental and evolutionary model. In this are obtained and specified, but requirements that emerge later are accommodated.

2.1 DATA ANALYSIS USING AI

As the word suggests Data Analytics refers to the techniques to analyze data to enhance productivity and business gain. Data is extracted from various sources and is cleaned and categorized to analyze different behavioral patterns. The techniques and the tools used vary according to the organization or individual.

2.1.1 Why is Data Analytics important?

As an enormous amount of data gets generated, the need to extract useful insights is a must for a business enterprise. Data Analytics has a key role in improving your business. Here are 4 main factors which signify the need for Data Analytics:

- **Gather Hidden Insights** – Hidden insights from data are gathered and then analyzed with respect to business requirements.
- **Generate Reports** – Reports are generated from the data and are passed on to the respective teams and individuals to deal with further actions for a high rise in business.
- **Perform Market Analysis** – Market Analysis can be performed to understand the strengths and the weaknesses of competitors.
- **Improve Business Requirement** – Analysis of Data allows improving Business to customer requirements and experience.

2.1.2 Top Tools in Data Analytics

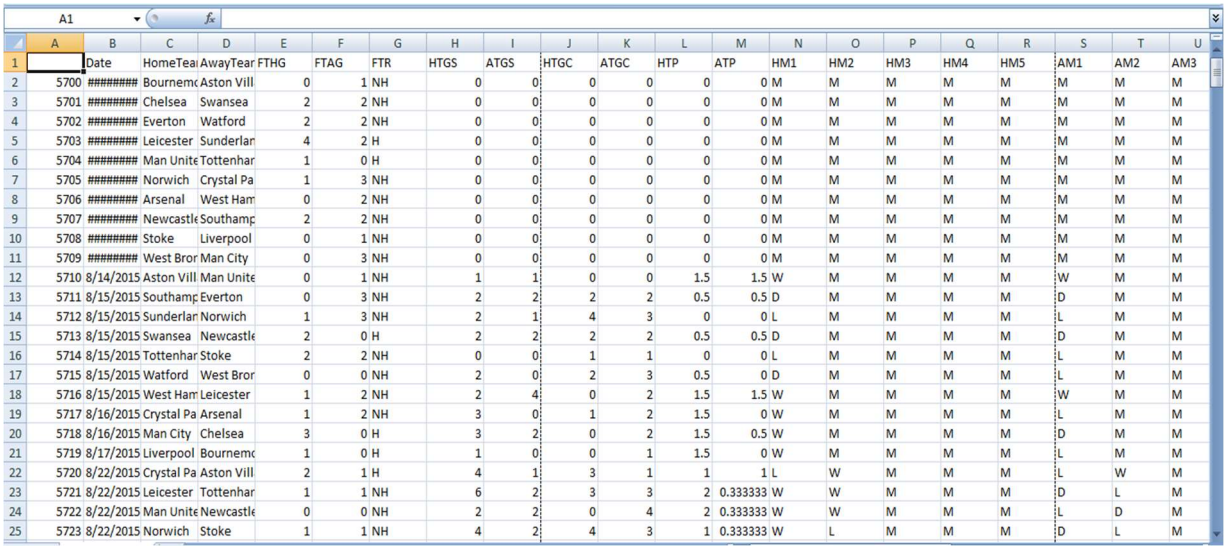
With the increasing demand for Data Analytics in the market, many tools have emerged with various functionalities for this purpose. Either open-source or user-friendly, the top tools in the data analytics market are as follows.

- **R programming** – This tool is the leading analytics tool used for statistics and data modeling. R compiles and runs on various platforms such as UNIX, Windows, and Mac OS. It also provides tools to automatically install all packages as per user-requirement.
- **Python** – Python is an open-source, object-oriented programming language which is easy to read, write and maintain. It provides various machine learning and visualization libraries such as Scikit-learn, TensorFlow, Matplotlib, Pandas, Keras etc. It also can be assembled on any platform like SQL server, a MongoDB database or JSON

- **Tableau Public** – This is a free software that connects to any data source such as Excel, corporate Data Warehouse etc. It then creates visualizations, maps, dashboards etc with real-time updates on the web.
- **QlikView** – This tool offers in-memory data processing with the results delivered to the end-users quickly. It also offers data association and data visualization with data being compressed to almost 10% of its original size.
- **SAS** – A programming language and environment for data manipulation and analytics, this tool is easily accessible and can analyze data from different sources.

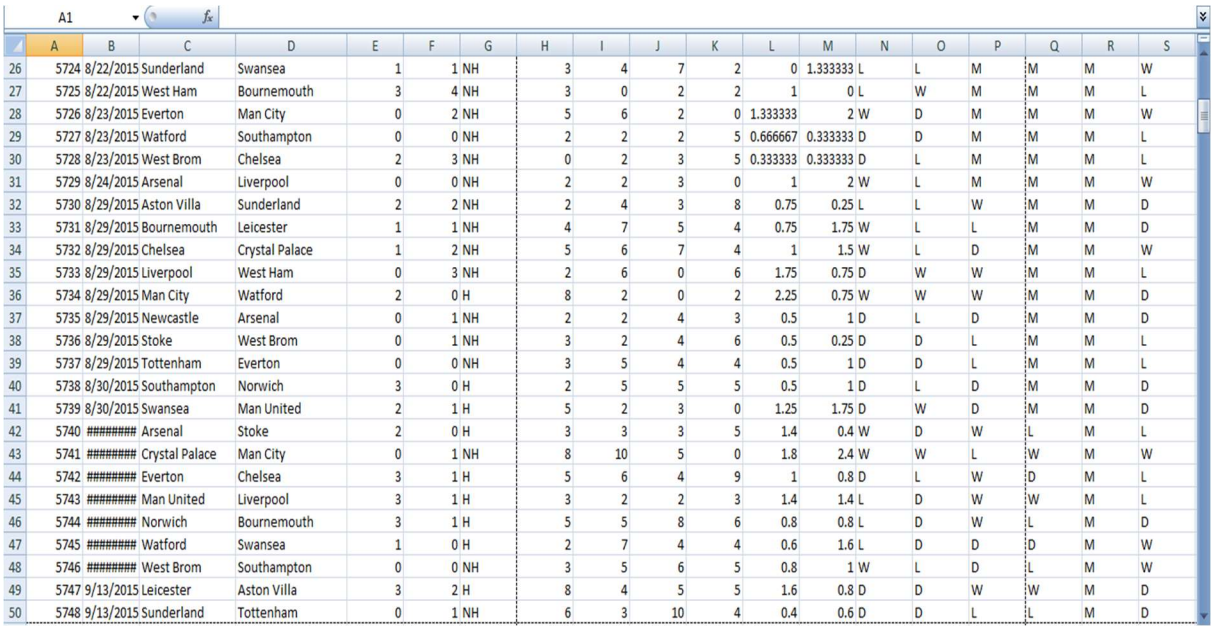
- **Microsoft Excel** – This tool is one of the most widely used tools for data analytics. Mostly usedfor clients' internal data, this tool analyzes the tasks that summarize the data with a preview of pivot tables.
- **RapidMiner** – A powerful, integrated platform that can integrate with any data source types such as Access, Excel, Microsoft SQL, Tera data, Oracle, Sybase etc. This tool is mostly used for predictive analytics, such as data mining, text analytics, machine learning.
- **KNIME** – Konstanz Information Miner (KNIME) is an open-source data analytics platform, which allows you to analyze and model data. With the benefit of visual programming, KNIME provides a platform for reporting and integration through its modular data pipeline concept.
- **OpenRefine** – Also known as GoogleRefine, this data cleaning software will help you clean up data for analysis. It is used for cleaning messy data, the transformation of data and parsing data from websites.
- **Apache Spark** – One of the largest large-scale data processing engine, this tool executes applications in Hadoop clusters 100 times faster in memory and 10 times faster on disk. This tool is also popular for data pipelines and machine learning model development.

2.2 DESCRIPTION OF TRANNING AND TESTING DATASET



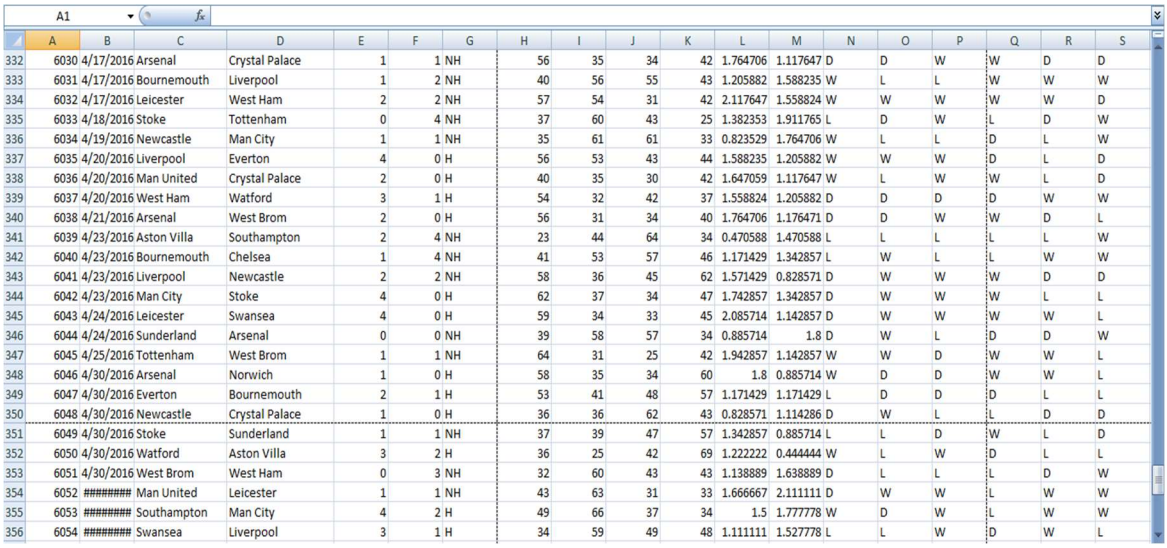
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		Date	HomeTea	AwayTear	FTHG	FTAG	FTR	HTGS	ATGS	HTGC	ATGC	HTP	ATP	HM1	HM2	HM3	HM4	HM5	AM1	AM2	AM3
2	5700	########	Bournemc	Aston Vill	0	1	NH	0	0	0	0	0	0	M	M	M	M	M	M	M	M
3	5701	########	Chelsea	Swansea	2	2	NH	0	0	0	0	0	0	M	M	M	M	M	M	M	M
4	5702	########	Everton	Watford	2	2	NH	0	0	0	0	0	0	M	M	M	M	M	M	M	M
5	5703	########	Leicester	Sunderlar	4	2	H	0	0	0	0	0	0	M	M	M	M	M	M	M	M
6	5704	########	Man Unite	Tottenhar	1	0	H	0	0	0	0	0	0	M	M	M	M	M	M	M	M
7	5705	########	Norwich	Crystal Pa	1	3	NH	0	0	0	0	0	0	M	M	M	M	M	M	M	M
8	5706	########	Arsenal	West Ham	0	2	NH	0	0	0	0	0	0	M	M	M	M	M	M	M	M
9	5707	########	Newcastle	Southamp	2	2	NH	0	0	0	0	0	0	M	M	M	M	M	M	M	M
10	5708	########	Stoke	Liverpool	0	1	NH	0	0	0	0	0	0	M	M	M	M	M	M	M	M
11	5709	########	West Bror	Man City	0	3	NH	0	0	0	0	0	0	M	M	M	M	M	M	M	M
12	5710	8/14/2015	Aston Vill	Man Unite	0	1	NH	1	1	0	0	1.5	1.5	W	M	M	M	M	W	M	M
13	5711	8/15/2015	Southamp	Everton	0	3	NH	2	2	2	2	0.5	0.5	D	M	M	M	M	D	M	M
14	5712	8/15/2015	Sunderlar	Norwich	1	3	NH	2	1	4	3	0	0	L	M	M	M	M	L	M	M
15	5713	8/15/2015	Swansea	Newcastle	2	0	H	2	2	2	2	0.5	0.5	D	M	M	M	M	D	M	M
16	5714	8/15/2015	Tottenhar	Stoke	2	2	NH	0	0	1	1	0	0	L	M	M	M	M	L	M	M
17	5715	8/15/2015	Watford	West Bror	0	0	NH	2	0	2	3	0.5	0	D	M	M	M	M	L	M	M
18	5716	8/15/2015	West Ham	Leicester	1	2	NH	2	4	0	2	1.5	1.5	W	M	M	M	M	W	M	M
19	5717	8/16/2015	Crystal Pa	Arsenal	1	2	NH	3	0	1	2	1.5	0	W	M	M	M	M	L	M	M
20	5718	8/16/2015	Man City	Chelsea	3	0	H	3	2	0	2	1.5	0.5	W	M	M	M	M	D	M	M
21	5719	8/17/2015	Liverpool	Bournemc	1	0	H	1	0	0	1	1.5	0	W	M	M	M	M	L	M	M
22	5720	8/22/2015	Crystal Pa	Aston Vill	2	1	H	4	1	3	1	1	1	L	W	M	M	M	L	W	M
23	5721	8/22/2015	Leicester	Tottenhar	1	1	NH	6	2	3	3	2	0.333333	W	W	M	M	M	D	L	M
24	5722	8/22/2015	Man Unite	Newcastle	0	0	NH	2	2	0	4	2	0.333333	W	W	M	M	M	L	D	M
25	5723	8/22/2015	Norwich	Stoke	1	1	NH	4	2	4	3	1	0.333333	W	L	M	M	M	D	L	M

Figure 2.1 dataset of tranning data



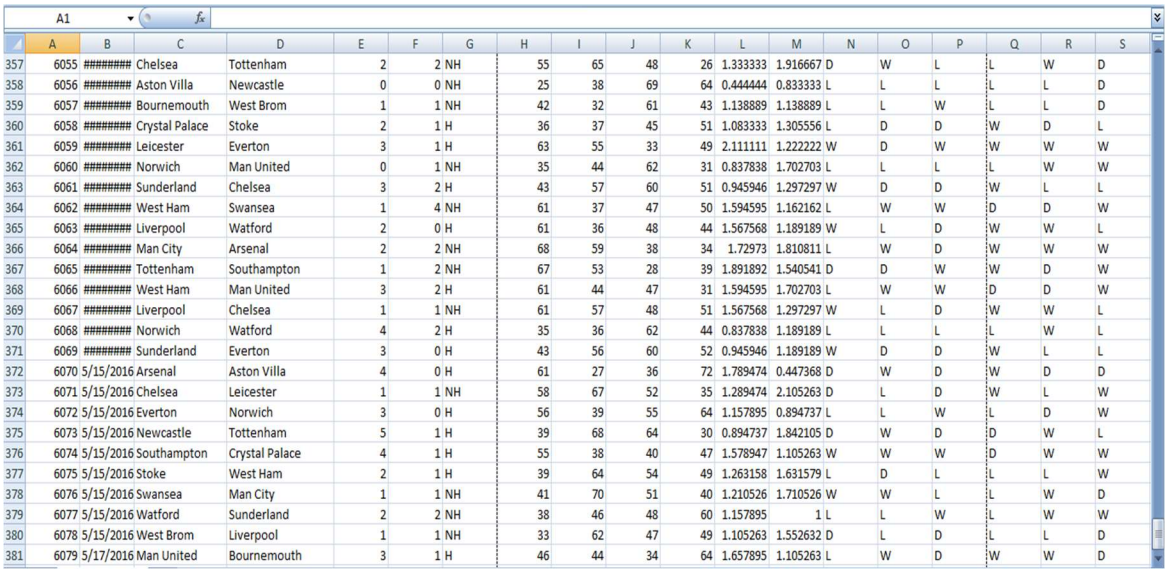
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
26	5724	8/22/2015	Sunderland	Swansea	1	1	NH	3	4	7	2	0	1.333333	L	L	M	M	M	W
27	5725	8/22/2015	West Ham	Bournemouth	3	4	NH	3	0	2	2	1	0	L	W	M	M	M	L
28	5726	8/23/2015	Everton	Man City	0	2	NH	5	6	2	0	1.333333	2	W	D	M	M	M	W
29	5727	8/23/2015	Watford	Southampton	0	0	NH	2	2	2	5	0.666667	0.333333	D	D	M	M	M	L
30	5728	8/23/2015	West Brom	Chelsea	2	3	NH	0	2	3	5	0.333333	0.333333	D	L	M	M	M	L
31	5729	8/24/2015	Arsenal	Liverpool	0	0	NH	2	2	3	0	1	2	W	L	M	M	M	W
32	5730	8/29/2015	Aston Villa	Sunderland	2	2	NH	2	4	3	8	0.75	0.25	L	L	W	M	M	D
33	5731	8/29/2015	Bournemouth	Leicester	1	1	NH	4	7	5	4	0.75	1.75	W	L	L	M	M	D
34	5732	8/29/2015	Chelsea	Crystal Palace	1	2	NH	5	6	7	4	1	1.5	W	L	D	M	M	W
35	5733	8/29/2015	Liverpool	West Ham	0	3	NH	2	6	0	6	1.75	0.75	D	W	W	M	M	L
36	5734	8/29/2015	Man City	Watford	2	0	H	8	2	0	2	2.25	0.75	W	W	W	M	M	D
37	5735	8/29/2015	Newcastle	Arsenal	0	1	NH	2	2	4	3	0.5	1	D	L	D	M	M	D
38	5736	8/29/2015	Stoke	West Brom	0	1	NH	3	2	4	6	0.5	0.25	D	D	L	M	M	L
39	5737	8/29/2015	Tottenham	Everton	0	0	NH	3	5	4	4	0.5	1	D	D	L	M	M	L
40	5738	8/30/2015	Southampton	Norwich	3	0	H	2	5	5	5	0.5	1	D	L	D	M	M	D
41	5739	8/30/2015	Swansea	Man United	2	1	H	5	2	3	0	1.25	1.75	D	W	D	M	M	D
42	5740	########	Arsenal	Stoke	2	0	H	3	3	3	5	1.4	0.4	W	D	W	L	M	L
43	5741	########	Crystal Palace	Man City	0	1	NH	8	10	5	0	1.8	2.4	W	W	L	W	M	W
44	5742	########	Everton	Chelsea	3	1	H	5	6	4	9	1	0.8	D	L	W	D	M	L
45	5743	########	Man United	Liverpool	3	1	H	3	2	2	3	1.4	1.4	L	D	W	W	M	L
46	5744	########	Norwich	Bournemouth	3	1	H	5	5	8	6	0.8	0.8	L	D	W	L	M	D
47	5745	########	Watford	Swansea	1	0	H	2	7	4	4	0.6	1.6	L	D	D	D	M	W
48	5746	########	West Brom	Southampton	0	0	NH	3	5	6	5	0.8	1	W	L	D	L	M	W
49	5747	9/13/2015	Leicester	Aston Villa	3	2	H	8	4	5	5	1.6	0.8	D	D	W	W	M	D
50	5748	9/13/2015	Sunderland	Tottenham	0	1	NH	6	3	10	4	0.4	0.6	D	D	L	L	M	D

Figure 2.2 dataset of tranning data



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
332	6030	4/17/2016	Arsenal	Crystal Palace	1	1	NH	56	35	34	42	1.764706	1.117647	D	D	W	W	D	D
333	6031	4/17/2016	Bournemouth	Liverpool	1	2	NH	40	56	55	43	1.205882	1.588235	W	L	L	W	W	W
334	6032	4/17/2016	Leicester	West Ham	2	2	NH	57	54	31	42	2.117647	1.558824	W	W	W	W	W	D
335	6033	4/18/2016	Stoke	Tottenham	0	4	NH	37	60	43	25	1.382353	1.911765	L	D	W	L	D	W
336	6034	4/19/2016	Newcastle	Man City	1	1	NH	35	61	61	33	0.823529	1.764706	W	L	L	D	L	W
337	6035	4/20/2016	Liverpool	Everton	4	0	H	56	53	43	44	1.588235	1.205882	W	W	W	D	L	D
338	6036	4/20/2016	Man United	Crystal Palace	2	0	H	40	35	30	42	1.647059	1.117647	W	L	W	W	L	D
339	6037	4/20/2016	West Ham	Watford	3	1	H	54	32	42	37	1.558824	1.205882	D	D	D	D	W	W
340	6038	4/21/2016	Arsenal	West Brom	2	0	H	56	31	34	40	1.764706	1.176471	D	D	W	W	D	L
341	6039	4/23/2016	Aston Villa	Southampton	2	4	NH	23	44	64	34	0.470588	1.470588	L	L	L	L	L	W
342	6040	4/23/2016	Bournemouth	Chelsea	1	4	NH	41	53	57	46	1.171429	1.342857	L	W	L	L	W	W
343	6041	4/23/2016	Liverpool	Newcastle	2	2	NH	58	36	45	62	1.571429	0.828571	D	W	W	W	D	D
344	6042	4/23/2016	Man City	Stoke	4	0	H	62	37	34	47	1.742857	1.342857	D	W	W	W	L	L
345	6043	4/24/2016	Leicester	Swansea	4	0	H	59	34	33	45	2.085714	1.142857	D	W	W	W	W	L
346	6044	4/24/2016	Sunderland	Arsenal	0	0	NH	39	58	57	34	0.885714	1.8	D	W	L	D	D	W
347	6045	4/25/2016	Tottenham	West Brom	1	1	NH	64	31	25	42	1.942857	1.142857	W	W	D	W	W	L
348	6046	4/30/2016	Arsenal	Norwich	1	0	H	58	35	34	60	1.8	0.885714	W	D	D	W	W	L
349	6047	4/30/2016	Everton	Bournemouth	2	1	H	53	41	48	57	1.171429	1.171429	L	D	D	D	L	L
350	6048	4/30/2016	Newcastle	Crystal Palace	1	0	H	36	36	62	43	0.828571	1.114286	D	W	L	L	D	D
351	6049	4/30/2016	Stoke	Sunderland	1	1	NH	37	39	47	57	1.342857	0.885714	L	L	D	W	L	D
352	6050	4/30/2016	Watford	Aston Villa	3	2	H	36	25	42	69	1.222222	0.444444	W	L	W	D	L	L
353	6051	4/30/2016	West Brom	West Ham	0	3	NH	32	60	43	43	1.138889	1.638889	D	L	L	L	D	W
354	6052	########	Man United	Leicester	1	1	NH	43	63	31	33	1.666667	2.111111	D	W	W	L	W	W
355	6053	########	Southampton	Man City	4	2	H	49	66	37	34	1.5	1.777778	W	D	W	L	W	W
356	6054	########	Swansea	Liverpool	3	1	H	34	59	49	48	1.111111	1.527778	L	L	W	D	W	L

Figure 2.3 dataset of testing data



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
357	6055	########	Chelsea	Tottenham	2	2	NH	55	65	48	26	1.333333	1.916667	D	W	L	L	W	D
358	6056	########	Aston Villa	Newcastle	0	0	NH	25	38	69	64	0.444444	0.833333	L	L	L	L	L	D
359	6057	########	Bournemouth	West Brom	1	1	NH	42	32	61	43	1.138889	1.138889	L	L	W	L	L	D
360	6058	########	Crystal Palace	Stoke	2	1	H	36	37	45	51	1.083333	1.305556	L	D	D	W	D	L
361	6059	########	Leicester	Everton	3	1	H	63	55	33	49	2.111111	1.222222	W	D	W	W	W	W
362	6060	########	Norwich	Man United	0	1	NH	35	44	62	31	0.837838	1.702703	L	L	L	L	W	W
363	6061	########	Sunderland	Chelsea	3	2	H	43	57	60	51	0.945946	1.297297	W	D	D	W	L	L
364	6062	########	West Ham	Swansea	1	4	NH	61	37	47	50	1.594595	1.162162	L	W	W	D	D	W
365	6063	########	Liverpool	Watford	2	0	H	61	36	48	44	1.567568	1.189189	W	L	D	W	W	L
366	6064	########	Man City	Arsenal	2	2	NH	68	59	38	34	1.72973	1.810811	L	W	D	W	W	W
367	6065	########	Tottenham	Southampton	1	2	NH	67	53	28	39	1.891892	1.540541	D	D	W	W	D	W
368	6066	########	West Ham	Man United	3	2	H	61	44	47	31	1.594595	1.702703	L	W	W	D	D	W
369	6067	########	Liverpool	Chelsea	1	1	NH	61	57	48	51	1.567568	1.297297	W	L	D	W	W	L
370	6068	########	Norwich	Watford	4	2	H	35	36	62	44	0.837838	1.189189	L	L	L	L	W	L
371	6069	########	Sunderland	Everton	3	0	H	43	56	60	52	0.945946	1.189189	W	D	D	W	L	L
372	6070	5/15/2016	Arsenal	Aston Villa	4	0	H	61	27	36	72	1.789474	0.447368	D	W	D	W	D	D
373	6071	5/15/2016	Chelsea	Leicester	1	1	NH	58	67	52	35	1.289474	2.105263	D	L	D	W	L	W
374	6072	5/15/2016	Everton	Norwich	3	0	H	56	39	55	64	1.157895	0.894737	L	L	W	L	D	W
375	6073	5/15/2016	Newcastle	Tottenham	5	1	H	39	68	64	30	0.894737	1.842105	D	W	D	D	W	L
376	6074	5/15/2016	Southampton	Crystal Palace	4	1	H	55	38	40	47	1.578947	1.105263	W	W	W	D	W	W
377	6075	5/15/2016	Stoke	West Ham	2	1	H	39	64	54	49	1.263158	1.631579	L	D	L	L	L	W
378	6076	5/15/2016	Swansea	Man City	1	1	NH	41	70	51	40	1.210526	1.710526	W	W	L	L	W	D
379	6077	5/15/2016	Watford	Sunderland	2	2	NH	38	46	48	60	1.157895	1	L	L	W	L	W	W
380	6078	5/15/2016	West Brom	Liverpool	1	1	NH	33	62	47	49	1.105263	1.552632	D	L	D	L	L	D
381	6079	5/17/2016	Man United	Bournemouth	3	1	H	46	44	34	64	1.657895	1.105263	L	W	D	W	W	D

Figure 2.4 dataset of testing data

2.3 KNN Algorithm

K nearest neighbors or KNN Algorithm is a simple algorithm which uses the entire dataset in its training phase. Whenever a prediction is required for an unseen data instance, it searches through the entire training dataset for k-most similar instances and the data with the most similar instance is finally returned as the prediction.

How does a KNN Algorithm work?

The k-nearest neighbors algorithm uses a very simple approach to perform classification. When tested with a new example, it looks through the training data and finds the k training examples that are closest to the new example. It then assigns the most common class label (among those k-training examples) to the test example.

What does 'k' in kNN Algorithm represent?

k in kNN algorithm represents the number of nearest neighbor points which are voting for the new test data's class.

If k=1, then test examples are given the same label as the closest example in the training set.

If k=3, the labels of the three closest classes are checked and the most common (i.e., occurring at least twice) label is assigned, and so on for larger k.

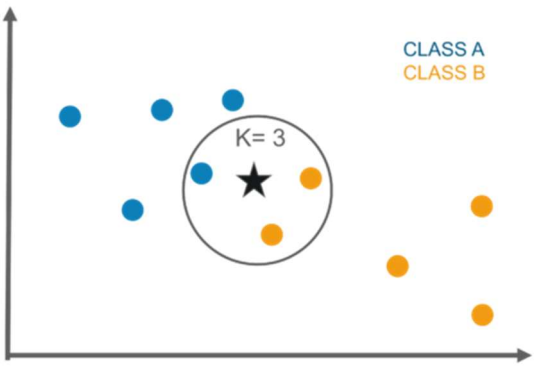


Figure 2.5 **kNN Algorithm represent**

- KNN Algorithm Manual Implementation

Step1: Calculate the Euclidean distance between the new point and the existing points

Step 2**:** Choose the value of K and select K neighbors closet to the new point.

Step 3: Count the votes of all the K neighbors / Predicting Values

- **Implementation of kNN Algorithm using Python**

- **Handling the data**

  The very first step will be handling the iris dataset. Open the dataset using the open function and read the data lines with the reader function available under the csv module.

- **Calculate the distance**

  In order to make any predictions, you have to calculate the distance between the new point and the existing points, as you will be needing k closest points.

- **Find k nearest point**

  Now that you have calculated the distance from each point, we can use it collect the k most similar points/instances for the given test data/instance.

- **Predict the class**

  Now that you have the k nearest points/neighbors for the given test instance, the next task is to predicted response based on those neighbors

- **Check the accuracy**

  Now that we have all of the pieces of the KNN algorithm in place. Let's check how accurate our prediction.

2.4 DECISION TREE CLASSIFIER

A ***Decision Tree*** has many analogies in real life and turns out, it has influenced a wide area of **Machine Learning**, covering both **Classification** and **Regression**. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making.

- **What is a Decision Tree?**

A decision tree is a map of the possible outcomes of a series of related choices. It allows an individual or organization to weigh possible actions against one another based on their costs, probabilities, and benefits.

As the name goes, it uses a tree-like model of decisions. They can be used either to drive informal discussion or to map out an algorithm that predicts the best choice mathematically.

A decision tree typically starts with a single node, which branches into possible outcomes. Each of those outcomes leads to additional nodes, which branch off into other possibilities. This gives it a tree-like shape.

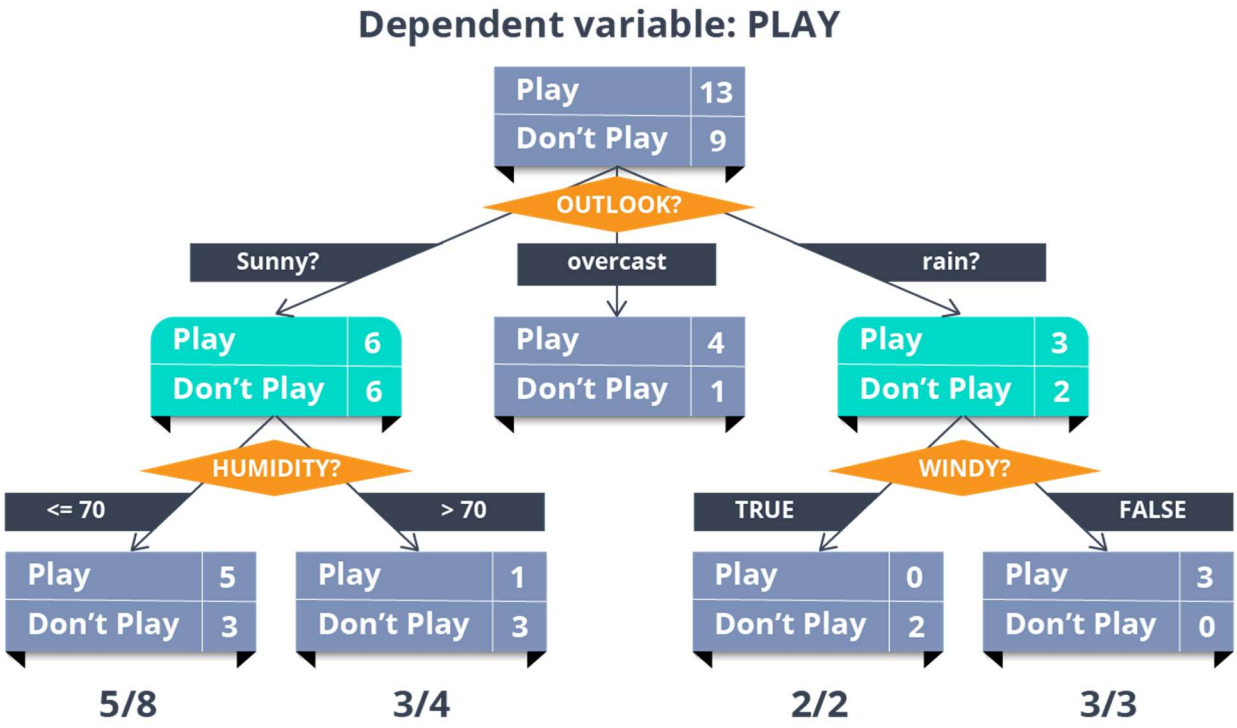


Figure 2.6 Decetion tree

There are three different types of nodes: chance nodes, decision nodes, and end nodes. A chance node, represented by a circle, shows the probabilities of certain results. A decision node, represented by a square, shows a decision to be made, and an end node shows the final outcome of a decision path.

- Advantages & Disadvantages of Decision Trees

Advantages

- Decision trees generate understandable rules.
- Decision trees perform classification without requiring much computation.
- Decision trees are capable of handling both continuous and categorical variables.
- Decision trees provide a clear indication of which fields are most important for prediction or classification.

Disadvantages

- Decision trees are less appropriate for estimation tasks where the goal is to predict the value of a continuous attribute.
- Decision trees are prone to errors in classification problems with many class and a relatively small number of training examples.
- Decision trees can be computationally expensive to train. The process of growing a decision tree is computationally expensive. At each node, each candidate splitting field must be sorted before its best split can be found. In some algorithms, combinations of fields are used and a
- search must be made for optimal combining weights. **Pruning algorithms** can also be expensive since many candidate sub-trees must be formed and compared.

- **Creating a Decision Tree**

Let us consider a scenario where a new planet is discovered by a group of astronomers. Now the question is whether it could be 'the next earth?' The answer to this question will revolutionize the way people live. Well, literally!

There is n number of deciding factors which need to be thoroughly researched to take an intelligent decision. These factors can be whether water is present on the planet, what is the temperature, whether the surface is prone to continuous storms, flora and fauna survives the climate or not, etc.

Let us create a decision tree to find out whether we have discovered a new habitat.

The habitable temperature falls into the range 0 to 100 Celsius.

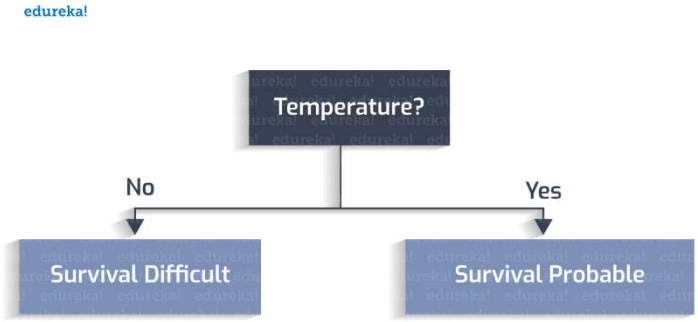


Figure 2.7 : Creating a Decision Tree

- **Whether water is present or not?**

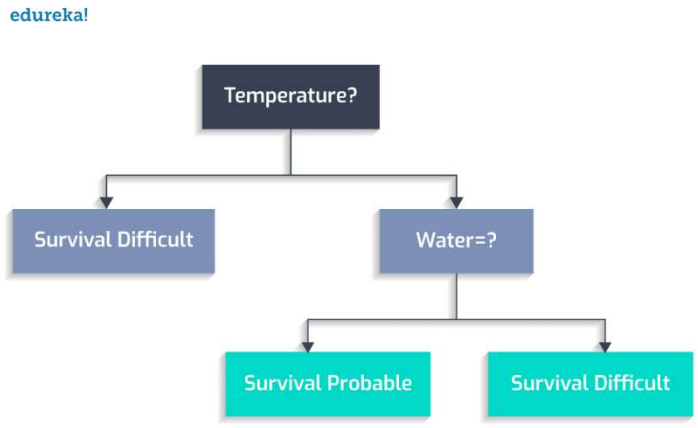


Figure 2.8 : Creating a Decision Tree

- **Whether flora and fauna flourishes?**

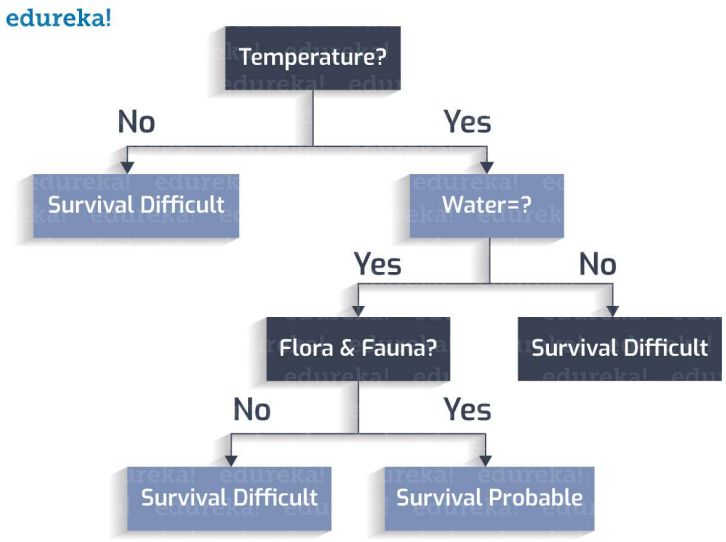


Figure 2.9: Creating a Decision Tree

- **The planet has a stormy surface?**

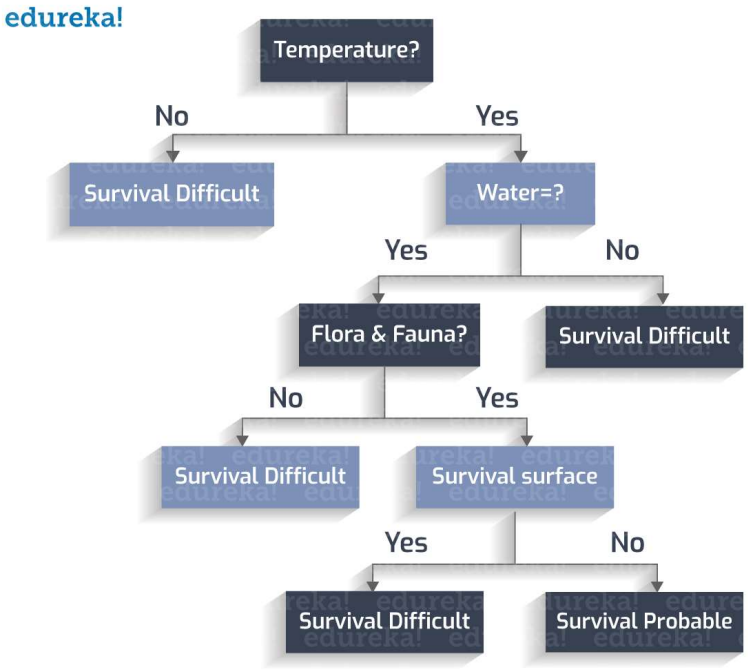


Figure 2.10 : Creating a Decision Tree

- Classification Rules:

Classification rules are the cases in which all the scenarios are taken into consideration and a class variable is assigned to each.

Class Variable:

Each leaf node is assigned a class-variable. A class-variable is the final output which leads to our decision.

Let us derive the classification rules from the Decision Tree created:

1. If Temperature is not between 273 to 373K, -> Survival Difficult

2. If Temperature is between 273 to 373K, and water is not present, -> Survival Difficult

3. If Temperature is between 273 to 373K, water is present, and flora and fauna is not present -> Survival Difficult

4. If Temperature is between 273 to 373K, water is present, flora and fauna is present, and a stormy surface is not present -> Survival Probable

5. If Temperature is between 273 to 373K, water is present, flora and fauna is present, and a stormy surface is present -> Survival Difficult

- **Decision Tree**

A decision tree has the following constituents :

- Root Node: The factor of 'temperature' is considered as the root in this case.
- Internal Node: The nodes with one incoming edge and 2 or more outgoing edges.
- Leaf Node: This is the terminal node with no out-going edge.

As the decision tree is now constructed, starting from the root-node we check the test condition and assign the control to one of the outgoing edges, and so the condition is again tested and a node is assigned. The decision tree is said to be complete when all the test conditions lead to a leaf node. The leaf node contains the class-labels, which vote in favor or against the decision.

Now, you might think why did we start with the 'temperature' attribute at the root? If you choose any other attribute, the decision tree constructed will be different.Correct. For a particular set of attributes, there can be numerous different trees created. We need to choose the optimal tree which is done by following an algorithmic approach. We will now see 'the greedy approach' to create a perfect decision tree.

- **Random Forest Classification**

Classification is the method of predicting the class of a given input data point. Classification problems are common in machine learning and they fall under the Supervised learning method.

Let's say you want to classify your emails into 2 groups, spam and non-spam emails. For this kind of problems, where you have to assign an input data point into different classes, you can make use of classification algorithms.

Under classification we have 2 types:

- Binary Classification
- Multi-Class Classification

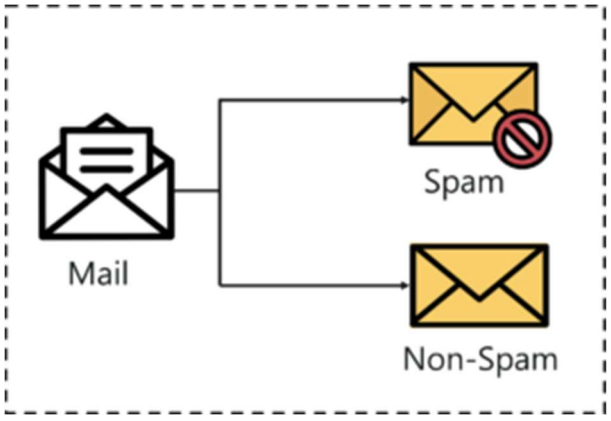


Figure 2.11 : Example

The example that I gave earlier about classifying emails as spam and non-spam is of binary type because here we're classifying emails into 2 classes (spam and non-spam).

But let's say that we want to classify our emails into 3 classes:

- Spam messages
- Non-Spam messages
- Drafts

So here we're classifying emails into more than 2 classes, this is exactly what multi-class classification means.

One more thing to note here is that it is common for classification models to predict a continuous value. But this continuous value represents the probability of a given data point belonging to each output class.

2.5 What Is Random Forest?

Random forest algorithm is a supervised classification and regression algorithm. As the name suggests, this algorithm randomly creates a forest with several trees.

Generally, the more trees in the forest the more robust the forest looks like. Similarly, in the random forest classifier, the higher the number of trees in the forest, greater is the accuracy of the results.

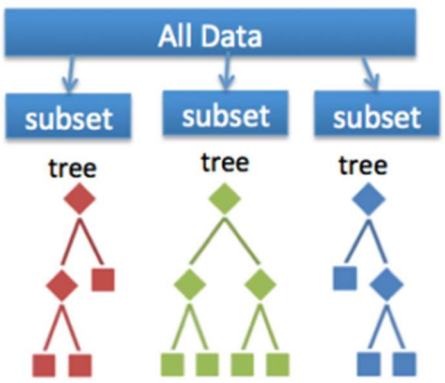


Figure 2.12 : Random Forest

In simple words, Random forest builds multiple decision trees (called the forest) and glues them together to get a more accurate and stable prediction. The forest it builds is a collection of Decision Trees, trained with the bagging method.

Before we discuss Random Forest in depth, we need to understand how Decision Trees work.

- ## What Is The Difference Between Random Forest And Decision Trees?

Let's say that you're looking to buy a house, but you're unable to decide which one to buy. So, you consult a few agents and they give you a list of parameters that you should consider before buying a house. The list includes:

- Price of the house
- Locality
- Number of bedrooms
- Parking space
- Available facilities

These parameters are known as *predictor variables*, which are used to find the response variable. Here's a diagrammatic illustration of how you can represent the above problem statement using a decision tree.

- ## Why Use Random Forest?

You might be wondering why we use Random Forest when we can solve the same problems using Decision trees. Let me explain.

- Even though Decision trees are convenient and easily implemented, they lack accuracy. Decision trees work very effectively with the training data that was used to build them, but they're not flexible when it comes to classifying the new sample. Which means that the accuracy during testing phase is very low.
- This happens due to a process called Over-fitting.
- This means that the disturbance in the training data is recorded and learned as concepts by the model. But the problem here is that these concepts do not

apply to the testing data and negatively impact the model's ability to classify the new data, hence reducing the accuracy on the testing data.

This is where Random Forest comes in. It is based on the idea of bagging, which is used to reduce the variation in the predictions by combining the result of multiple Decision trees on different samples of the data set.

How Does Random Forest Work?

To understand Random forest, consider the below sample data set. In this data set we have four predictor variables, namely:

- Weight
- Blood flow
- Blocked Arteries
- Chest Pain

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Abnormal	No	No	130	No
Normal	Yes	Yes	195	Yes
Normal	No	Yes	218	No
Abnormal	Yes	Yes	180	Yes

Figure 2.13 : Random Forest

These variables are used to predict whether or not a person has heart disease. We're going to use this data set to create a Random Forest that predicts if a person has heart disease or not.

- **Creating A Random Forest**

Step 1: Create a Bootstrapped Data Set

Bootstrapping is an estimation method used to make predictions on a data set by re-sampling it. To create a bootstrapped data set, we must randomly select samples from the original data set. A point to note here is that we can select the same sample more than once.

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Normal	Yes	Yes	195	Yes
Abnormal	No	No	130	No
Abnormal	Yes	Yes	180	Yes
Abnormal	Yes	Yes	180	Yes

Figure 2.14 : Random Forest

In the above figure, I have randomly selected samples from the original data set and created a bootstrapped data set. Simple, isn't it? Well, in real-world problems you'll never get such a small data set, thus creating a bootstrapped data set is a little more complex.

Step 2: Creating Decision Trees

- Our next task is to build a Decision Tree by using the bootstrapped data set created in the previous step. Since we're making a Random Forest we will not consider the entire data set that we created, instead we'll only use a random subset of variables at each step.
- In this example, we're only going to consider two variables at each step. So, we begin at the root node, here we randomly select two variables as candidates for the root node.
- Let's say we selected Blood Flow and Blocked arteries. Out of these 2 variables, we must now select the variable that best separates the samples. For

the sake of this example, let's say that Blocked Arteries is a more significant predictor and thus assign it as the root node.

- Our next step is to repeat the same process for each of the upcoming branch nodes. Here, we again select two variables at random as candidates for the branch node and then choose a variable that best separates the samples.

Just like this, we build the tree by only considering random subsets of variables at each step. By following the above process, our tree would look something like this:

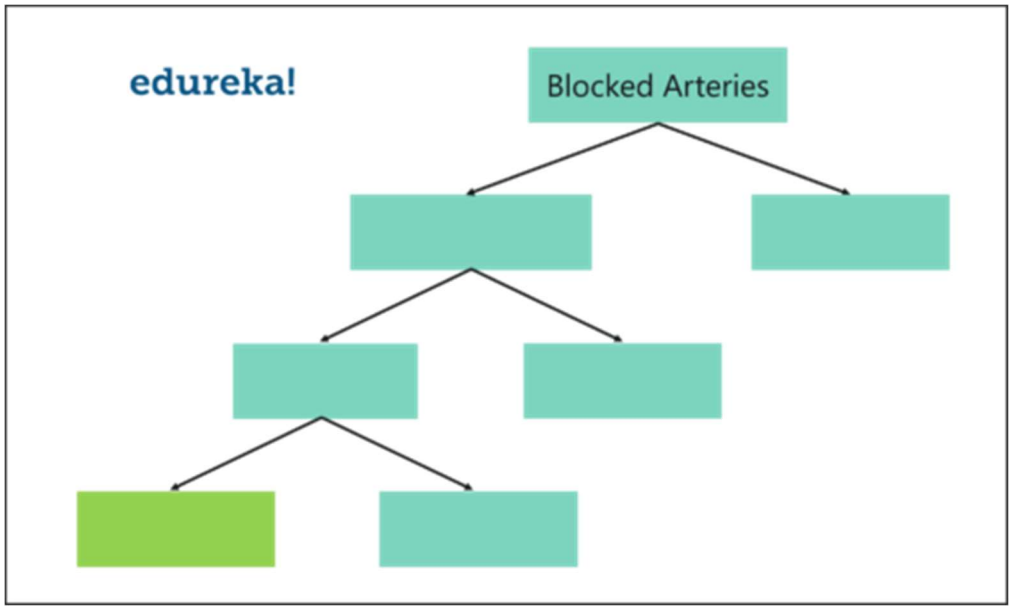


Figure 2.15 : Decetion tree

Step 3: Go back to Step 1 and Repeat

Like I mentioned earlier, Random Forest is a collection of Decision Trees. Each Decision Tree predicts the output class based on the respective predictor variables used in that tree. Finally, the outcome of all the Decision Trees in a Random Forest is recorded and the class with the majority votes is computed as the output class.

Thus, we must now create more decision trees by considering a subset of random predictor variables at each step. To do this, go back to step 1, create a new bootstrapped data set and then build a Decision Tree by considering only a subset of variables at each step. So, by following the above steps, our Random Forest would look something like this:

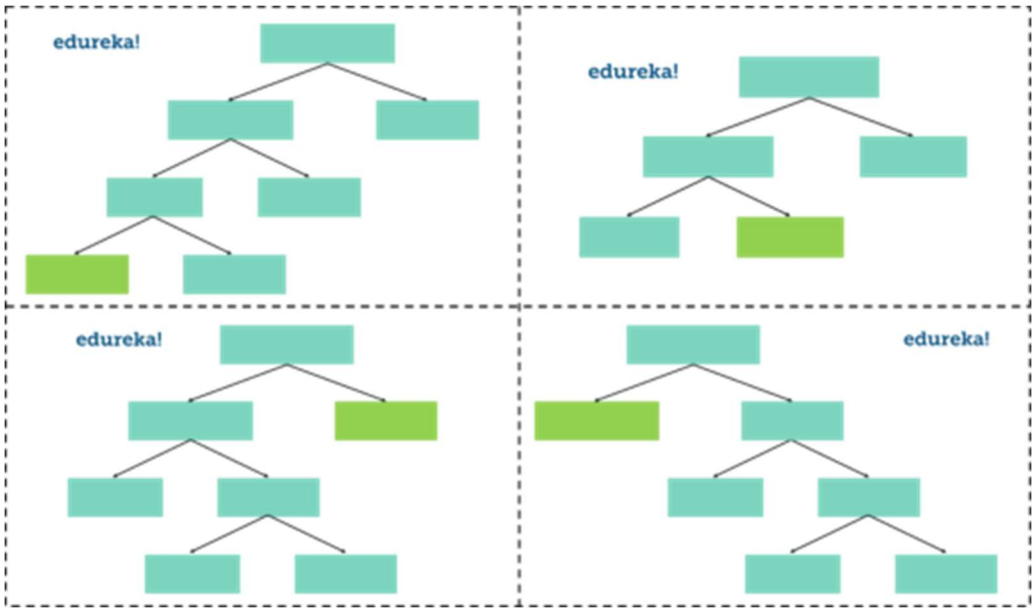


Figure 2.16 : Decetion Tree

This iteration is performed 100's of times, therefore creating multiple decision trees with each tree computing the output, by using a subset of randomly selected variables at each step.

Having such a variety of Decision Trees in a Random Forest is what makes it more effective than an individual Decision Tree created using all the features and the whole data set.

Step 4: Predicting the outcome of a new data point

Now that we've created a random forest, let's see how it can be used to predict whether a new patient has heart disease or not. The below diagram has the data about the new patient. All we have to do is run this data down the decision trees that we made.

The first tree shows that the patient has heart disease, so we keep a track of that in a table as shown in the figure.

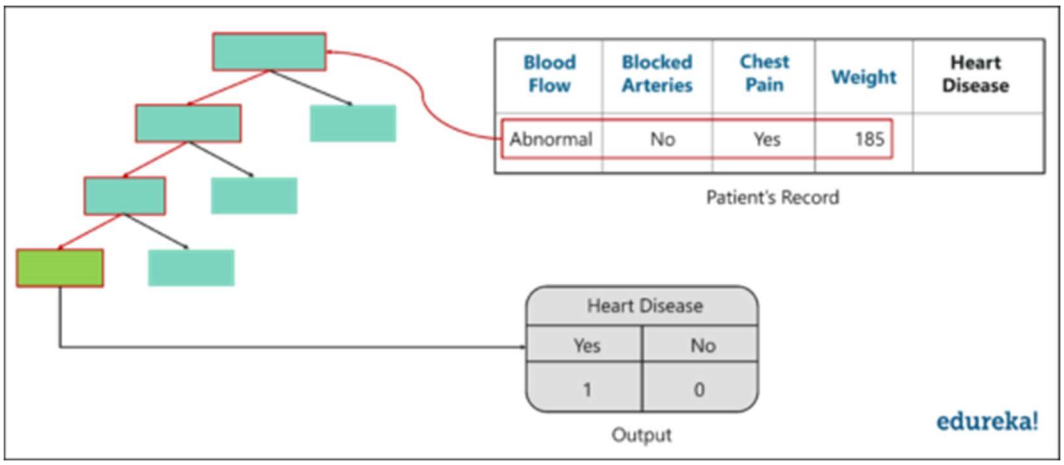


Figure 2.17 :**Output – Random Forest**

Similarly, we run this data down the other decision trees and keep a track of the class predicted by each tree. After running the data down all the trees in the Random Forest, we check which class got the majority votes. In our case, the class 'Yes' received the most number of votes, hence it's clear that the new patient has heart disease.

To conclude, we bootstrapped the data and used the aggregate from all the trees to make a decision, this process is known as Bagging.

Step 5: Evaluate the Model

Our final step is to evaluate the Random Forest model. Earlier while we created the bootstrapped data set, we left out one entry/sample since we duplicated another sample. In a real-world problem, about 1/3rd of the original data set is not included in the bootstrapped data set.

The below figure shows the entry that didn't end up in the bootstrapped data set.

Blood Flow	Blocked Arteries	Chest Pain	Weight	Heart Disease
Normal	No	Yes	218	No

Figure 2.18:This sample data set that does not include in the bootstrapped data set is known as the Out-Of-Bag (OOB) data set.

- **Why Is Boosting Used?**

To solve convoluted problems we require more advanced techniques. Let's suppose that on given a data set of images containing images of cats and dogs, you were asked to build a model that can classify these images into two separate classes. Like every other person, you will start by identifying the images by using some rules, like given below:

1. The image has pointy ears: Cat
2. The image has cat shaped eyes: Cat
3. The image has bigger limbs: Dog
4. The image has sharpened claws: Cat
5. The image has a wider mouth structure: Dog

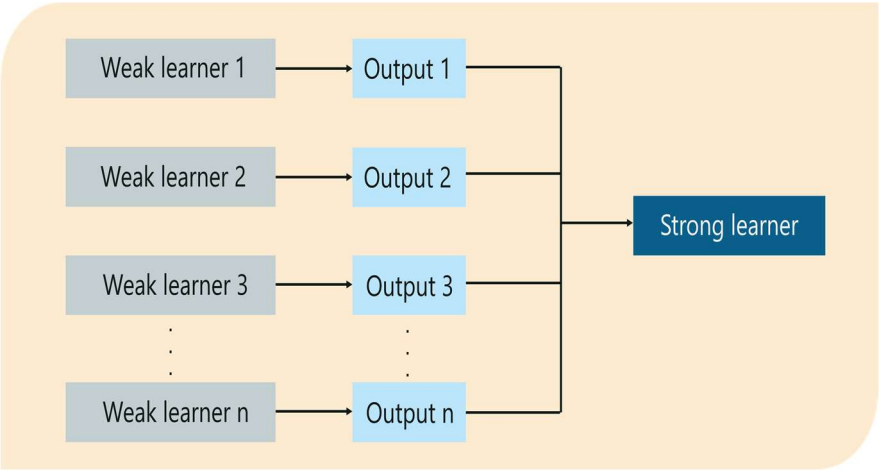
All these rules help us identify whether an image is a Dog or a cat, however, if we were to classify an image based on an individual (single) rule, the prediction would be flawed. Each of these rules, individually, are called weak learners because these rules are not strong enough to classify an image as a cat or dog.

Therefore, to make sure that our prediction is more accurate, we can combine the prediction from each of these weak learners by using the majority rule or weighted average. This makes a strong learner model.

In the above example, we have defined 5 weak learners and the majority of these rules (i.e. 3 out of 5 learners predict the image as a cat) gives us the prediction that the image is a cat. Therefore, our final output is a cat.

What Is Boosting?

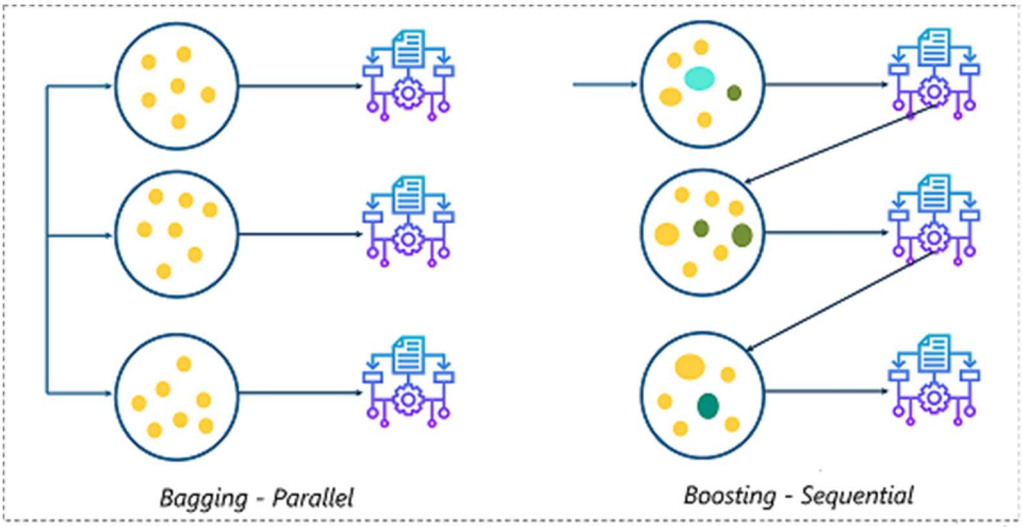
Boosting is an ensemble learning technique that uses a set of Machine Learning



algorithms to convert weak learner to strong learners in order to increase the accuracy of the model.

What Is Ensemble In Machine Learning?

Ensemble learning is a method that is used to enhance the performance of Machine Learning model by combining several learners. When compared to a single model, this type of learning builds models with improved efficiency and accuracy. This is exactly why ensemble methods are used to win market leading competitions such as the Netflix recommendation competition, Kaggle competitions and so on.



Boosting vs Bagging

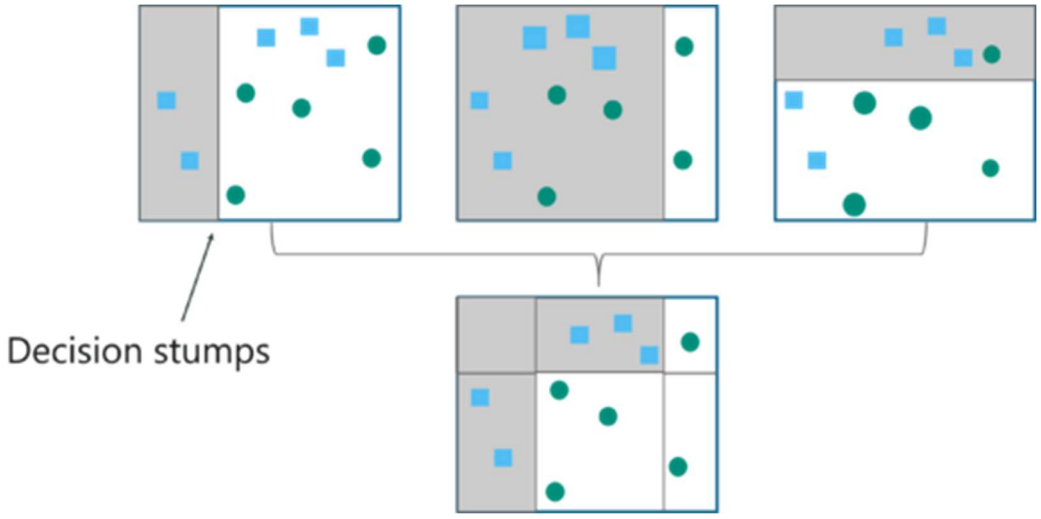
Ensemble learning can be performed in two ways:

1. **Sequential ensemble,** popularly known as ***boosting***, here the weak learners are sequentially produced during the training phase. The performance of the model is improved by assigning a higher weightage to the previous, incorrectly classified samples. An example of boosting is the AdaBoost algorithm.
2. **Parallel ensemble**, popularly known as ***bagging***, here the weak learners are produced parallelly during the training phase. The performance of the model can be increased by parallelly training a number of weak learners on bootstrapped data sets. An example of bagging is the Random Forest algorithm.

In this blog, I'll be focusing on the Boosting method, so in the below section we will understand how the boosting algorithm works.

How Boosting Algorithm Works?

The basic principle behind the working of the boosting algorithm is to generate multiple weak learners and combine their predictions to form one strong rule. These weak rules are generated by applying base Machine Learning algorithms on different distributions of the data set. These algorithms generate weak rules for each iteration. After multiple iterations, the weak learners are combined to form a strong learner that will predict a more accurate outcome.



Step 1: The base algorithm reads the data and assigns equal weight to each sample observation.

Step 2: False predictions made by the base learner are identified. In the next iteration, these false predictions are assigned to the next base learner with a higher weightage on these incorrect predictions.

Step 3: Repeat step 2 until the algorithm can correctly classify the output.

Therefore, the main aim of Boosting is to focus more on miss-classified predictions.

Now that we know how the boosting algorithm works, let's understand the different types of boosting techniques.

Types Of Boosting

There are three main ways through which boosting can be carried out:

1. Adaptive Boosting or AdaBoost
2. Gradient Boosting
3. XGBoost

I'll be discussing the basics behind each of these types.

Adaptive Boosting

- AdaBoost is implemented by combining several weak learners into a single strong learner.
- The weak learners in AdaBoost take into account a single input feature and draw out a single split decision tree called the decision stump. Each observation is weighed equally while drawing out the first decision stump.
- The results from the first decision stump are analyzed and if any observations are wrongfully classified, they are assigned higher weights.
- Post this, a new decision stump is drawn by considering the observations with higher weights as more significant.
- Again if any observations are misclassified, they're given higher weight and this process continues until all the observations fall into the right class.
- Adaboost can be used for both classification and regression-based problems, however, it is more commonly used for classification purpose.

Gradient Boosting

Gradient Boosting is also based on sequential ensemble learning. Here the base learners are generated sequentially in such a way that the present base learner is always more effective than the previous one, i.e. the overall model improves sequentially with each iteration.

The difference in this type of boosting is that the weights for misclassified outcomes are not incremented, instead, Gradient Boosting method tries to optimize the loss

function of the previous learner by adding a new model that adds weak learners in order to reduce the loss function.

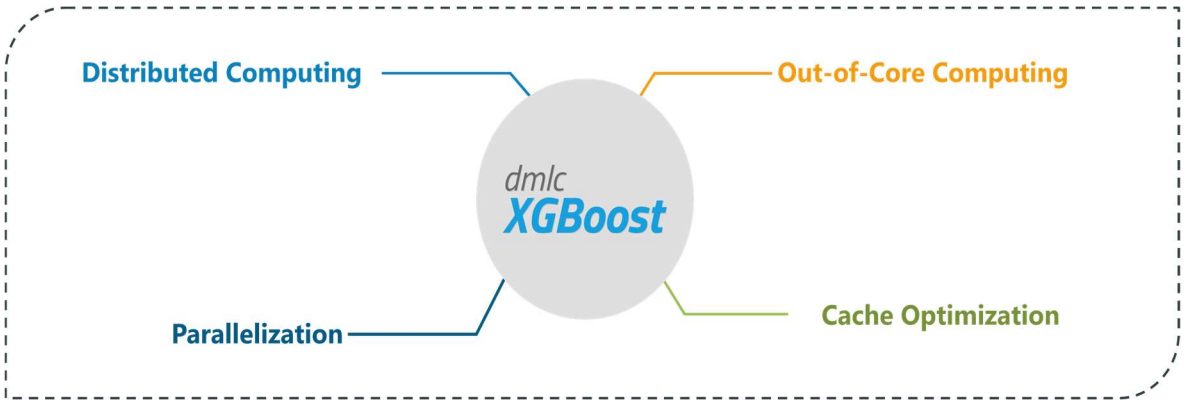
The main idea here is to overcome the errors in the previous learner's predictions. This type of boosting has three main components:

1. **Loss function** that needs to be ameliorated.
2. **Weak learner** for computing predictions and forming strong learners.
3. An **Additive Model** that will regularize the loss function.

XGBoost

XGBoost is an advanced version of Gradient boosting method, it literally means eXtreme Gradient Boosting. XGBoost developed by Tianqi Chen, falls under the category of Distributed Machine Learning Community (DMLC).

The main aim of this algorithm is to increase the speed and efficiency of computation. The Gradient Descent Boosting algorithm computes the output at a slower rate since they sequentially analyze the data set, therefore XGBoost is used to boost or extremely boost the performance of the model.



XGBoost is designed to focus on computational speed and model efficiency. The main features provided by XGBoost are:

- Parallelly creates decision trees.
- Implementing distributed computing methods for evaluating large and complex models.
- Using Out-of-Core Computing to analyze huge datasets.
- Implementing cache optimization to make the best use of resources.

So these were the different types of Boosting Machine Learning algorithms. To make things interesting, in the below section we will run a demo to see how boosting algorithms can be implemented in Python.

2.6 CODE :

```
In [ 1]:
#data preprocessing
import pandas as pd
#produces a prediction model in the form of an ensemble of weak prediction models, typically decision tree
import xgboost as xgb
#the outcome (dependent variable) has only a limited number of possible values.
#Logistic Regression is used when response variable is categorical in nature.
from sklearn.linear_model import LogisticRegression
#A random forest is a meta estimator that fits a number of decision tree classifiers
#on various sub-samples of the dataset and use averaging to improve the predictive
#accuracy and control over-fitting.
from sklearn.ensemble import RandomForestClassifier
#a discriminative classifier formally defined by a separating hyperplane.
from sklearn.svm import SVC
#displayd data
from IPython.display import display
%matplotlib inline



In [2]:
# Read data and drop redundant column.
data = pd.read_csv('final_dataset.csv')

# Preview data.
display(data.head())
```

```
27.
28.
29. #Full Time Result (H=Home Win, D=Draw, A=Away Win)
30. #HTGD - Home team goal difference
31. #ATGD - away team goal difference
32. #HTP - Home team points
33. #ATP - Away team points
34. #DiffFormPts Diff in points
35. #DiffLP - Differnece in last years prediction
36.
37. #Input - 12 other features (fouls, shots, goals, misses,corners, red card,
    yellow cards)
38. #Output - Full Time Result (H=Home Win, D=Draw, A=Away Win)
39.
40.
41.
42. In [3]:
43. #what is the win rate for the home team?
44.
45. # Total number of matches.
46. n_matches = data.shape[0]
47.
48. # Calculate number of features. -1 because we are saving one as the
    target variable (win/lose/draw)
49. n_features = data.shape[1] - 1
50.
51. # Calculate matches won by home team.
52. n_homewins = len(data[data.FTR == 'H'])
53.
54. # Calculate win rate for home team.
55. win_rate = (float(n_homewins) / (n_matches)) * 100
56.
57. # Print the results
58. print "Total number of matches: {}".format(n_matches)
```

```
print "Number of features: {}".format(n_features)
print "Number of matches won by home team: {}".format(n_homewins)
print "Win rate of home team: {:.2f}%".format(win_rate)


In [4]:
# Visualising distribution of data
from pandas.tools.plotting import scatter_matrix

#the scatter matrix is plotting each of the columns specified against each other column.
#You would have observed that the diagonal graph is defined as a histogram, which means that in the
#section of the plot matrix where the variable is against itself, a histogram is plotted.

#Scatter plots show how much one variable is affected by another.
#The relationship between two variables is called their correlation
#negative vs positive correlation

#HTGD - Home team goal difference
#ATGD - away team goal difference
#HTP - Home team points
#ATP - Away team points
#DiffFormPts Diff in points
#DiffLP - Differnece in last years prediction

scatter_matrix(data[['HTGD','ATGD','HTP','ATP','DiffFormPts','DiffLP']], figsize=(10,10))


```

```
88.
89.
90.
91.
92. In [5]:
93. # Separate into feature set and target variable
94. #FTR = Full Time Result (H=Home Win, D=Draw, A=Away Win)
95. X_all = data.drop(['FTR'],1)
96. y_all = data['FTR']
97.
98. # Standardising the data.
99. from sklearn.preprocessing import scale
100.
101.        #Center to the mean and component wise scale to unit variance.
102.        cols = [['HTGD','ATGD','HTP','ATP','DiffLP']]
103.        for col in cols:
104.            X_all[col] = scale(X_all[col])
105.
106.
107.
108.
109.
110.        In [6]:
111.        #last 3 wins for both sides
112.        X_all.HM1 = X_all.HM1.astype('str')
113.        X_all.HM2 = X_all.HM2.astype('str')
114.        X_all.HM3 = X_all.HM3.astype('str')
115.        X_all.AM1 = X_all.AM1.astype('str')
116.        X_all.AM2 = X_all.AM2.astype('str')
117.        X_all.AM3 = X_all.AM3.astype('str')
118.
119.        #we want continous vars that are integers for our input data, so
    lets remove any categorical vars
120.        def preprocess_features(X):
```

```
121.        ''' Preprocesses the football data and converts catagorical
    variables into dummy variables. '''
122.
123.        # Initialize new output DataFrame
124.        output = pd.DataFrame(index = X.index)
125.
126.        # Investigate each feature column for the data
127.        for col, col_data in X.iteritems():
128.
129.            # If data type is categorical, convert to dummy variables
130.            if col_data.dtype == object:
131.                col_data = pd.get_dummies(col_data, prefix = col)
132.
133.            # Collect the revised columns
134.            output = output.join(col_data)
135.
136.        return output
137.
138.    X_all = preprocess_features(X_all)
139.    print "Processed feature columns ({} total
    features):\n{}".format(len(X_all.columns), list(X_all.columns))
140.
141.
142.
143.
144.
145.
146.    In [7]:
147.    # Show the feature information by printing the first five rows
148.    print "\nFeature values:"
149.    display(X_all.head())
150.
151.
152.
```

```
153.
154.
155.
156.        In [8]:
157.        from sklearn.cross_validation import train_test_split
158.
159.        # Shuffle and split the dataset into training and testing set.
160.        X_train, X_test, y_train, y_test = train_test_split(X_all, y_all,
161.          test_size = 50,
162.        random_state = 2,
163.         stratify = y_all)
164.
165.
166.
167.
168.
169.
170.        In [9]:
171.        #for measuring training time
172.        from time import time
173.        # F1 score (also F-score or F-measure) is a measure of a test's
    accuracy.
174.        #It considers both the precision p and the recall r of the test to
    compute
175.        #the score: p is the number of correct positive results divided
    by the number of
176.        #all positive results, and r is the number of correct positive
    results divided by
177.        #the number of positive results that should have been returned.
    The F1 score can be
178.        #interpreted as a weighted average of the precision and recall,
    where an F1 score
179.        #reaches its best value at 1 and worst at 0.
180.        from sklearn.metrics import f1_score
```

```
181.
182.        def train_classifier(clf, X_train, y_train):
183.            ''' Fits a classifier to the training data. '''
184.
185.            # Start the clock, train the classifier, then stop the clock
186.            start = time()
187.            clf.fit(X_train, y_train)
188.            end = time()
189.
190.            # Print the results
191.            print "Trained model in {:.4f} seconds".format(end - start)
192.
193.
194.        def predict_labels(clf, features, target):
195.            ''' Makes predictions using a fit classifier based on F1 score.
    '''
196.
197.            # Start the clock, make predictions, then stop the clock
198.            start = time()
199.            y_pred = clf.predict(features)
200.
201.            end = time()
202.            # Print and return results
203.            print "Made predictions in {:.4f} seconds.".format(end -
    start)
204.
205.            return f1_score(target, y_pred, pos_label='H'), sum(target ==
    y_pred) / float(len(y_pred))
206.
207.
208.        def train_predict(clf, X_train, y_train, X_test, y_test):
209.            ''' Train and predict using a classifer based on F1 score. '''
210.
211.            # Indicate the classifier and the training set size
```

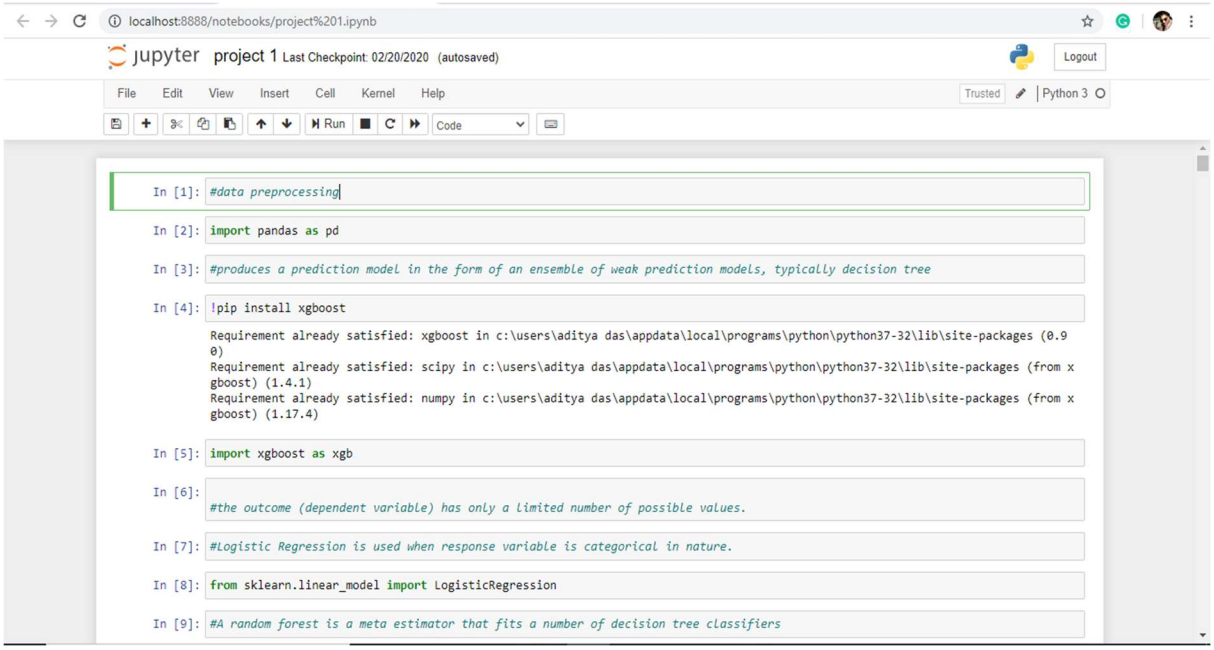
```
212.        print "Training a {} using a training set size of {}. . .".format(clf._class.name_, len(X_train))
213.
214.        # Train the classifier
215.        train_classifier(clf, X_train, y_train)
216.
217.        # Print the results of prediction for both training and testing
218.        f1, acc = predict_labels(clf, X_train, y_train)
219.        print f1, acc
220.        print "F1 score and accuracy score for training set: {:.4f} , {:.4f}.".format(f1 , acc)
221.
222.        f1, acc = predict_labels(clf, X_test, y_test)
223.        print "F1 score and accuracy score for test set: {:.4f} , {:.4f}.".format(f1 , acc)
224.
225.
226.
227.
228.
229.
230.
231.    In [10]:
232.    # Initialize the three models (XGBoost is initialized later)
233.    clf_A = LogisticRegression(random_state = 42)
234.    clf_B = SVC(random_state = 912, kernel='rbf')
235.    #Boosting refers to this general problem of producing a very accurate prediction rule
236.    #by combining rough and moderately inaccurate rules-of-thumb
237.    clf_C = xgb.XGBClassifier(seed = 82)
238.
239.    train_predict(clf_A, X_train, y_train, X_test, y_test)
240.    print "
```

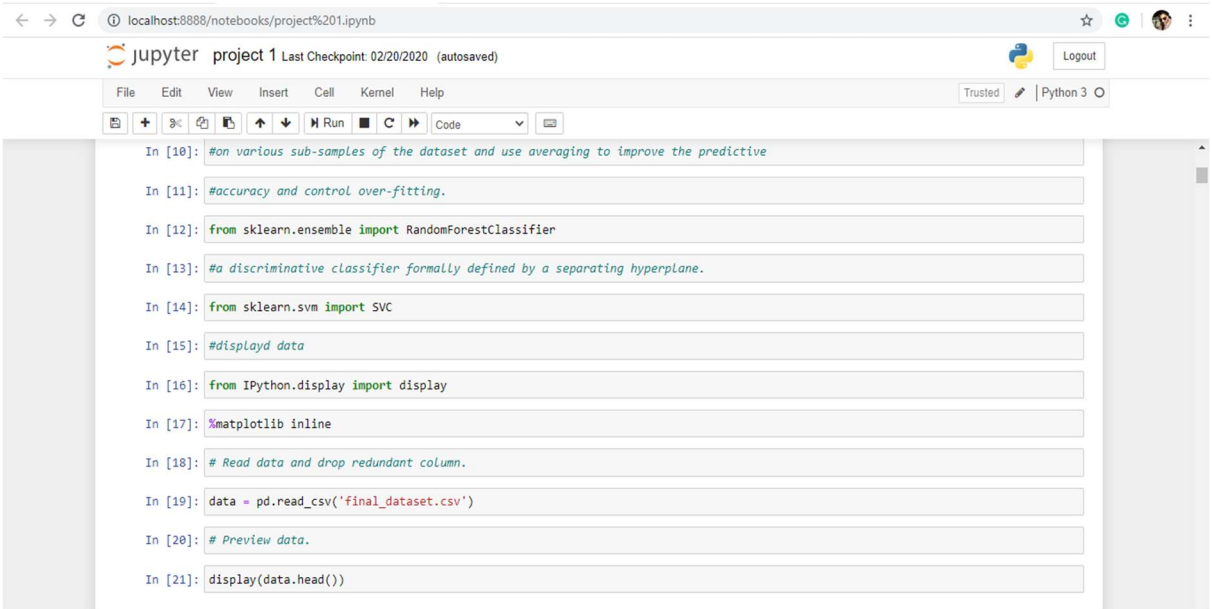
```
241.    train_predict(clf_B, X_train, y_train, X_test, y_test)
242.    print ''
243.    train_predict(clf_C, X_train, y_train, X_test, y_test)
244.    print ''
245.
246.
247.
248.
249.
250.
251.    In [39]:
252.    # TODO: Import 'GridSearchCV' and 'make_scorer'
253.    from sklearn.grid_search import GridSearchCV
254.    from sklearn.metrics import make_scorer
255.
256.
257.    # TODO: Create the parameters list you wish to tune
258.    parameters = { 'learning_rate' : [0.1],
259.                   'n_estimators' : [40],
260.                   'max_depth': [3],
261.                   'min_child_weight': [3],
262.                   'gamma':[0.4],
263.                   'subsample' : [0.8],
264.                   'colsample_bytree' : [0.8],
265.                   'scale_pos_weight' : [1],
266.                   'reg_alpha':[1e-5]
267.                  }
268.
269.    # TODO: Initialize the classifier
270.    clf = xgb.XGBClassifier(seed=2)
271.
272.    # TODO: Make an f1 scoring function using 'make_scorer'
273.    f1_scorer = make_scorer(f1_score,pos_label='H')
274.
```

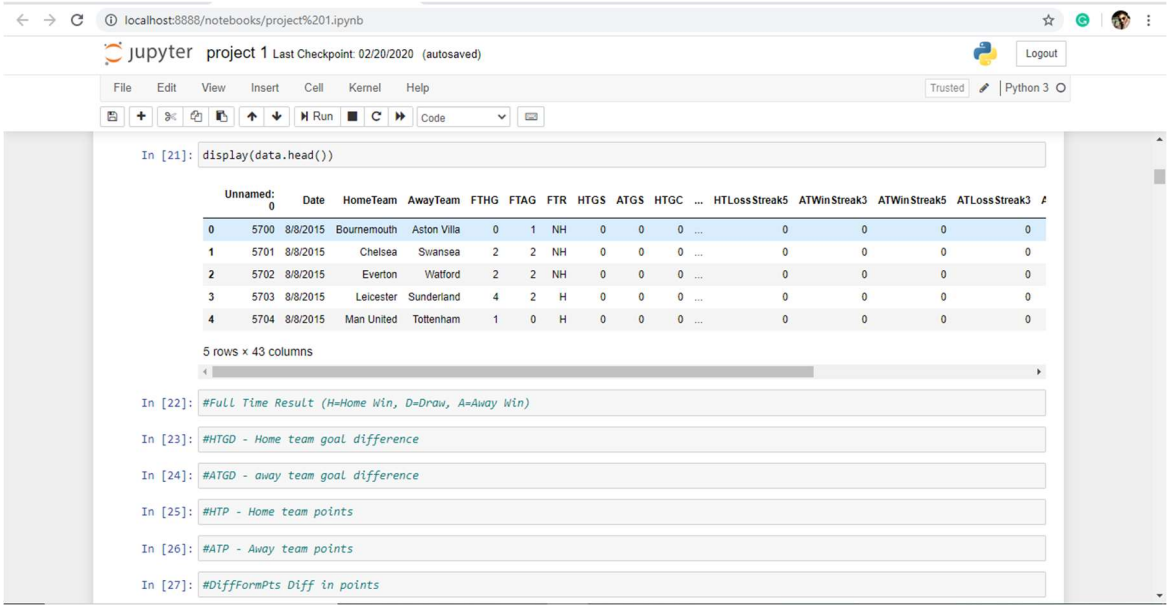
```
    # TODO: Perform grid search on the classifier using the f1_scorer as the scoring method
    grid_obj = GridSearchCV(clf,
                            scoring=f1_scorer,
                            param_grid=parameters,
                            cv=5)

    # TODO: Fit the grid search object to the training data and find the optimal parameters
    grid_obj = grid_obj.fit(X_train,y_train)

    # Get the estimator
    clf = grid_obj.best_estimator_
    print clf

    # Report the final F1 score for training and testing after parameter tuning
    f1, acc = predict_labels(clf, X_train, y_train)
    print "F1 score and accuracy score for training set: {:.4f} , {:.4f}.".format(f1 , acc)

    f1, acc = predict_labels(clf, X_test, y_test)
    print "F1 score and accuracy score for test set: {:.4f} , {:.4f}.".format(f1 , acc)



    In [ ]:
    #prediction
```

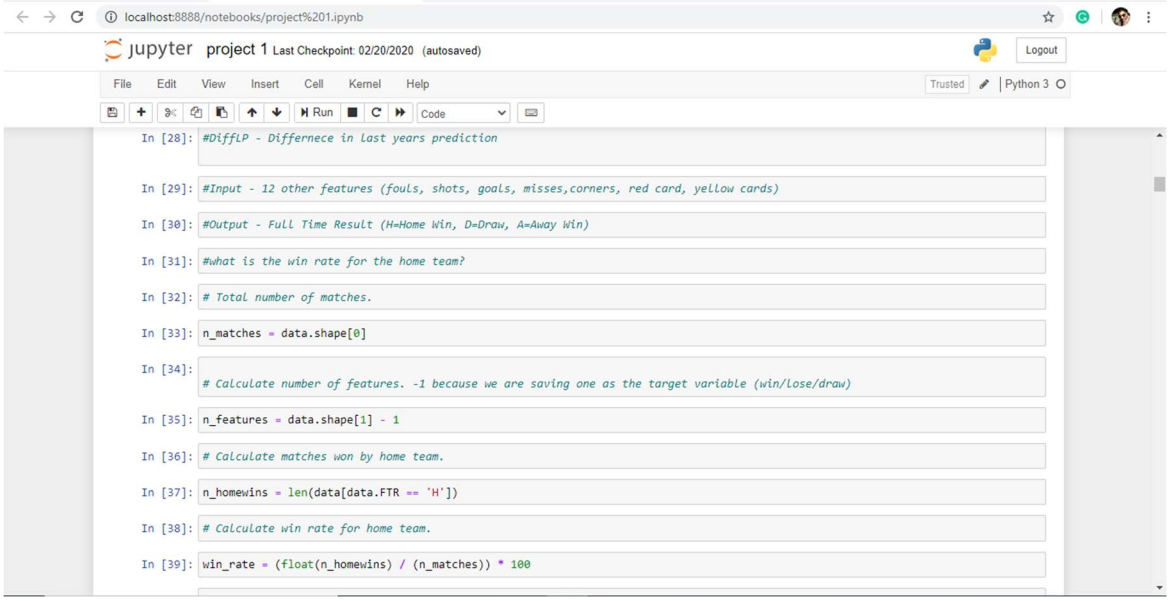
3.0 RESULT SECTION

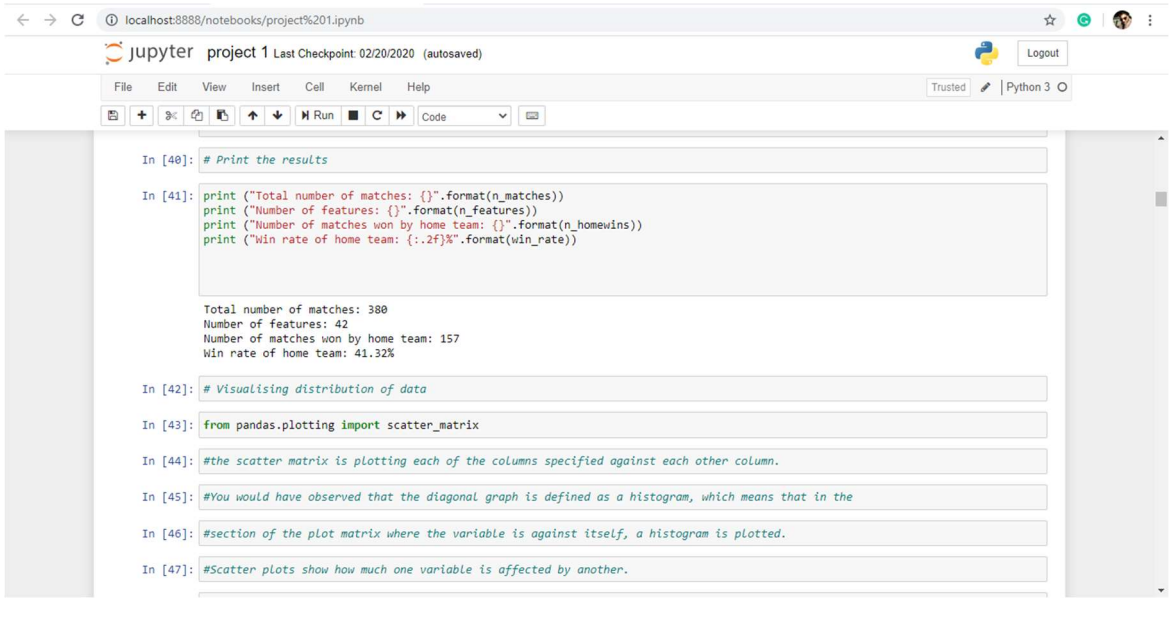
- ## 3.1 SNAPSORT OF TRANNING AND ACCURACY LEVEL

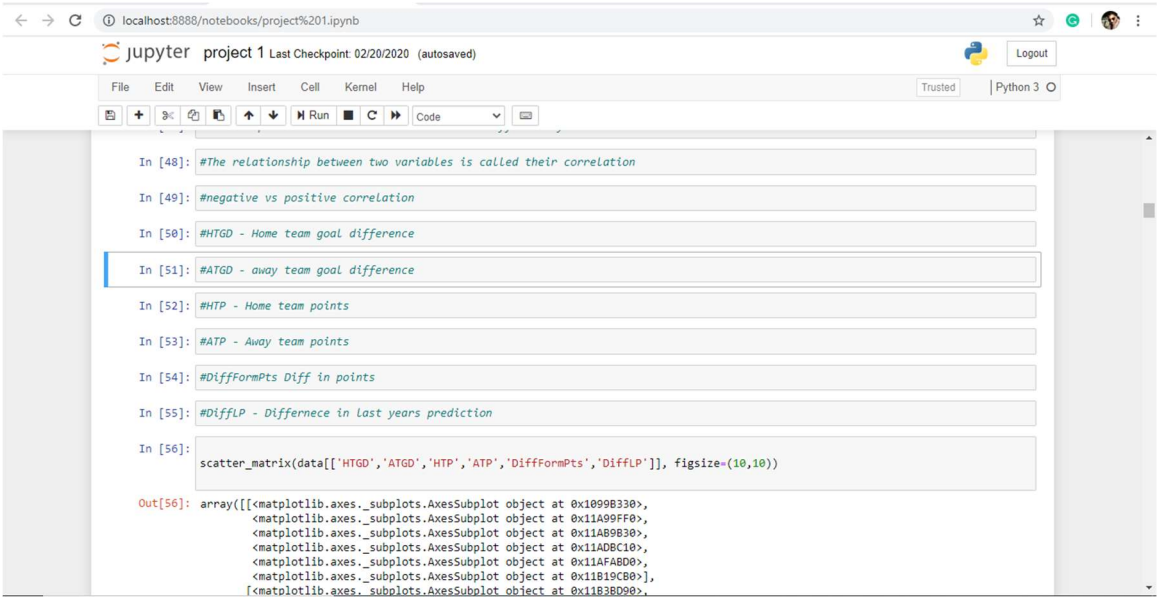


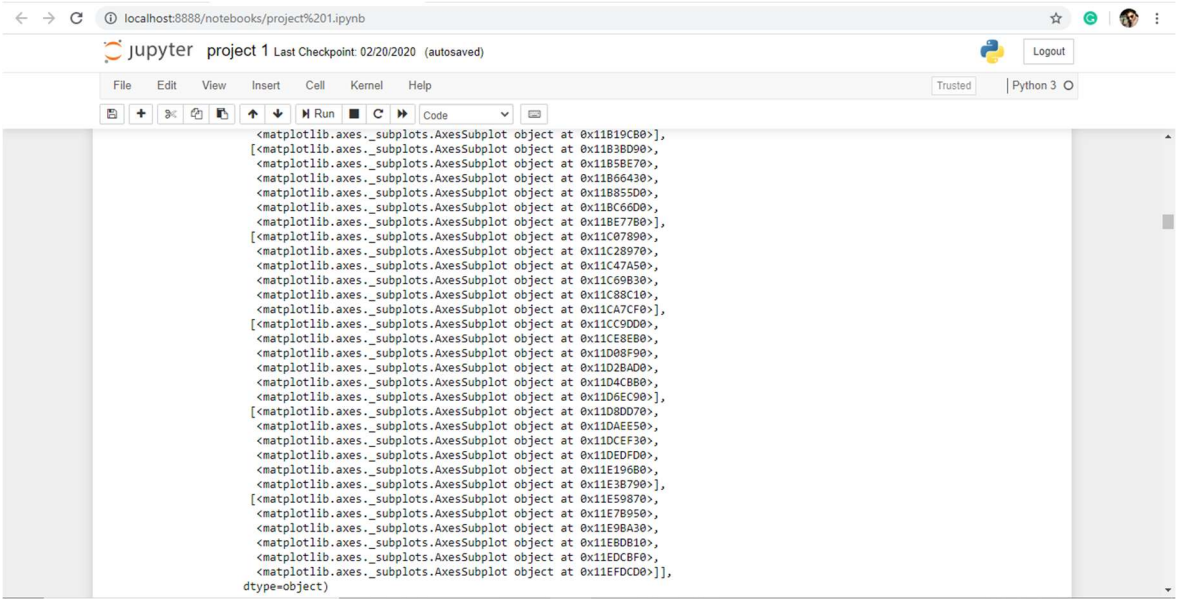


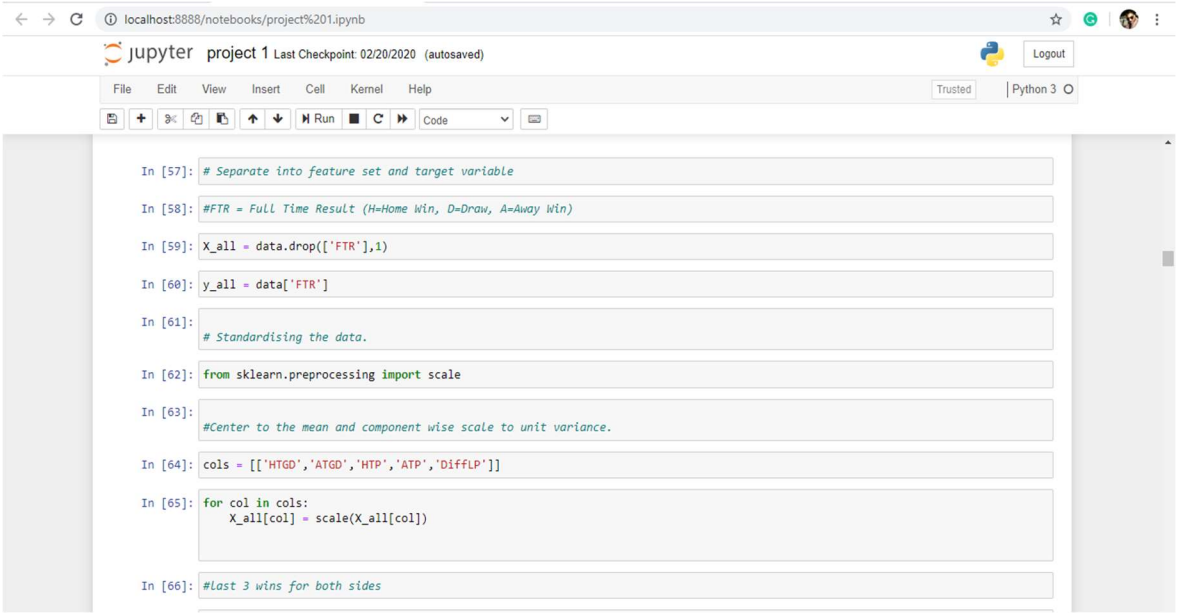

localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted Python 3
Run Code
In [21]: display(data.head())
Unnamed: 0 Date HomeTeam AwayTeam FTHG FTAG FTR HTGS ATGS HTGC ... HTLossStreak5 ATWinStreak3 ATWinStreak5 ATLossStreak3
0 5700 8/8/2015 Bournemouth Aston Villa 0 1 NH 0 0 0 ... 0 0 0 0
1 5701 8/8/2015 Chelsea Swansea 2 2 NH 0 0 0 ... 0 0 0 0
2 5702 8/8/2015 Everton Watford 2 2 NH 0 0 0 ... 0 0 0 0
3 5703 8/8/2015 Leicester Sunderland 4 2 H 0 0 0 ... 0 0 0 0
4 5704 8/8/2015 Man United Tottenham 1 0 H 0 0 0 ... 0 0 0 0
5 rows × 43 columns
In [22]: #Full Time Result (H=Home Win, D=Draw, A=Away Win)
In [23]: #HTGD - Home team goal difference
In [24]: #ATGD - away team goal difference
In [25]: #HTP - Home team points
In [26]: #ATP - Away team points
In [27]: #DiffFormPts Diff in points


localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted Python 3
Run Code
In [28]: #DiffLP - Differnece in last years prediction
In [29]: #Input - 12 other features (fouls, shots, goals, misses,corners, red card, yellow cards)
In [30]: #Output - Full Time Result (H=Home Win, D=Draw, A=Away Win)
In [31]: #what is the win rate for the home team?
In [32]: # Total number of matches.
In [33]: n_matches = data.shape[0]
In [34]: # Calculate number of features. -1 because we are saving one as the target variable (win/lose/draw)
In [35]: n_features = data.shape[1] - 1
In [36]: # Calculate matches won by home team.
In [37]: n_homewins = len(data[data.FTR == 'H'])
In [38]: # Calculate win rate for home team.
In [39]: win_rate = (float(n_homewins) / (n_matches)) * 100


localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted Python 3
Run Code
In [40]: # Print the results
In [41]: print ("Total number of matches: {}".format(n_matches))
print ("Number of features: {}".format(n_features))
print ("Number of matches won by home team: {}".format(n_homewins))
print ("Win rate of home team: {:.2f}%".format(win_rate))
Total number of matches: 380
Number of features: 42
Number of matches won by home team: 157
Win rate of home team: 41.32%
In [42]: # Visualising distribution of data
In [43]: from pandas.plotting import scatter_matrix
In [44]: #the scatter matrix is plotting each of the columns specified against each other column.
In [45]: #You would have observed that the diagonal graph is defined as a histogram, which means that in the
In [46]: #section of the plot matrix where the variable is against itself, a histogram is plotted.
In [47]: #Scatter plots show how much one variable is affected by another.


localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
In [48]: #The relationship between two variables is called their correlation
In [49]: #negative vs positive correlation
In [50]: #HTGD - Home team goal difference
In [51]: #ATGD - away team goal difference
In [52]: #HTP - Home team points
In [53]: #ATP - Away team points
In [54]: #DiffFormPts Diff in points
In [55]: #DiffLP - Differnece in last years prediction
In [56]: scatter_matrix(data[['HTGD','ATGD','HTP','ATP','DiffFormPts','DiffLP']], figsize=(10,10))
Out[56]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0x1099B330>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11A99FF0>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11AB9B30>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11ADBC10>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11AFABD0>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11B19CB0>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x11B3BD90>,


localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
<matplotlib.axes._subplots.AxesSubplot object at 0x11B19CB0>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x11B3BD90>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11B5BE70>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11B66430>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11B855D0>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11BC66D0>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11BE77B0>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x11C07890>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11C28970>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11C47A50>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11C69B30>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11C88C10>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11CA7CF0>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x11CC9DD0>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11CE8EB0>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11D08F90>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11D2BAD0>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11D4CBB0>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11D6EC90>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x11D8DD70>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11DAEE50>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11DCEF30>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11DEDFD0>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11E196B0>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11E3B790>],
[<matplotlib.axes._subplots.AxesSubplot object at 0x11E59870>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11E7B950>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11E9BA30>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11EBDB10>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11EDCBF0>,
<matplotlib.axes._subplots.AxesSubplot object at 0x11EFDCD0>]],
dtype=object)


localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
In [57]: # Separate into feature set and target variable
In [58]: #FTR = Full Time Result (H=Home Win, D=Draw, A=Away Win)
In [59]: X_all = data.drop(['FTR'],1)
In [60]: y_all = data['FTR']
In [61]: # Standardising the data.
In [62]: from sklearn.preprocessing import scale
In [63]: #Center to the mean and component wise scale to unit variance.
In [64]: cols = [['HTGD','ATGD','HTP','ATP','DiffLP']]
In [65]: for col in cols:
X_all[col] = scale(X_all[col])
In [66]: #Last 3 wins for both sides

localhost:8888/notebooks/project%201.ipynb

jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Run Code

In [67]:
```
X_all.HM1 = X_all.HM1.astype('str')
```

In [68]:
```
X_all.HM2 = X_all.HM2.astype('str')
```

In [69]:
```
X_all.HM3 = X_all.HM3.astype('str')
```

In [70]:
```
X_all.AM1 = X_all.AM1.astype('str')
```

In [71]:
```
X_all.AM2 = X_all.AM2.astype('str')
```

In [72]:
```
X_all.AM3 = X_all.AM3.astype('str')
```

In [73]:
```
#we want continous vars that are integers for our input data, so lets remove any categorical vars
def preprocess_features(X):
    ''' Preprocesses the football data and converts catagorical variables into dummy variables. '''

    # Initialize new output DataFrame
    output = pd.DataFrame(index = X.index)

    # Investigate each feature column for the data
    for col, col_data in X.iteritems():

        # If data type is categorical, convert to dummy variables
        if col_data.dtype == object:
            col_data = pd.get_dummies(col_data, prefix = col)

        # Collect the revised columns
        output = output.join(col_data)
```

localhost:8888/notebooks/project%201.ipynb

jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Run Code

In [73]:
```
#we want continous vars that are integers for our input data, so lets remove any categorical vars
def preprocess_features(X):
    ''' Preprocesses the football data and converts catagorical variables into dummy variables. '''

    # Initialize new output DataFrame
    output = pd.DataFrame(index = X.index)

    # Investigate each feature column for the data
    for col, col_data in X.iteritems():

        # If data type is categorical, convert to dummy variables
        if col_data.dtype == object:
            col_data = pd.get_dummies(col_data, prefix = col)

        # Collect the revised columns
        output = output.join(col_data)

    return output
```

In [74]:
```
X_all = preprocess_features(X_all)
print ("Processed feature columns ({} total features):\n{}".format(len(X_all.columns), list(X_all.columns)))
```

```
Processed feature columns (606 total features):
['Unnamed: 0', 'Date_1/11/2015', 'Date_1/13/2016', 'Date_1/16/2016', 'Date_1/17/2016', 'Date_1/18/2016', 'Date_1/23/2016', 'Dat
e_1/24/2016', 'Date_1/3/2016', 'Date_1/5/2016', 'Date_10/17/2015', 'Date_10/18/2015', 'Date_10/19/2015', 'Date_10/24/2015', 'Da
te_10/25/2015', 'Date_10/31/2015', 'Date_10/4/2016', 'Date_10/5/2016', 'Date_10/8/2015', 'Date_11/21/2015', 'Date_11/22/2015',
'Date_11/23/2015', 'Date_11/28/2015', 'Date_11/29/2015', 'Date_11/5/2016', 'Date_12/1/2016', 'Date_12/12/2015', 'Date_12/13/201
5', 'Date_12/14/2015', 'Date_12/19/2015', 'Date_12/20/2015', 'Date_12/21/2015', 'Date_12/26/2015', 'Date_12/28/2015', 'Date_12/
```

localhost:8888/notebooks/project%201.ipynb

jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Run Code

In [75]:
```
# Show the feature information by printing the first five rows
print ("\nFeature values:")
display(X_all.head())
```

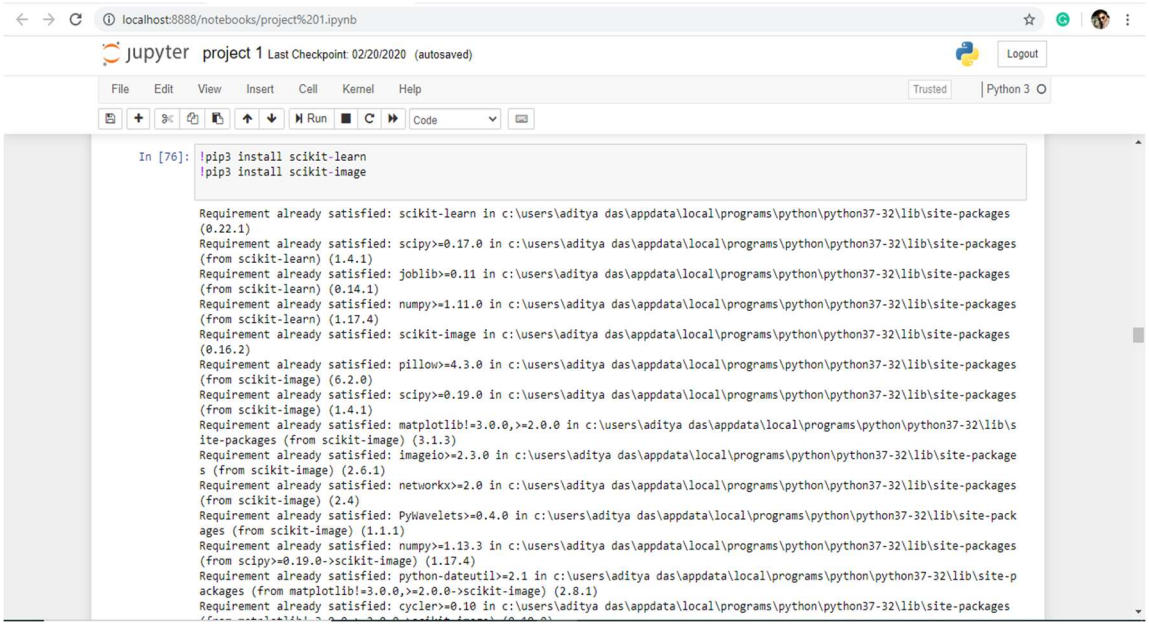
Feature values:

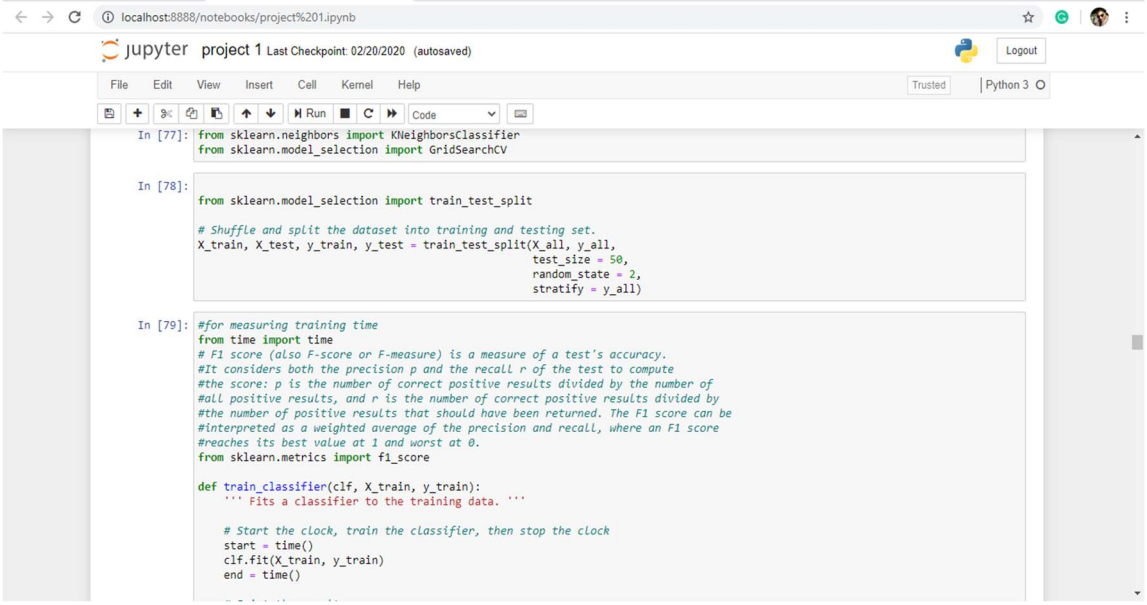
	Unnamed: 0	Date_1/11/2015	Date_1/13/2016	Date_1/16/2016	Date_1/17/2016	Date_1/18/2016	Date_1/23/2016	Date_1/24/2016	Date_1/3/2016	Date_1/5/2016	...
0	5700	0	0	0	0	0	0	0	0	0	...
1	5701	0	0	0	0	0	0	0	0	0	...
2	5702	0	0	0	0	0	0	0	0	0	...
3	5703	0	0	0	0	0	0	0	0	0	...
4	5704	0	0	0	0	0	0	0	0	0	...

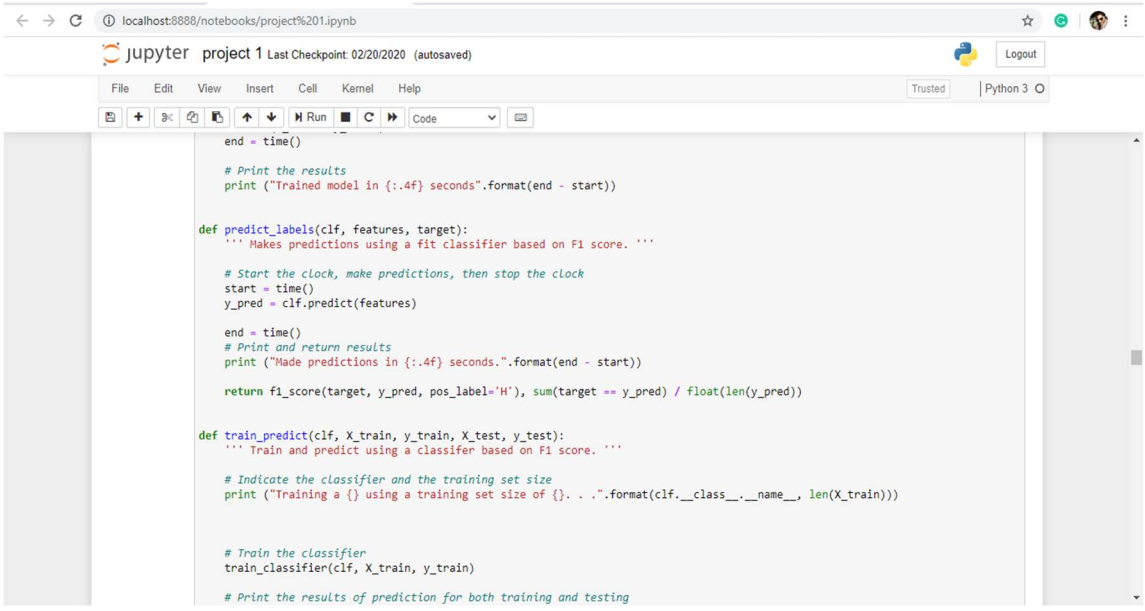
5 rows × 606 columns

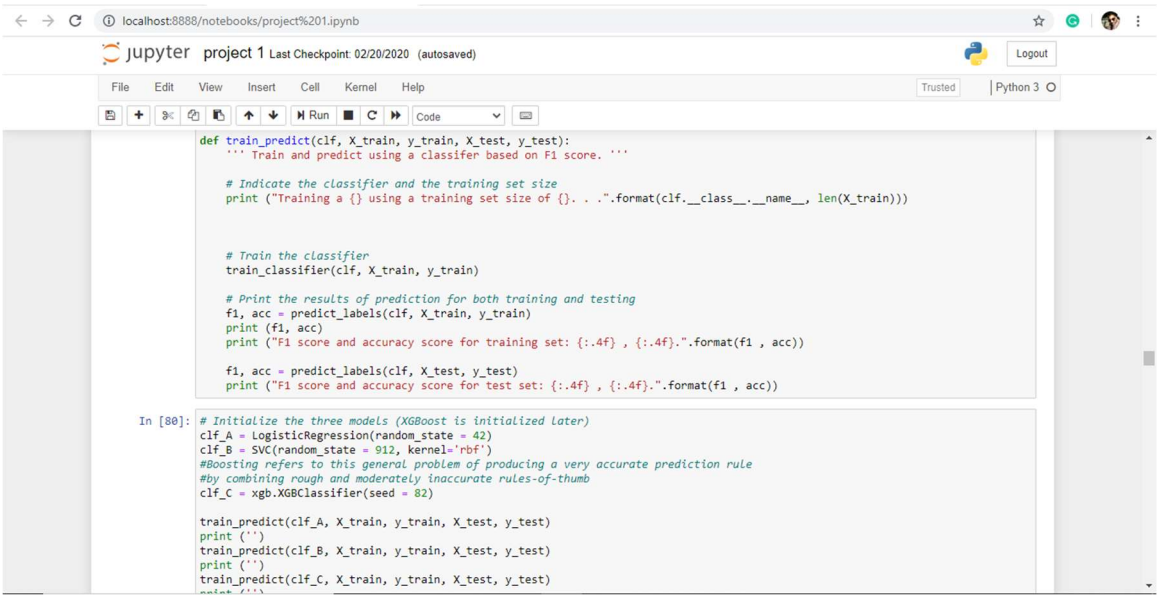
In [76]:
```
!pip3 install scikit-learn
!pip3 install scikit-image
```

```
Requirement already satisfied: scikit-learn in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages
(0.22.1)
Requirement already satisfied: scipy>=0.17.0 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages
(from scikit-learn) (1.4.1)
```


localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
In [76]:
!pip3 install scikit-learn
!pip3 install scikit-image
Requirement already satisfied: scikit-learn in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (0.22.1)
Requirement already satisfied: scipy>=0.17.0 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from scikit-learn) (1.4.1)
Requirement already satisfied: joblib>=0.11 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from scikit-learn) (0.14.1)
Requirement already satisfied: numpy>=1.11.0 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from scikit-learn) (1.17.4)
Requirement already satisfied: scikit-image in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (0.16.2)
Requirement already satisfied: pillow>=4.3.0 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from scikit-image) (6.2.0)
Requirement already satisfied: scipy>=0.19.0 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from scikit-image) (1.4.1)
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from scikit-image) (3.1.3)
Requirement already satisfied: imageio>=2.3.0 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from scikit-image) (2.6.1)
Requirement already satisfied: networkx>=2.0 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from scikit-image) (2.4)
Requirement already satisfied: PyWavelets>=0.4.0 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from scikit-image) (1.1.1)
Requirement already satisfied: numpy>=1.13.3 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from scipy>=0.19.0->scikit-image) (1.17.4)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from matplotlib!=3.0.0,>=2.0.0->scikit-image) (2.8.1)
Requirement already satisfied: cycler>=0.10 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages


localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
In [77]:
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
In [78]:
from sklearn.model_selection import train_test_split
# Shuffle and split the dataset into training and testing set.
X_train, X_test, y_train, y_test = train_test_split(X_all, y_all,
test_size = 50,
random_state = 2,
stratify = y_all)
In [79]:
#for measuring training time
from time import time
# F1 score (also F-score or F-measure) is a measure of a test's accuracy.
#It considers both the precision p and the recall r of the test to compute
#the score: p is the number of correct positive results divided by the number of
#all positive results, and r is the number of correct positive results divided by
#the number of positive results that should have been returned. The F1 score can be
#interpreted as a weighted average of the precision and recall, where an F1 score
#reaches its best value at 1 and worst at 0.
from sklearn.metrics import f1_score
def train_classifier(clf, X_train, y_train):
''' Fits a classifier to the training data. '''
# Start the clock, train the classifier, then stop the clock
start = time()
clf.fit(X_train, y_train)
end = time()


localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
end = time()
# Print the results
print ("Trained model in {:.4f} seconds".format(end - start))
def predict_labels(clf, features, target):
''' Makes predictions using a fit classifier based on F1 score. '''
# Start the clock, make predictions, then stop the clock
start = time()
y_pred = clf.predict(features)
end = time()
# Print and return results
print ("Made predictions in {:.4f} seconds.".format(end - start))
return f1_score(target, y_pred, pos_label='H'), sum(target == y_pred) / float(len(y_pred))
def train_predict(clf, X_train, y_train, X_test, y_test):
''' Train and predict using a classifer based on F1 score. '''
Indicate the classifier and the training set size
print ("Training a {} using a training set size of {}. . .".format(clf.__class__.__name__, len(X_train)))
# Train the classifier
train_classifier(clf, X_train, y_train)
# Print the results of prediction for both training and testing



localhost:8888/notebooks/project%201.ipynb

jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)

Logout

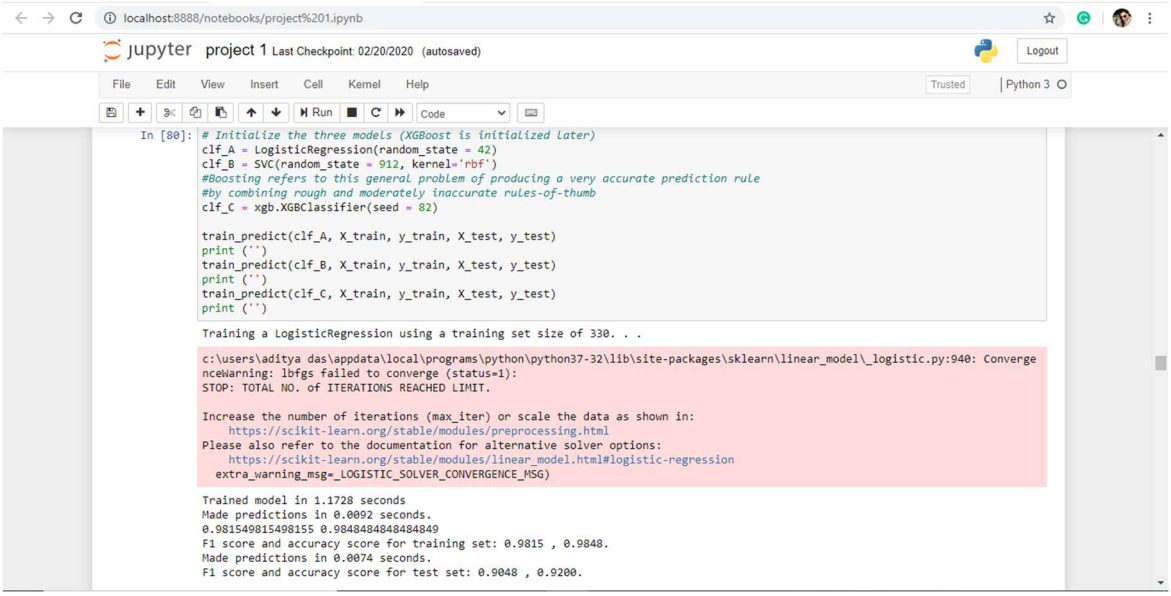
File Edit View Insert Cell Kernel Help Trusted Python 3

Run Code

```
def train_predict(clf, X_train, y_train, X_test, y_test):
    ''' Train and predict using a classifer based on F1 score. '''

    # Indicate the classifier and the training set size
    print ("Training a {} using a training set size of {}. . .".format(clf.__class__.__name__, len(X_train)))



    # Train the classifier
    train_classifier(clf, X_train, y_train)

    # Print the results of prediction for both training and testing
    f1, acc = predict_labels(clf, X_train, y_train)
    print (f1, acc)
    print ("F1 score and accuracy score for training set: {:.4f} , {:.4f}.".format(f1 , acc))

    f1, acc = predict_labels(clf, X_test, y_test)
    print ("F1 score and accuracy score for test set: {:.4f} , {:.4f}.".format(f1 , acc))
```

In [80]:

```
# Initialize the three models (XGBoost is initialized later)
clf_A = LogisticRegression(random_state = 42)
clf_B = SVC(random_state = 912, kernel='rbf')
#Boosting refers to this general problem of producing a very accurate prediction rule
#by combining rough and moderately inaccurate rules-of-thumb
clf_C = xgb.XGBClassifier(seed = 82)

train_predict(clf_A, X_train, y_train, X_test, y_test)
print ('')
train_predict(clf_B, X_train, y_train, X_test, y_test)
print ('')
train_predict(clf_C, X_train, y_train, X_test, y_test)
```



localhost:8888/notebooks/project%201.ipynb

jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)

Logout

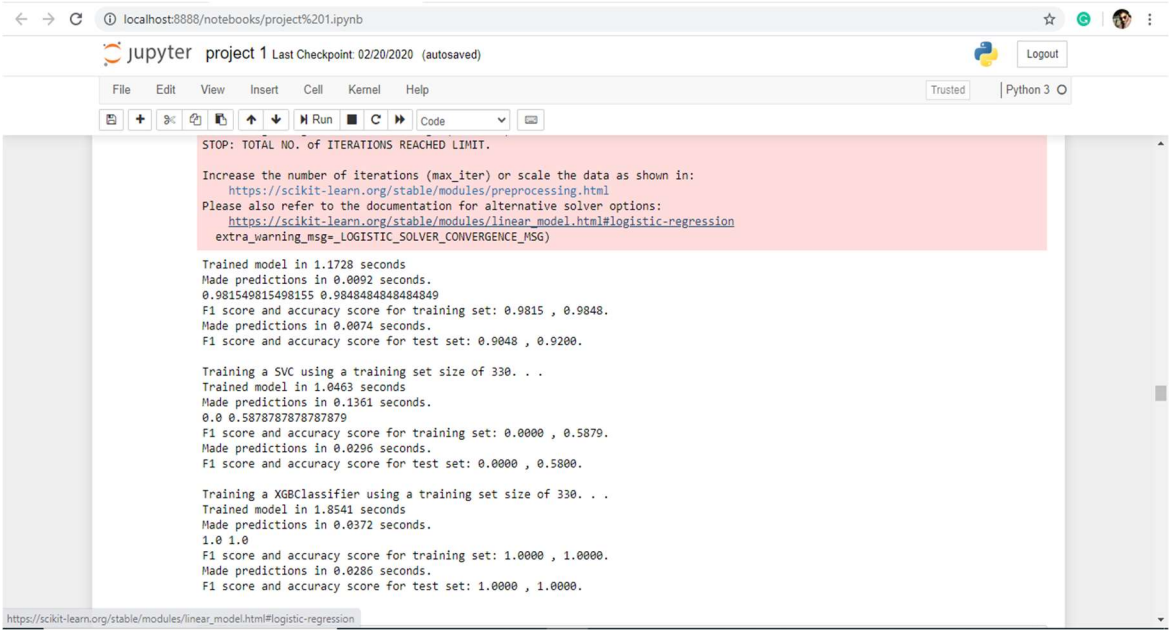
File Edit View Insert Cell Kernel Help Trusted Python 3

Run Code

In [80]:

```
# Initialize the three models (XGBoost is initialized later)
clf_A = LogisticRegression(random_state = 42)
clf_B = SVC(random_state = 912, kernel='rbf')
#Boosting refers to this general problem of producing a very accurate prediction rule
#by combining rough and moderately inaccurate rules-of-thumb
clf_C = xgb.XGBClassifier(seed = 82)

train_predict(clf_A, X_train, y_train, X_test, y_test)
print ('')
train_predict(clf_B, X_train, y_train, X_test, y_test)
print ('')
train_predict(clf_C, X_train, y_train, X_test, y_test)
print ('')
```

```
Training a LogisticRegression using a training set size of 330. . .
c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages\sklearn\linear_model\_logistic.py:940: Converge
nceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
Trained model in 1.1728 seconds
Made predictions in 0.0092 seconds.
0.981549815498155 0.9848484848484849
F1 score and accuracy score for training set: 0.9815 , 0.9848.
Made predictions in 0.0074 seconds.
F1 score and accuracy score for test set: 0.9048 , 0.9200.
```



localhost:8888/notebooks/project%201.ipynb

jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)

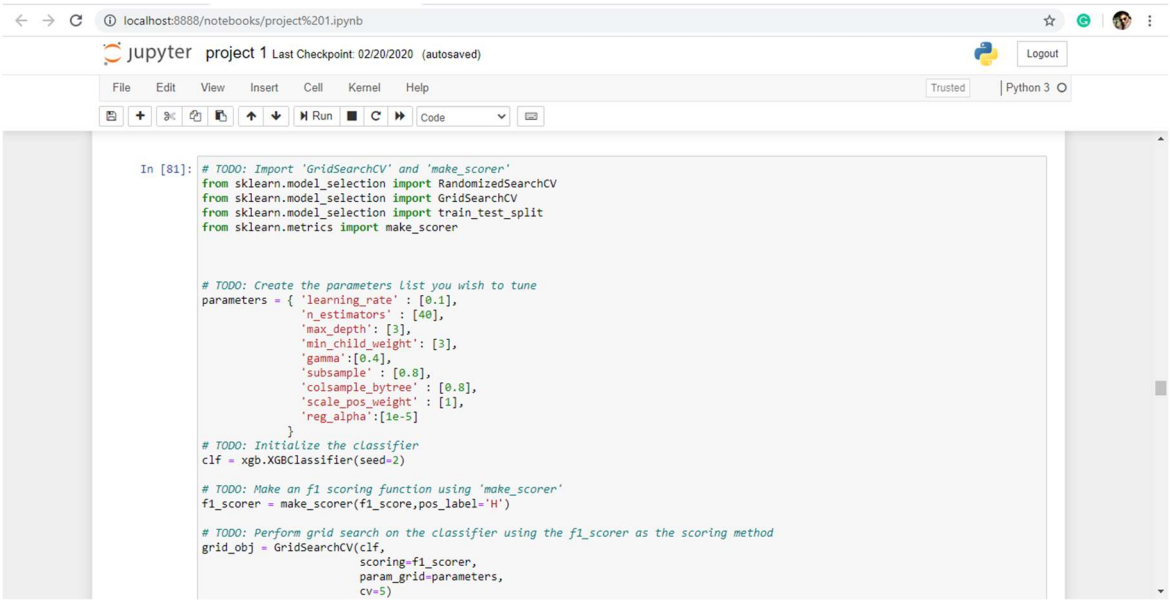
Logout

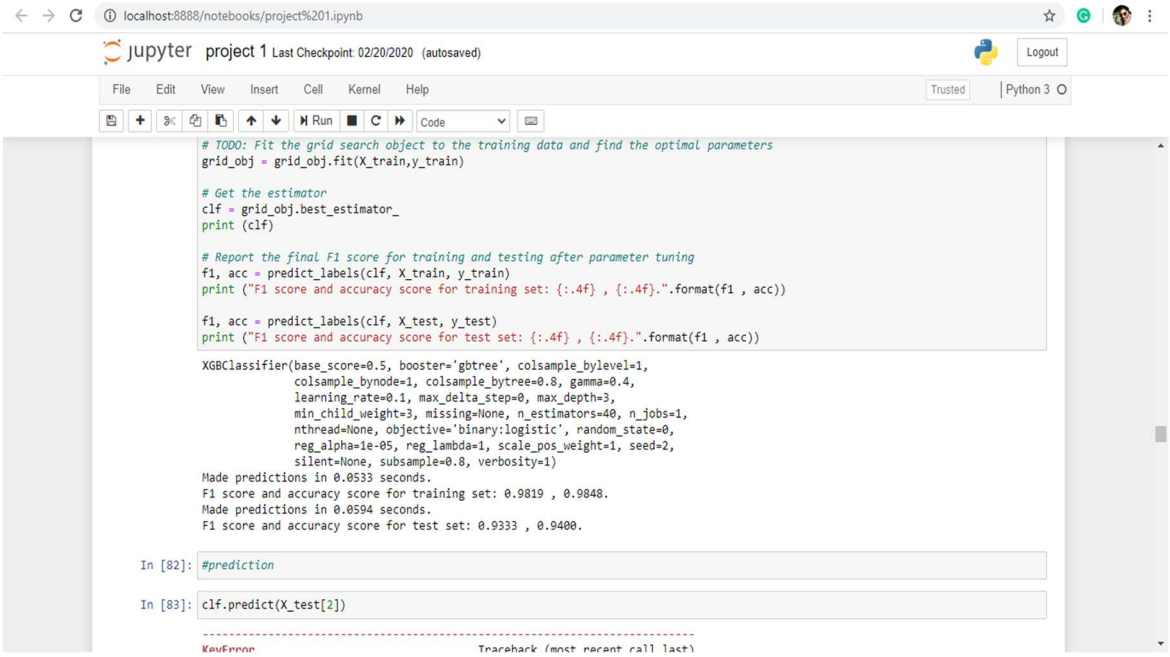
File Edit View Insert Cell Kernel Help Trusted Python 3

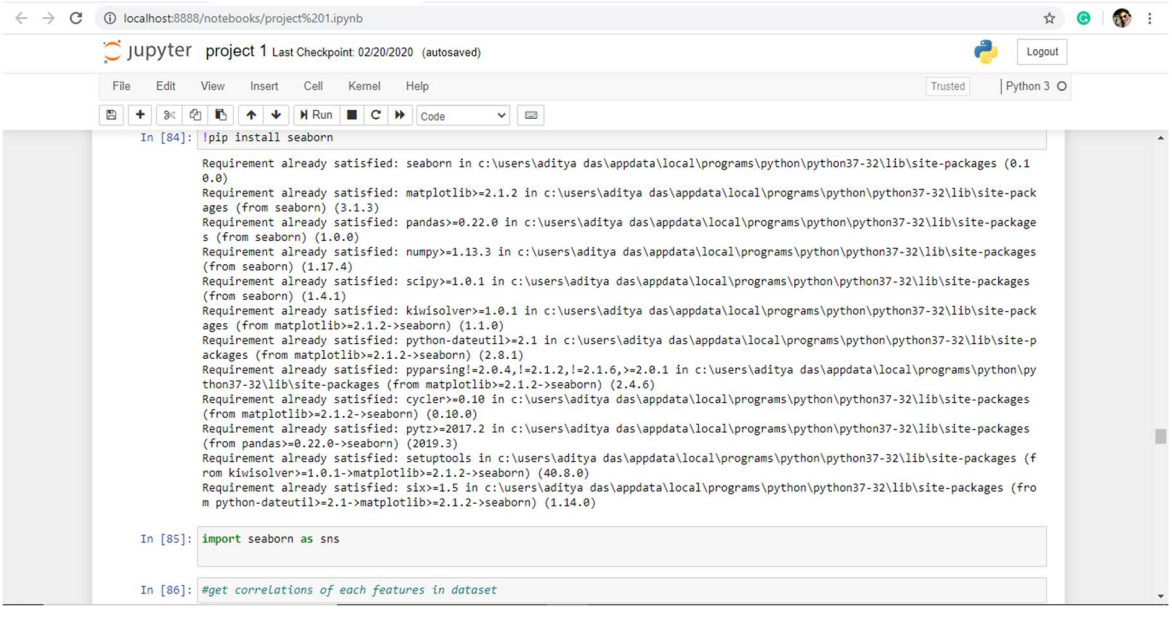
Run Code

```
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
Trained model in 1.1728 seconds
Made predictions in 0.0092 seconds.
0.981549815498155 0.9848484848484849
F1 score and accuracy score for training set: 0.9815 , 0.9848.
Made predictions in 0.0074 seconds.
F1 score and accuracy score for test set: 0.9048 , 0.9200.

Training a SVC using a training set size of 330. . .
Trained model in 1.0463 seconds
Made predictions in 0.1361 seconds.
0.0 0.5878787878787879
F1 score and accuracy score for training set: 0.0000 , 0.5879.
Made predictions in 0.0296 seconds.
F1 score and accuracy score for test set: 0.0000 , 0.5800.

Training a XGBClassifier using a training set size of 330. . .
Trained model in 1.8541 seconds
Made predictions in 0.0372 seconds.
1.0 1.0
F1 score and accuracy score for training set: 1.0000 , 1.0000.
Made predictions in 0.0286 seconds.
F1 score and accuracy score for test set: 1.0000 , 1.0000.
```

https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression


localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
In [81]:
# TODO: Import 'GridSearchCV' and 'make_scorer'
from sklearn.model_selection import RandomizedSearchCV
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import make_scorer
# TODO: Create the parameters list you wish to tune
parameters = { 'learning_rate' : [0.1],
'n_estimators' : [40],
'max_depth': [3],
'min_child_weight': [3],
'gamma':[0.4],
'subsample' : [0.8],
'colsample_bytree' : [0.8],
'scale_pos_weight' : [1],
'reg_alpha':[1e-5]
}
# TODO: Initialize the classifier
clf = xgb.XGBClassifier(seed=2)
# TODO: Make an f1 scoring function using 'make_scorer'
f1_scorer = make_scorer(f1_score,pos_label='H')
# TODO: Perform grid search on the classifier using the f1_scorer as the scoring method
grid_obj = GridSearchCV(clf,
scoring=f1_scorer,
param_grid=parameters,
cv=5)


localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
# TODO: Fit the grid search object to the training data and find the optimal parameters
grid_obj = grid_obj.fit(X_train,y_train)
# Get the estimator
clf = grid_obj.best_estimator_
print (clf)
Report the final F1 score for training and testing after parameter tuning
f1, acc = predict_labels(clf, X_train, y_train)
print ("F1 score and accuracy score for training set: {:.4f} , {:.4f}.".format(f1 , acc))
f1, acc = predict_labels(clf, X_test, y_test)
print ("F1 score and accuracy score for test set: {:.4f} , {:.4f}.".format(f1 , acc))
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
colsample_bynode=1, colsample_bytree=0.8, gamma=0.4,
learning_rate=0.1, max_delta_step=0, max_depth=3,
min_child_weight=3, missing=None, n_estimators=40, n_jobs=1,
nthread=None, objective='binary:logistic', random_state=0,
reg_alpha=1e-05, reg_lambda=1, scale_pos_weight=1, seed=2,
silent=None, subsample=0.8, verbosity=1)
Made predictions in 0.0533 seconds.
F1 score and accuracy score for training set: 0.9819 , 0.9848.
Made predictions in 0.0594 seconds.
F1 score and accuracy score for test set: 0.9333 , 0.9400.
In [82]:
#prediction
In [83]:
clf.predict(X_test[2])
KeyError Traceback (most recent call last)


localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
In [84]:
!pip install seaborn
Requirement already satisfied: seaborn in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (0.10.0)
Requirement already satisfied: matplotlib>=2.1.2 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from seaborn) (3.1.3)
Requirement already satisfied: pandas>=0.22.0 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from seaborn) (1.0.0)
Requirement already satisfied: numpy>=1.13.3 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from seaborn) (1.17.4)
Requirement already satisfied: scipy>=1.0.1 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from seaborn) (1.4.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from matplotlib>=2.1.2->seaborn) (1.1.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from matplotlib>=2.1.2->seaborn) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from matplotlib>=2.1.2->seaborn) (2.4.6)
Requirement already satisfied: cycler>=0.10 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from matplotlib>=2.1.2->seaborn) (0.10.0)
Requirement already satisfied: pytz>=2017.2 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from pandas>=0.22.0->seaborn) (2019.3)
Requirement already satisfied: setuptools in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from kiwisolver>=1.0.1->matplotlib>=2.1.2->seaborn) (40.8.0)
Requirement already satisfied: six>=1.5 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (from python-dateutil>=2.1->matplotlib>=2.1.2->seaborn) (1.14.0)
In [85]:
import seaborn as sns
In [86]:
#get correlations of each features in dataset

localhost:8888/notebooks/project%201.ipynb

jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Run Code

In [88]:
```
top_corr_features = corrmat.index
```

In [89]:
```
!pip install matplotlib
```

```
Requirement already satisfied: matplotlib in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages
(3.1.3)
Requirement already satisfied: numpy>=1.11 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages
(from matplotlib) (1.17.4)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in c:\users\aditya das\appdata\local\programs\python\py
thon37-32\lib\site-packages (from matplotlib) (2.4.6)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-p
ackages (from matplotlib) (2.8.1)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-pack
ages (from matplotlib) (1.1.0)
Requirement already satisfied: cycler>=0.10 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages
(from matplotlib) (0.10.0)
Requirement already satisfied: six>=1.5 in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (fro
m python-dateutil>=2.1->matplotlib) (1.14.0)
Requirement already satisfied: setuptools in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (f
rom kiwisolver>=1.0.1->matplotlib) (40.8.0)
```

In [90]:
```
import matplotlib.pyplot as plt
```

In [91]:
```
plt.figure(figsize=(20,20))
```

Out[91]:
```
<Figure size 1440x1440 with 0 Axes>
```

```
<Figure size 1440x1440 with 0 Axes>
```

In [92]:
```
fig, ax = plt.subplots(figsize=(10,10))
```

localhost:8888/notebooks/project%201.ipynb

jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Run Code

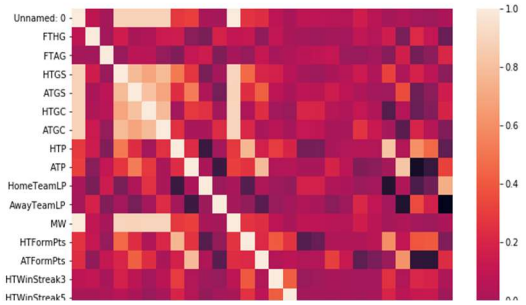
In [90]:
```
import matplotlib.pyplot as plt
```

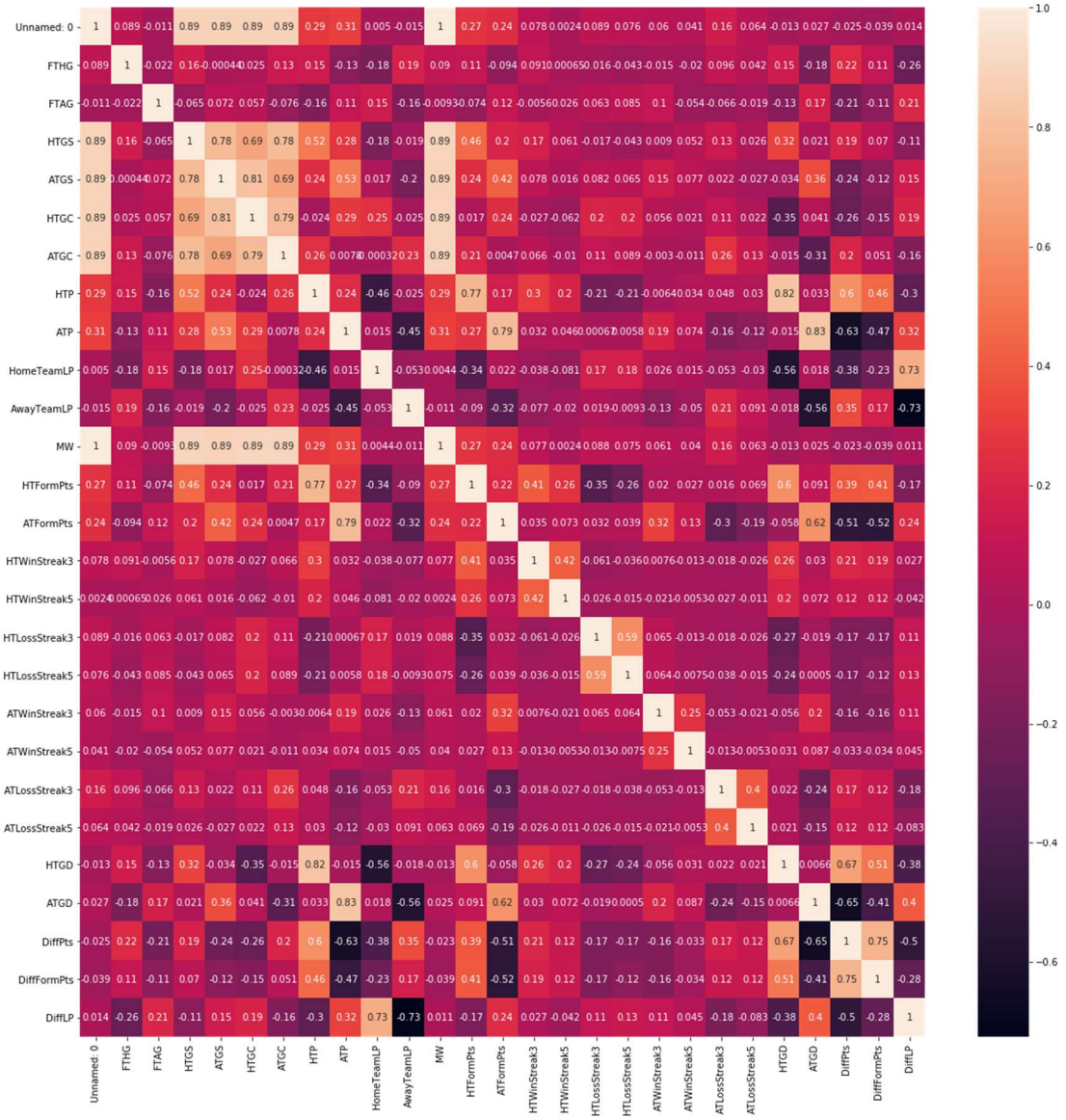
In [91]:
```
plt.figure(figsize=(20,20))
```

Out[91]:
```
<Figure size 1440x1440 with 0 Axes>
```

```
<Figure size 1440x1440 with 0 Axes>
```

In [92]:
```
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(data.corr(), ax=ax)
plt.show()
```




Unnamed: 0
FTHG
FTAG
HTGS
ATGS
HTGC
ATGC
HTP
ATP
HomeTeamLP
AwayTeamLP
MW
HTFormPts
ATFormPts
HTWinStreak3
HTWinStreak5
HTLossStreak3
HTLossStreak5
ATWinStreak3
ATWinStreak5
ATLossStreak3
ATLossStreak5
HTGD
ATGD
DiffPts
DiffFormPts
DiffLP
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
In [93]: #plot heat map
In [94]: fig, ax = plt.subplots(figsize=(20,20))
g=sns.heatmap(data[top_corr_features].corr(), ax=ax, annot=True,annot_kws={'size':10}, cmap="RdYlGn")
plt.show()

localhost:8888/notebooks/project%201.ipynb

jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Run Code

In [113]:
```
!pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\aditya das\appdata\local\programs\python\python37-32\lib\site-packages (1.17.
4)
```

In [114]:
```
import numpy as np
```

In [115]:
```
predictions = pd.DataFrame(np.column_stack([y_test.values, y_test_pred]), columns=['Actual', 'Prediction'], index=y_test.index)
predictions.head(10)
```

Out[115]:

	Actual	Prediction
188	NH	NH
90	H	H
264	H	H
108	H	H
107	H	H
243	H	H
19	H	H
78	NH	NH
262	H	H
292	H	H

localhost:8888/notebooks/project%201.ipynb

jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)

Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Run Code

In [118]:
```
# Check back
usecols = ['Date', 'MW', 'HomeTeam', 'AwayTeam']
data = pd.read_csv('final_dataset.csv', usecols=usecols)

mapped_predictions = data.join(predictions, how='right')
mapped_predictions.head(10)
```

Out[118]:

	Date	HomeTeam	AwayTeam	MW	Actual	Prediction
188	12/29/2015	Leicester	Man City	19	NH	NH
90	10/24/2015	Arsenal	Everton	10	H	H
264	2/27/2016	West Brom	Crystal Palace	27	H	H
108	1/11/2015	Southampton	Bournemouth	11	H	H
107	1/11/2015	Everton	Sunderland	11	H	H
243	6/2/2016	Newcastle	West Brom	25	H	H
19	8/17/2015	Liverpool	Bournemouth	2	H	H
78	4/10/2015	Everton	Liverpool	8	NH	NH
262	2/27/2016	Stoke	Aston Villa	27	H	H
292	3/14/2016	Leicester	Newcastle	30	H	H

In [134]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
import seaborn as sns
```

localhost:8888/notebooks/project%201.ipynb

jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)

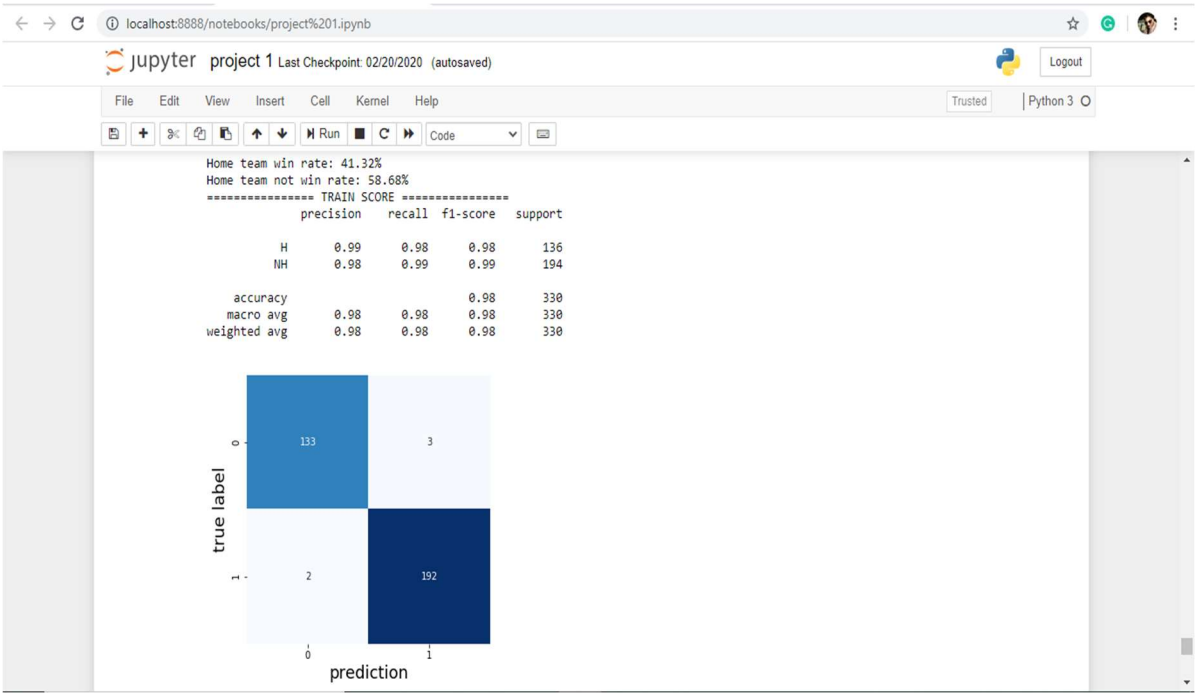
Logout

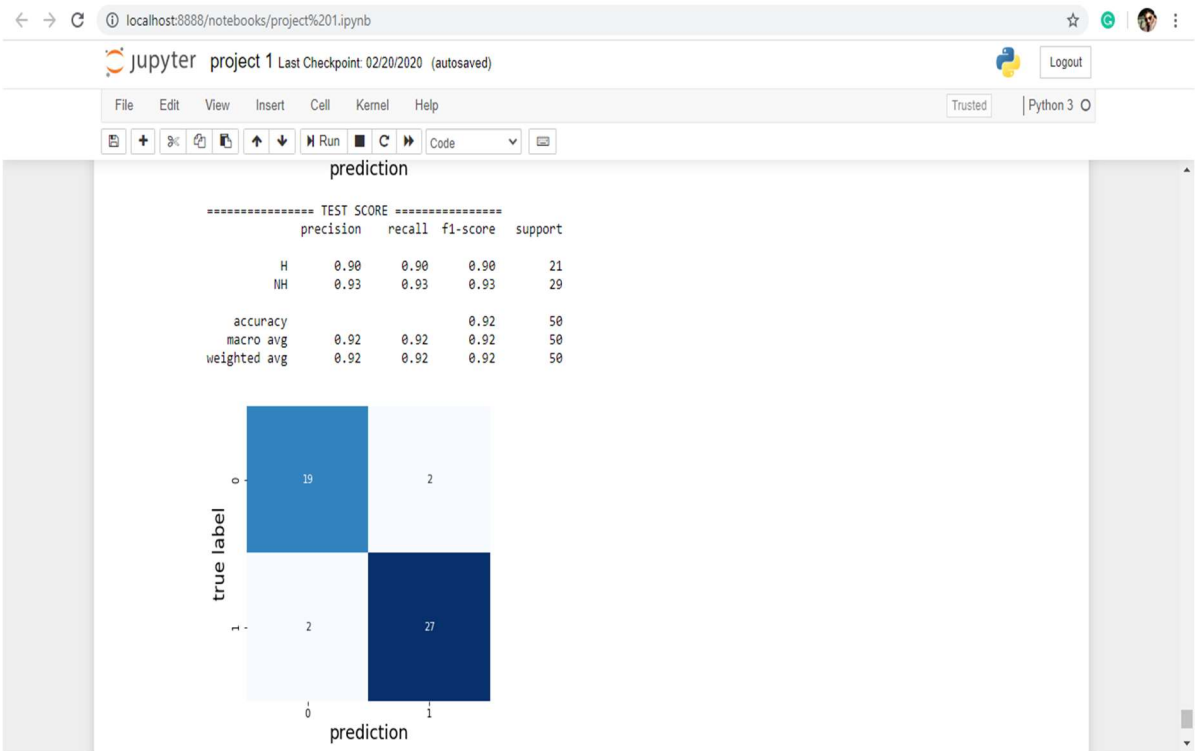
File Edit View Insert Cell Kernel Help Trusted Python 3

Run Code

In [135]:
```
print("Home team win rate: {:.2f}%".format(win_rate))
print("Home team not win rate: {:.2f}%".format(100-win_rate))

print('================ TRAIN SCORE ================')
confusion_train = pd.DataFrame(confusion_matrix(y_train, y_train_pred))

print(classification_report(y_train, y_train_pred))
plt.figure(figsize = (5,5))
sns.heatmap(confusion_train, annot=True, cmap=plt.cm.Blues, cbar=False, fmt='d')
plt.ylabel('true label', fontsize=18)
plt.xlabel('prediction', fontsize=18)
plt.show()

print('================ TEST SCORE ================')
confusion_test = pd.DataFrame(confusion_matrix(y_test, y_test_pred))

print(classification_report(y_test, y_test_pred))
plt.figure(figsize = (5,5))
sns.heatmap(confusion_test, annot=True, cmap=plt.cm.Blues, cbar=False, fmt='d')
plt.ylabel('true label', fontsize=18)
plt.xlabel('prediction', fontsize=18)
plt.show()
```

```
Home team win rate: 41.32%
Home team not win rate: 58.68%
================ TRAIN SCORE ================
              precision    recall  f1-score   support

           H       0.99      0.98      0.98       136
          NH       0.98      0.99      0.99       194
```


localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
Home team win rate: 41.32%
Home team not win rate: 58.68%
================ TRAIN SCORE ================
precision recall f1-score support
H 0.99 0.98 0.98 136
NH 0.98 0.99 0.99 194
accuracy 0.98 330
macro avg 0.98 0.98 0.98 330
weighted avg 0.98 0.98 0.98 330
133
3
2
192
true label
prediction


localhost:8888/notebooks/project%201.ipynb
jupyter project 1 Last Checkpoint: 02/20/2020 (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Trusted
Python 3
Run
Code
prediction
================ TEST SCORE ================
precision recall f1-score support
H 0.90 0.90 0.90 21
NH 0.93 0.93 0.93 29
accuracy 0.92 50
macro avg 0.92 0.92 0.92 50
weighted avg 0.92 0.92 0.92 50
19
2
2
27
true label
prediction

3.2 TESTING

■ UNIT TESTING

Unit is undertaken after coding of a module is complete, all syntax errors have been reomoved, and the code has been reviewed. This activity is typically undertaken by the coder of the module himself in the coding phase. Before carrying out unit testing, the unit test cases have to be designed and the test environment for the unit under test has to be developed. In this section, we first discuss the environment needed to perform unit testing.

Driver and stub modules

In order to test a single module, we need a complete environment to provide all relevant code that is necessary for execution of the module. That is, besides the module under ties the following are needed to test the module:

- The procedures belonging to other modules that the module under test calls.
- Non-local data structures that the module accesses.
- A procedure to call the functions of the module under test with appropriate parameters.

Modules required to provide the necessary environment (which either call, provide the required global data, or are called by the module under test) are usually not available unit

They too have been unit tested. In this context, stubs and drivers are designed to provide the complete environment for a module so that testing can be carried out. The role of stub and driver modules is pictorially shown in Figure 10.3. We briefly discuss the stub and driver modules that are required to provide the necessary environment for carrying out unit testing are briefly discussed in the following.

Stub: A sub module consists of several stub procedures that are called by the module under test. A stub procedure is a dummy procedure that takes the same parameters as the function called by the unit under test but has a highly simplified behavior. For example, a stub procedure may produce the expected behavior using a simple table look up

mechanism, rather than performing actual computations.

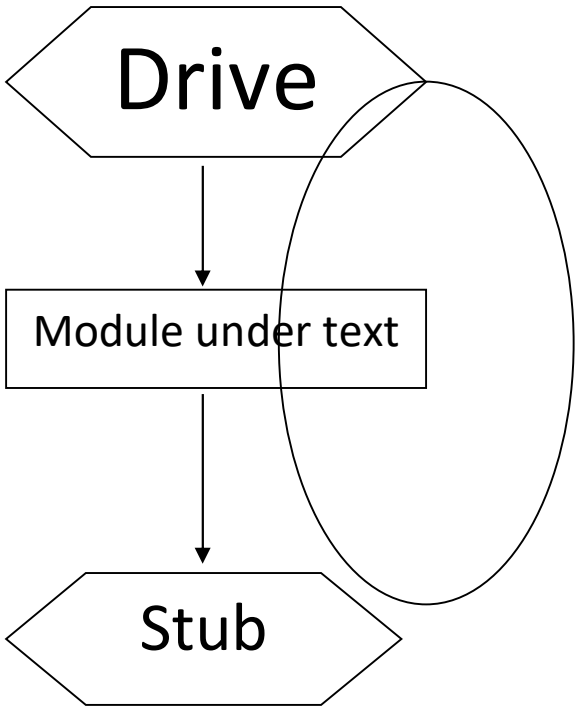


FIGURE Unit testing with the help of driver and stub modules.

Driver: A driver module contains the non-local data structures that are accessed by the module under test. Additionally, it should also have the code to call the different functions of the unit under test with appropriate parameter values for testing.

■ BLACK-BOX TESTING

In black-box testing, test cases are designed from an examination of the input/output values only and no knowledge of design or code is required. The following are the two main approaches available to design black box test cases:

- Equivalence class partitioning
- Boundary value analysis

In the following subsections, we will elaborate these two test case design techniques.

- Equivalence Class Partitioning

In the equivalence class partitioning approach, the domain of input values to the unit under test is partitioned into a set of equivalence classes. The partitioning is done such that for every input data belonging to the same equivalence class, the program behaves similarly. Equivalence classes for a unit under test can be designed by examining the input data and output data. The following are two general guidelines for designing the equivalence classes:

- If the input data values to a system can be specified by a range of values, then one valid and two invalid equivalence classes can be defined. For example, if the equivalence class is the set of integers in the range 1 to 10 (i.e., [1, 10]), then the two invalid equivalence classes are [- ,0], [11,+] and the valid equivalence class is [1,10].
- If the input data assumes values from a set of discrete members of some domain, then one equivalence class for the valid input values and another equivalence class for the invalid input values should be defined. For example, if the valid equivalence classes are {A, B, C), then the invalid equivalence class is U-{A,B,C}, where U is the universe of all possible input values.
- Boundary Value Analysis

A type of programming error that is frequency committed by programmers is missing out on the special consideration that should be to the values at the boundaries of different equivalence classes of inputs. The reason behind programmers committing such errors might purely be due to psychological factors. Programmers often fail to properly address the special processing required by the input values that lie at the boundary of the different equivalence classes. For example, a programmer may improperly use < instead of <=, or conversely <= for <, etc...)

To design boundary value test cases, it is required to examine the equivalence classes to check if any of the equivalence classes contains a range of values. For those equivalence classes that are not a range of values (i.e., consist of a discrete collection of values) no boundary value test cases can be defined. For an equivalence class that is a range of values, the boundary values need to be included in the test suite. For

example, if equivalence contains the integers in the range 1 to 10, then the boundary value test suite is {0, 1, 10, 11}.

- Summary of the Black-box Test Suite Design Approach

We now summarise the important steps in the black-box test suite design approach:

- Examine the input and output values of the program.
- Identify the equivalence classes.
- Design equivalence class test cases by picking one representative value from each equivalence class.
- Design the boundary value test cases as follows. Examine if any equivalence class is a range of values. Include the values at the boundaries of such equivalence classes in the test suite.

The strategy for black-box testing is intuitive and simple. For black-box testing, the most important step is the identification of the equivalence classes. Often, the identification of the equivalence classes is not straightforward. However, with little practice one would be able to identify all equivalence classes in the input data domain. Without practice, one may overlook many equivalence classes in the input data set. Once the equivalence classes are identified, the equivalence class and boundary value test cases can be selected almost mechanically.

■ WHITE-BOX TESTING

White-box testing is an important type of unit testing. A large number of white-box testing strategies exist. Each testing strategy essentially designs test cases based on analysis of some aspect of source code and is based on some heuristic. We first discuss some basic concepts associated with white-box testing, and follow it up with a discussion on specific testing strategies.

- Basic Concepts

A white-box testing strategy can either be coverage-based or fault-based.

Fault-based testing

A fault-based testing strategy targets to detect certain types of faults. An example of a fault-based strategy is mutation testing, which is discussed later in this section.

Coverage-based testing

A coverage-based testing strategy attempts to execute (or cover) certain elements of a program. Popular examples of coverage-based testing strategies are statement branch coverage, multiple condition coverage, and path coverage-based testing.

■ INTEGRATION TESTING

Integration testing is carried out after all (or at least some of) the modules have been unit tested. Successful completion of unit testing, to a large extent, ensures that the unit (or module) as a whole works satisfactorily. In this context, the objective of integration testing is to detect the errors at the module interfaces (call parameters). For example, it is checked that no parameter mismatch occurs when one module invokes the functionality of another module. Thus, the primary objective of integration

testing is to test the module interfaces, i.e., there are no

errors in parameter passing, when one module invokes the functionality of another module.

During integration testing, different modules of a system are integrated in a planned manner using an integration plan. The integration plan specifies the steps and the order in which modules are combined to realise the full system. After each integration step, the partially integrated system is tested.

An important factor that guides the integration plan is the module dependency graph. Thus, by examining the structure chart, the integration plan can be developed. Any one (or a mixture) of the following approaches can be used to develop the test plan:

- Big-bang approach to integration testing
- Top-down approach to integration testing
- Bottom-up approach to integration testing
- Mixed (also called sandwiched) approach to integration testing

In the following subsections, we provide an overview of these approaches to integration testing.

- Big-bang approach to integration testing

Big-bang testing is the most obvious approach to integration testing. In this approach, all the modules making up a system are integrated in a single step. In simple words, all the unit tested modules of the system are simply linked together and tested. However, this technique can meaningfully be used only for very small systems. The main problem with this approach is that once a failure has been detected during integration testing, it is very difficult to localise the error as the error may potentially exist in any of the modules. Therefore, debugging errors reported during big-bang integration testing are very expensive to fix. As a result, bing-bang integration testing is almost never used for large programs.

- Bottom-up approach to integration testing

Large software products are often made up of several subsystems. A subsystem might consist of many modules which communicate among each other through well defined interfaces. In bottom-up integration testing, first the modules for the each subsystem are integrated. Thus, the subsystems can be integrated separately and independency.

The primary purpose of carrying out the integration testing a subsystem is to test whether the interfaces among various modules making up the subsystem work satisfactorily. The test cases must be carefully chosen to exercise the interfaces in all possible manners.

In a pure bottom-up testing no stubs are required, and only test-drivers are required. Large software systems normally require several levels of subsystem testing, lower-level subsystems are successively combined to form higher-level subsystems. The principal advantage of bottom-up integration testing is that several disjoint subsystems can be tested simultaneously. Another advantage of bottom-up testing is that the low-level modules get tested thoroughly, since they are exercised in each integration step. Since the low-level modules do I/O and other critical functions, testing the low-level modules thoroughly increases the reliability of the system. A disadvantage of bottom-up testing is the complexity that occurs when the system is made up of a large number of small subsystems that are at the same level. This extreme case corresponds to the big-bang approach.

- Top-down approach to integration testing

Top-down integration testing starts with the root module in the structure chart and one or two subordinate modules of the root module. After the top-level 'skeleton' has been tested, the modules that are at the immediately lower layer of the 'skeleton' are combined with it and tested. Top-down integration testing approach requires the use of program stubs to simulate the effect of lower-level routines that are called by the routines under test. Pure top-down integration does not require any driver routines. An advantage top-down integration testing is that it requires writing only stubs, and stubs are simpler to write compared to drivers. A disadvantage of the top-down integration testing approach is that in the absence of lower-level routines, it becomes difficult to exercise the top-level routines in the desired manner since the lower level routines usually perform input/output (I/0) operations.

- ## SYSTEM TESTING

After all the units of a program have been integrated

together and tested, system testing is taken up.

The system testing procedures are the same for both

object-oriented and procedural programs, since system test cases are designed solely based on the SRS document and the actual implementation (procedural or object-oriented) is immaterial.

There are three main kinds of system testing. These are essentially similar tests, but differ in who carries out the testing:

- **Alpha Testing**: Alpha testing refers to the system testing carried out by the test team within the developing organisation.
- **Beta Testing**: Beta testing is the system testing performed by a select group of friendly customers.
- **Acceptance Testing**: Acceptance testing is the system testing performed by the customer to determine whether to accept the delivery of the system.

As can be observed from the above discussions, in the different types of system tests, the test cases can be the same,but the difference is with respect to who designs test cases and carries out testing.

Before a fully integrated system is accepted for system testing, smoke testing is performed. Smoke testing is done to check whether at least the main functionalities of the software are working properly. Unless the software is stable and at least the main functionalities are working satisfactorily, system testing is not undertaken.The functionality tests are designed to check whether the software satisfies the functional requirements as documented in the SRS document. The performance tests, on the other hand, test the conformance of the system with the non-functional requirements of the system. We have already discussed how to design the functionality test cases by using a black-box approach. So, in the following subsection we discuss only smoke and performance testing.

3.3 CONCLUSION AND FUTURE SCOPE

CONCLUTION AND FUTURE SCOPE

1. AI in Science and Research

AI is making lots of progress in the scientific sector. Artificial Intelligence can handle large quantities of data and processes it quicker than human minds. This makes it perfect for research where the sources contain high data volumes.

AI is already making breakthroughs in this field. A great example is 'Eve,' which is an AI-based robot. It discovered an ingredient of toothpaste that can cure a dangerous disease like Malaria. Imagine a common substance present in an everyday item that is capable of treating Malaria; it's a significant breakthrough, no doubt.

Drug discovery is a fast-growing sector, and AI is aiding the researchers considerably in this regard. Biotechnology is another field where researchers are using AI to design microorganisms for industrial applications. Science is witnessing significant changes thanks to AI and ML. Learn more about the benefits of AI.

2. AI in Cyber Security

Cybersecurity is another field that's benefitting from AI. As organizations are transferring their data to IT networks and cloud, the threat of hackers is becoming more significant.

One triumphant attack can wreak havoc on an organization. To keep their data and resources secure, organizations are making massive investments in cybersecurity. The future scope of AI in cybersecurity is bright.

Cognitive AI is an excellent example of this field. It detects and analyses threats, while also providing insights to the analysts for making better-informed decisions. By using Machine Learning algorithms and Deep Learning networks, the AI gets better and more durable over time. This makes it capable of fighting more advanced threats that might develop with them.

Many institutions are using AI-based solutions to automate the repetitive processes present in cybersecurity. For example, IBM has IBM Resilient, which is an agnostic and open platform that gives infrastructure and hub for managing security responses.

Another field is fraud detection. AI can help in detecting frauds and help organizations and people in avoiding scams. For example, Recurrent Neural Networks are capable of detecting fraud in their early stages. They can scan extensive quantities of transactions quickly and classify them according to their trustworthiness.

By identifying fraudulent transactions and tendencies, organizations can save a lot of time and resources. It surely lessens the risk of losing money.

3. AI in Data Analysis

Data analysis can benefit largely from AI and ML. AI algorithms are capable of improving with iterations, and this way, their accuracy, and precision increase accordingly. AI can help data analysts with handling and processing large datasets.

AI can identify patterns and insights that human eyes can't notice without putting in a lot of effort. Moreover, it is faster and more scalable at doing so. For example, Google Analytics has Analytics Intelligence, which uses machine learning to help webmasters get insights on their websites faster.

You can ask Analytics Intelligence a question in simple English, and it would give you a prompt reply. It also provides webmasters with Smart Lists, Smart Goals, Conversion Probability, and other features that help the webmaster in improving the results of their site.

The scope of AI in data analytics is rising rapidly. Another example of AI applications in this sector is predicting outcomes from data. Such systems use the analytics data to predict results and the appropriate course of action to achieve those results. Learn more about AI applications.

As mentioned earlier, AI systems can handle tons of data and process it much faster than humans. So, they can take customer data and make more accurate predictions of customer behavior, preferences, and other required factors. Helixa.ai is a great example of such an AI application. They use AI to provide insights into customer (or audience) behavior for higher accuracy and better results. Agencies and marketers can use their services to build precise buyer personas and create better-targeted ad campaigns.

4. AI in Transport

The transport sector has been using AI for decades. Airplanes have been using autopilot to steer them in the air since 1912. An autopilot system controls the trajectory of a plane, but it isn't restricted to aircraft alone. Ships and spacecraft also use autopilot to help them maintain the correct course.

Autopilot helps the human operator and assists them in heading in the right direction. A pilot of a modern aircraft usually works for 7 minutes; the autopilot handles most of the steering of the plane. This allows the pilots to focus on other more important areas of the flight, such as the weather and the trajectory of the plane.

Another area where the future scope of AI is quite broad is driverless cars. Many companies are developing autonomous vehicles, which will rely heavily on AI and ML to operate optimally. Experts believe self-driving cars will bring many long-term and short-term benefits, including lower emissions and enhanced road safety. For example, self-driving cars will be free from human errors, which account for 90% of traffic accidents. Many companies, including Tesla and Uber, are developing these vehicles.

5. AI in Home

AI has found a special place in people's homes in the form of Smart Home Assistants. Amazon Echo and Google Home are popular smart home devices that let you perform various tasks with just voice commands.

You can order groceries, play music, or even switch on/off the lights in your living room with just a few voice commands. Both of them rely on Voice Recognition technologies, which are a result of Artificial Intelligence and Machine Learning. They constantly learn from the commands of their users to understand them better and become more efficient.

Smart assistants are also present in mobile phones. Apple's Siri and Google Assistant are great examples of this sort. They also learn to recognize their users' voices to interpret them better all the time. And they can perform a plethora of tasks. Microsoft also has a smart assistant, which is called Cortana.

You can use these smart assistants for various tasks such as:

- Playing a song
- Asking a question
- Buying something online
- Opening an app

There's a lot of room left for improvement, but surely, the scope of AI in the smart home sector is booming.

6. AI in Healthcare

The medical sector is also using this technology for its advantages. AI is helping medical researchers and professionals in numerous ways.

For example, the Knight Career Institute and Intel have made a collaborative cancer cloud. This cloud takes data from the medical history of cancer (and similar) patients to help doctors in making a better diagnosis. Preventing cancer from moving to higher stages is its most effective treatment at this time.

We've already mentioned how AI is helping researchers in their field too. Apart from finding a cure for cancer, some organizations are using AI to help patients get telemedicine. The UK's National Health Service uses Google's DeepMind platform to detect health risks in people through apps.

Wrong diagnoses are a significant problem in the medical sector. AI can help doctors in avoiding these errors by providing them with relevant databases and recommendations. It can analyze the database of patients with similar symptoms and suggest the treatment that was the most successful in those cases.

Many major organizations, including IBM and Microsoft, are collaborating with medical institutions to solve the various problems present in the healthcare sector.

AI can also help in reducing medical costs by preventing diseases beforehand and helping doctors in making better diagnoses. BCIs (Brain-computer Interfaces) is another area where the medical sector is utilizing AI. These interfaces help in predicting problems related to speaking or moving that might develop due to problems in the brain. They use AI to help these patients overcome these issues, too, by decoding neural activates.