**INTRODUCTION**

**1.INTRODUCTION**

**1.1 MOTIVATION**

Mitchell’s famous machine learning textbook [1] begins with the statement: “Ever since computers were invented, we have wondered whether they might be made to learn. If we could understand how to program them to learn - to improve automatically with experience - the impact would be dramatic”. This quest gave birth to a new research area, i.e., machine learning, for Computer Science decades ago. Till now, machine learning techniques have been deeply rooted in our every day’s life, such as recommendation when we are reading news and handwriting recognition when we are using our cell-phones. Furthermore, machine learning has also gained significant achievements. For example, AlphaGO [2] defeated human champion in the game of GO, ResNet [3] surpassed human performance in image recognition, Microsoft’s speech system approximated human level in speech transcription. However, these successful applications of machine learning are far from fully automated, i.e., “improving automatically with experience ”. Since there are no algorithms that can achieve good performance on all possible learning problems with equal importance (according to No Free Lunch theorems [5] [6]), every aspect of machine learning applications, such as feature engineering, model selection, and algorithm selection needs to be carefully configured. Human experts are hence heavily involved in machine learning applications. As these experts are rare, the success of machine learning comes at a great price.

We know machine learning finds its use in many of the technical and non-technical streams of the 21st century. But at the same time its not knowledge and labour friendly to inculcate in any running infrastructure. So to utilize the power of multiple machine learning algorithms we created a platform where the prediction and classification tasks can be performed on any type of data by just uploading the data to the platform.This platform trains multiple machine learning algorithms on the uploaded data and at the same time checks the accuracy of the models. The model with the best accuracy can then be used to predict the required results.

**1.2 OBJECTIVES**

The objective of the proposed system is to build that provides the facility to the user to upload their dataset and get solutions with in less time.

**LITERATURE SURVEY**

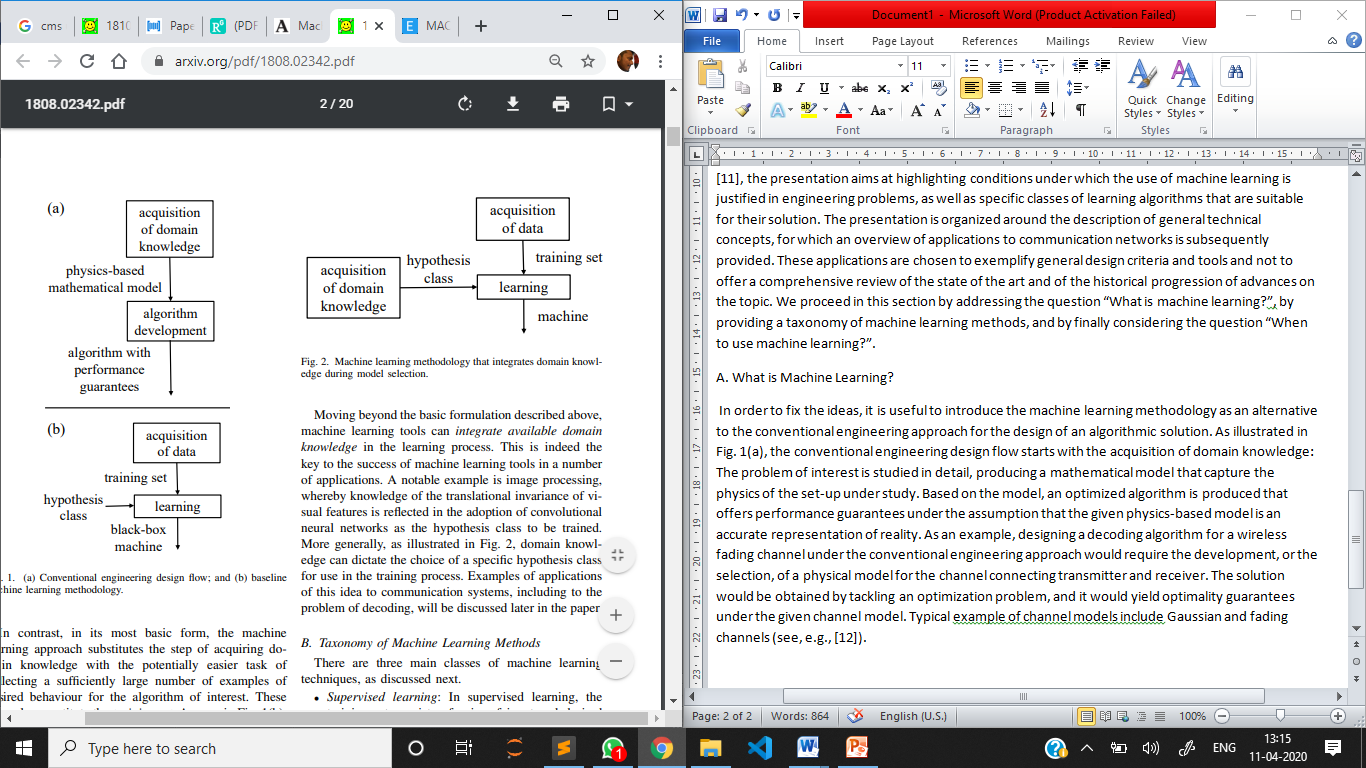
**2. LITERATURE SURVEY**

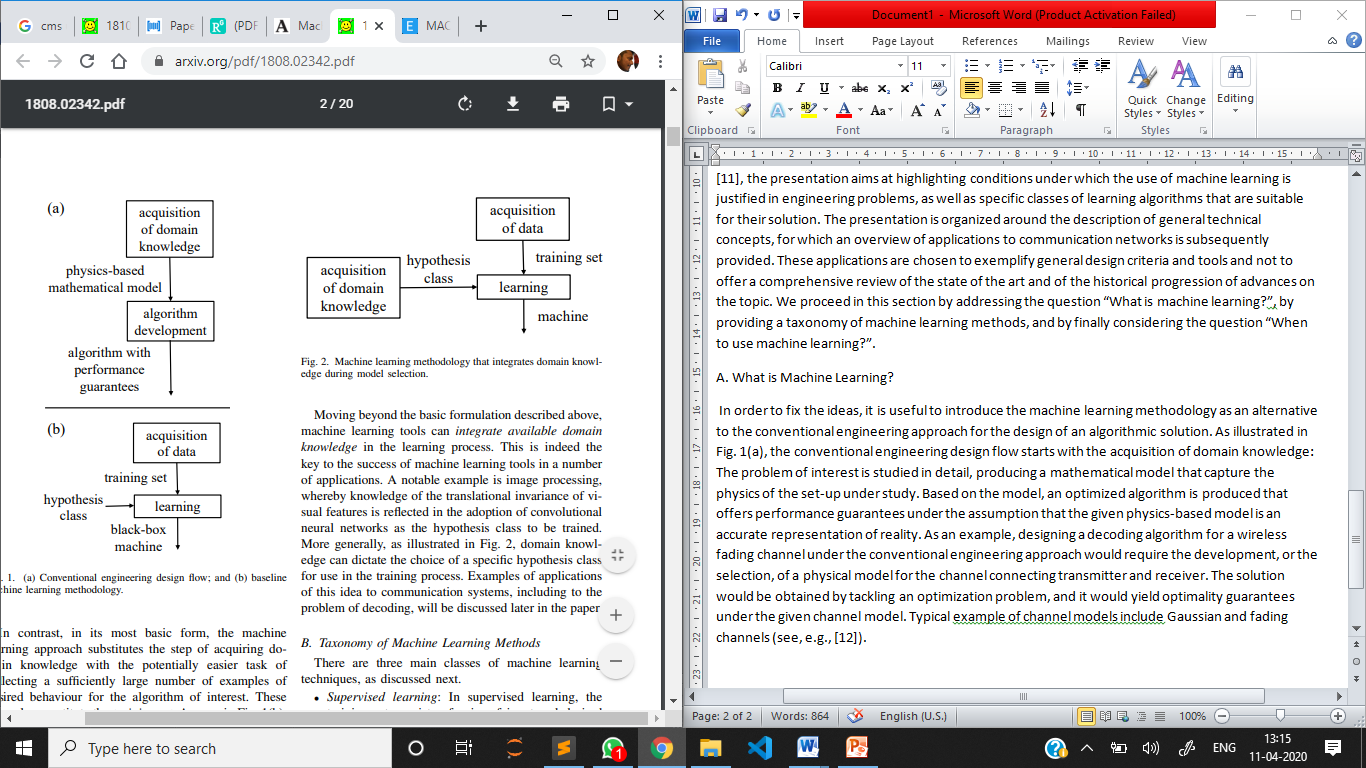
After the “AI winter” of the 80s and the 90s, interest in the application of data-driven Artificial Intelligence (AI) techniques has been steadily increasing in a number of engineering fields, including speech and image analysis and communications . Unlike the logic-based expert systems that were dominant in the earlier work on AI the renewed confidence in data driven methods is motivated by the successes of pattern recognition tools based on machine learning. These tools rely on decades-old algorithms, such as back propagation , the Expectation Maximization (EM) algorithm , and Q-learning , with a number of modern algorithmic advances, including novel regularization techniques and adaptive learning rate schedules Their success is built on the unprecedented availability of data and computing resources in many engineering domains. While the new wave of promises and breakthroughs around machine learning arguably falls short, at least for now, of the requirements that drove early AI research learning algorithms have proven to be useful in a number of important applications – and more is certainly on the way.

This paper provides a very brief introduction to key concepts in machine learning and to the literature on machine learning for communication systems. Unlike other review papers such as, the presentation aims at highlighting conditions under which the use of machine learning is justified in engineering problems, as well as specific classes of learning algorithms that are suitable for their solution. The presentation is organized around the description of general technical concepts, for which an overview of applications to communication networks is subsequently provided. These applications are chosen to exemplify general design criteria and tools and not to offer a comprehensive review of the state of the art and of the historical progression of advances on the topic. We proceed in this section by addressing the question “What is machine learning?”, by providing a taxonomy of machine learning methods, and by finally considering the question “When to use machine learning?”.

A. What is Machine Learning?

In order to fix the ideas, it is useful to introduce the machine learning methodology as an alternative to the conventional engineering approach for the design of an algorithmic solution. As illustrated in Fig. 1(a), the conventional engineering design flow starts with the acquisition of domain knowledge: The problem of interest is studied in detail, producing a mathematical model that capture the physics of the set-up under study. Based on the model, an optimized algorithm is produced that offers performance guarantees under the assumption that the given physics-based model is an accurate representation of reality. As an example, designing a decoding algorithm for a wireless fading channel under the conventional engineering approach would require the development, or the selection, of a physical model for the channel connecting transmitter and receiver. The solution would be obtained by tackling an optimization problem, and it would yield optimality guarantees under the given channel model. Typical example of channel models include Gaussian and fading channels .





In contrast, in its most basic form, the machine learning approach substitutes the step of acquiring domain knowledge with the potentially easier task of collecting a sufficiently large number of examples of desired behaviour for the algorithm of interest. These examples constitute the training set. As seen in Fig. 1(b), the examples in the training set are fed to a learning algorithm to produce a trained “machine” that carries out the desired task. Learning is made possible by the choice of a set of possible “machines”, also known as the hypothesis class, from which the learning algorithm makes a selection during training. An example of an hypothesis class is given by a neural network architecture with learnable synaptic weights. Learning algorithms are generally based on the optimization of a performance criterion that measures how well the selected “machine” matches the available data. For the problem of designing a channel decoder, a machine learning approach can hence operate even in the absence of a well-established channel model. It is in fact enough to have a sufficiently large number of examples of received signals – the inputs to the decoding machine – and transmitted messages – the desired outputs of the decoding machine – to be used for the training of a given class of decoding functions

Moving beyond the basic formulation described above, machine learning tools can integrate available domain knowledge in the learning process. This is indeed the key to the success of machine learning tools in a number of applications. A notable example is image processing, whereby knowledge of the translational invariance of visual features is reflected in the adoption of convolutional neural networks as the hypothesis class to be trained. More generally, as illustrated in Fig. 2, domain knowledge can dictate the choice of a specific hypothesis class for use in the training process. Examples of applications of this idea to communication systems, including to the problem of decoding, will be discussed later in the paper.

B. Taxonomy of Machine Learning

Methods There are three main classes of machine learning techniques, as discussed next. •

* Supervised learning: In supervised learning, the training set consists of pairs of input and desired output, and the goal is that of learning a mapping between input and output spaces. As an illustration, in Fig. 3(a), the inputs are points in the twodimensional plane, the outputs are the labels assigned to each input (circles or crosses), and the goal is to learn a binary classifier. Applications include the channel decoder discussed above, as well as email spam classification on the basis of examples of spam/ non-spam emails.
* Unsupervised learning: In unsupervised learning, the training set consists of unlabelled inputs, that is, of inputs without any assigned desired output. For instance, in Fig. 3(b), the inputs are again points in the two-dimensional plane, but no indication is provided by the data about the corresponding desired output. Unsupervised learning generally aims at discovering properties of the mechanism generating the data. In the example of Fig. 3(b), the goal of unsupervised learning is to cluster together input points that are close to each other, hence assigning a label – the cluster index – to each input point (clusters are delimited by dashed lines). Applications include clustering of documents with similar topics. It is emphasized that clustering is only one of the learning tasks that fall under the category of unsupervised learning (see Sec. V)
* Reinforcement learning: Reinforcement learning lies, in a sense, between supervised and unsupervised learning. Unlike unsupervised learning, some form of supervision exists, but this does not come in the form of the specification of a desired output for every input in the data. Instead, a reinforcement learning algorithm receives feedback from the environment only after selecting an output for a given input or observation. The feedback indicates the degree to which the output, known as action in reinforcement learning, fulfils the goals of the learner. Reinforcement learning applies to sequential decision making problems in which the learner interacts with an environment by sequentially taking actions – the outputs – on the basis of its observations – its inputs – while receiving feedback regarding each selected action. Most current machine learning applications fall in the supervised learning category, and hence aim at learning an existing pattern between inputs and outputs. Supervised learning is relatively well-understood at a theoretical level and it benefits from well established algorithmic tools. Unsupervised learning has so far defied a unified theoretical treatment .Nevertheless, it arguably poses a more fundamental practical problem in that it directly tackles the challenge of learning by direct observation without any form of explicit feedback. Reinforcement learning has found extensive applications in problems that are characterized by clear feedback signals, such as win/lose outcomes in games, and that entail searches over large trees of possible action-observation histories.
* This paper only covers supervised . Reinforcement learning requires a different analytical framework grounded in Markov Decision Processes and will not be discussed here For a broader discussion on the technical aspects of supervised and unsupervised learning, we point to and references therein.

C. When to Use Machine Learning?

Based on the discussion in Sec. I-A, the use of a machine learning approach in lieu of a more conventional engineering design should be justified on a case-bycase basis on the basis of its suitability and potential advantages. The following criteria, inspired by offer useful guidelines on the type of engineering tasks that can benefit from the use of machine learning tools.

1.The traditional engineering flow is not applicable or is undesirable due to a model deficit or to an algorithm deficit

• With a model deficit, no physics-based mathematical models exist for the problem due to insufficient domain knowledge. As a result, a conventional model-based design is inapplicable.

• With an algorithm deficit, a well-established mathematical model is available, but existing algorithms optimized on the basis of such model are too complex to be implemented for the given application. In this case, the use of hypothesis classes including efficient “machines”, such as neural network of limited size or with tailored hardware implementations ,can yield lower-complexity solutions.

2. A sufficiently large training data sets exist or can be created.

3. The task does not require the application of logic, common sense, or explicit reasoning based on background knowledge.

4. The task does not require detailed explanations for how the decision was made. The trained machine is by and large a black box that maps inputs to outputs. As such, it does not provide direct means to ascertain why a given output has been produced in response to an input, although recent research has made some progress on this front. This contrasts with engineered optimal solutions, which can be typically interpreted on the basis of physical performance criteria. For instance, a maximum likelihood decoder chooses a given output because it minimizes the probability of error under the assumed model.

5. The phenomenon or function being learned is stationary for a sufficiently long period of time. This is in order to enable data collection and learning.

6. The task has either loose requirement constraints, or, in the case of an algorithm deficit, the required performance guarantees can be provided via numerical simulations. With the conventional engineering approach, theoretical performance guarantees can be obtained that are backed by a physics-based mathematical model. These guarantees can be relied upon insofar as the model is trusted to be an accurate representation of reality. If a machine learning approach is used to address an algorithm deficit and a physics-based model is available, then numerical results may be sufficient in order to compute satisfactory performance measures. In contrast, weaker guarantees can be offered by machine learning in the absence of a physics-based model. In this case, one can provide performance bounds only under the assumptions that the hypothesis class is sufficiently general to include “machines” that can perform well on the problem and that the data is representative of the actual data distribution to be encountered at runtime .The selection of a biased hypothesis class or the use of an unrepresentative data set may hence yield strongly suboptimal performance. We will return to these criteria when discussing applications to communication systems.

**SYSTEM ANALYSIS**

**3. SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

The existing system is incapable of running multiple algorithms at a time rather it works with only one algorithm at a time. If multiple algorithms are required to run at the same time, that needs extensive coding thus it is not beginner friendly. The existing system tend making use of the local system hardware making it less efficient.

**3.2 PROBLEM STATEMENT**

In the existing system, the time taken for running model is high and needs high coding skills manually .It becomes nearly impossible to generate result for huge amount of data present in today’s world.

**3.3 PROPOSED SYSTEM**

The proposed system overcomes the limitations of the existing system by enabling the working of multiple algorithms at the same time. Since extensive coding is not required it also gets a beginner friendly approach. Another limitation which the proposed overcomes is the hardware efficiency as it makes use of server side hardware.

**3.4 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

**1. ECONOMICAL FEASIBILITY**

**2. TECHNICAL FEASIBILITY**

**3. SOCIAL FEASIBILITY**

**3.4.1 ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### 3.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**3.4.3 SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently.

**(i) Technical feasibility**

In the technical feasibility study, one has to test whether the proposed system can be developed using existing technology or not. It is planned to implement the proposed system in Android. The project entitled is technically feasible because of the following reasons.

(i) All necessary technology exists to develop the system.

(ii) The existing system is so flexible that it can be developed further.

**(ii) Economic feasibility**

As a part of this, the costs and benefits associated with the proposed systems are to be compared. The project is economically feasible only if tangible and intangible benefits outweigh the cost. We can say the proposed system is feasible based on the following grounds.

(i) The cost of developing the full system is reasonable.

(ii) The cost of hardware and software for the application is less.

**(iii) Operational feasibility**

The project is operationally feasible because there is sufficient support from the project management and the users of the proposed system. Proposed system definitely does not harm and will not produce the bad results and no problem will arise after implementation of the system.

**3.1 REQUIREMENTS**

Requirements analysis is done in order to understand the problem the software system is to solve. The problem could be automating an existing manual process, developing a new automated system, or a combination of the two. For large systems that have many features, and that need to perform many different tasks, understanding the requirements of the system is a major task.

The emphasis in requirements analysis is on identifying *what* is needed from the system, not *how* the system will achieve its goals. This task is complicated by the fact that there are often at least two parties involved in software development-a client and a developer. The developer usually does not understand the client’s problem domain and the client does not understand the issues involved in the system software systems developed by the developers. Hence causes a communication gap between them.

**3.1.1 SRS**

This communication gap is bridged during the analysis. This analysis phase ends with a document describing all the requirements called as SRS (Software Requirements Specification).

There are two major activities involved in this phase. Problem understanding or analysis and requirement specification. In problem analysis, the analyst has to understand the problem and its context. Such analysis typically requires through understanding of the existing system, parts which have to be automated. A clear understanding is needed of the important data entities in the system, major centers where action is taken, the purpose of the different actions that are performed and the inputs and outputs.

**3.1.2 Functional Requirements**

The functional requirements describe the interactions between the system and its environment independent of its implementation.

The proposed system should provide the facilities for the following modules.

* Registration Module
* Processing Module
* Output Module

**Registration Module:-**

In this module the user is required to get registered with the platform to make use of the services offered by the platform.

**Processing Module:-**

The data uploaded to the platform is fed to the algorithms for the processing.

**Output Module:-**

The results obtained after the processing must be diplayed and to do so, this module is included. This module arranges the resultant accuracies in the best to worst order making it easy to interpret.

**3.1.3 Non-Functional Requirements**

Non-functional requirements describe user-visible aspects of the system that are not directly related to functionality of the system.

**User Interface**

A menu interface has been provided to the client to be user friendly.

**Documentations**

The client is provided with an introductory help about the client interface and the user documentation has been developed through help hyperlink.

**Performance Constraints**

* Requests should be processed within no time.
* Users should be authenticated for accessing the requested data.

Error Handling and Extreme Conditions:

In case of User Error, the System should display a meaningful error message to the user, such   that the user can correct his Error.

The high-level components in proposed system should handle exceptions that occur while connecting to database server, IOExceptions etc.

**3.5 Software Requirement Specification**

**3.5.1 Software Requirements**

Operating System : Windows, Linux or UNIX

Software : Anaconda

IDE : Visual Studio

Frame work : Flask

Data Base : SQLite

Code Language : Python 3.7 and above

Front End : HTML,CSS

**3.5.2 Hardware Requirements**

Processor : Intel i3, i5, i7 and above

Hard Disk : 100 GB

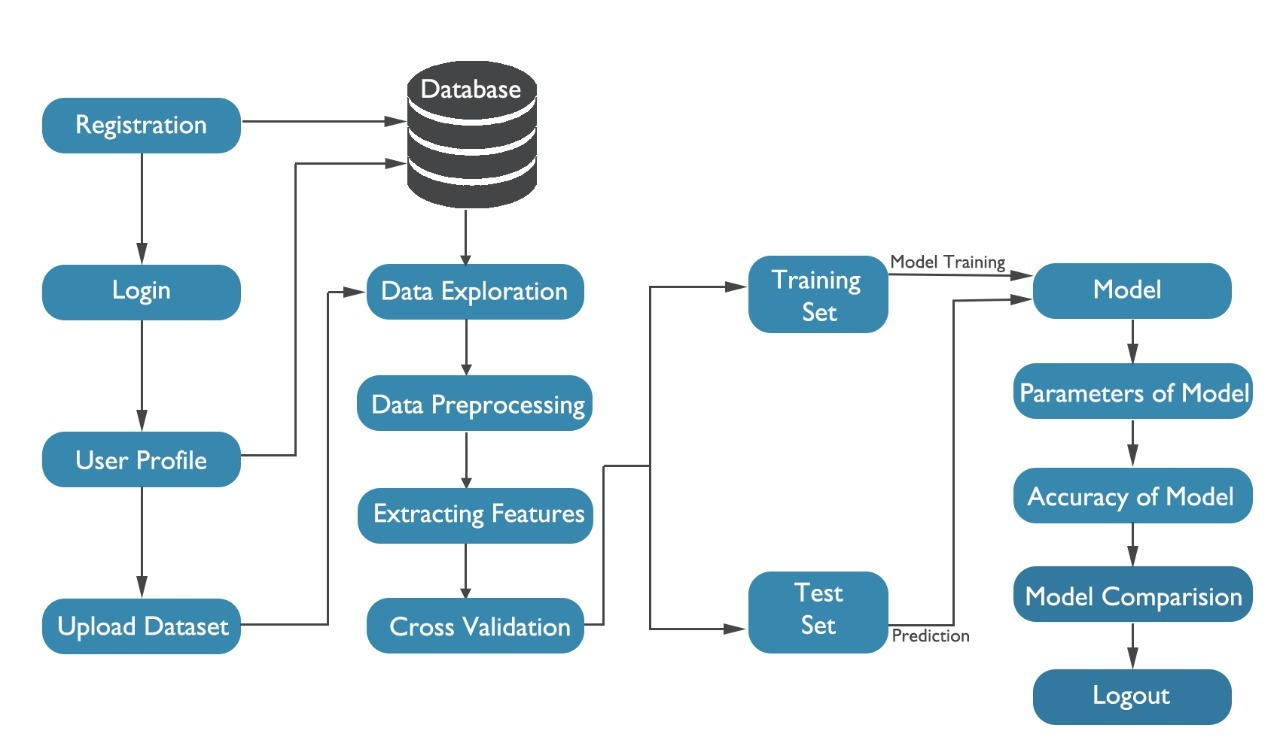
RAM : 4 GB or above

**SYSTEM DESIGN**

**4. SYSTEM DESIGN**

**4.1 SYSTEM ARCHITECTURE**

Here, we provide a brief overview of the process of semi-automated machine learning .A more detailed description is provided in Chapter 5. A diagram summarizing the steps required for semi-automated machine learning is shown in Figure 1.1.The registration module will allow new user to register into the system and then login to system. The user profile will interface to upload dataset for model training.



**4.1.2 Data Exploration**

Data exploration is an approach similar to initial data analysis, whereby a data analyst uses visual exploration to understand what is in a dataset and the characteristics of the data, rather than through traditional data management systems. These characteristics can include size or amount of data, completeness of the data, correctness of the data, possible relationships amongst data elements or files/tables in the data.

Data exploration is typically conducted using a combination of automated and manual activities. Automated activities can include data profiling or data visualization or tabular reports to give the analyst an initial view into the data and an understanding of key characteristics.

This is often followed by manual drill-down or filtering of the data to identify anomalies or patterns identified through the automated actions. Data exploration can also require manual scripting and queries into the data (e.g. using languages such as SQL or R) or using spreadsheets or similar tools to view the raw data.

All of these activities are aimed at creating a mental model and understanding of the data in the mind of the analyst, and defining basic metadata (statistics, structure, relationships) for the data set that can be used in further analysis.

Once this initial understanding of the data is had, the data can be pruned or refined by removing unusable parts of the data (data cleansing), correcting poorly formatted elements and defining relevant relationships across datasets. This process is also known as determining data quality.

Data exploration can also refer to the ad hoc querying and visualization of data to identify potential relationships or insights that may be hidden in the data.

**4.1.3 Data Pre-processing**

Data pre-processing is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly applicable to data mining and machine learning projects. Data-gathering methods are often loosely controlled, resulting in out-of-range values (e.g., Income: −100), impossible data combinations (e.g., Sex: Male, Pregnant: Yes), missing values, etc. Analysing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running an analysis often, data pre-processing is the most important phase of a machine learning project, especially in computational biology.

**4.1.4 Extracting Features**

In machine learning, pattern recognition and in image processing, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations.

When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters, or the repetitiveness of images presented as pixels), then it can be transformed into a reduced set of features (also named a feature vector). Determining a subset of the initial features is called feature selection. The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data

**4.1.5 Data Splitting**

**Training Dataset**: The sample of data used to fit the model.

**Test Dataset**: The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

**4.1.6 Model Building**

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop conventional algorithms to perform the needed tasks.

**4.1.7 Accuracy of Models**

Machine learning model accuracy is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data. The better a model can generalize to ‘unseen’ data, the better predictions and insights it can produce, which in turn deliver more business value.

**4.1.8 Parameters of Models**

A model parameter is a configuration variable that is internal to the model and whose value can be estimated from the given data. They are required by the model when making predictions. Their values define the skill of the model on your problem.

**4.2 UML**

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering.

The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object-oriented computer software.

In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non-software systems

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems.

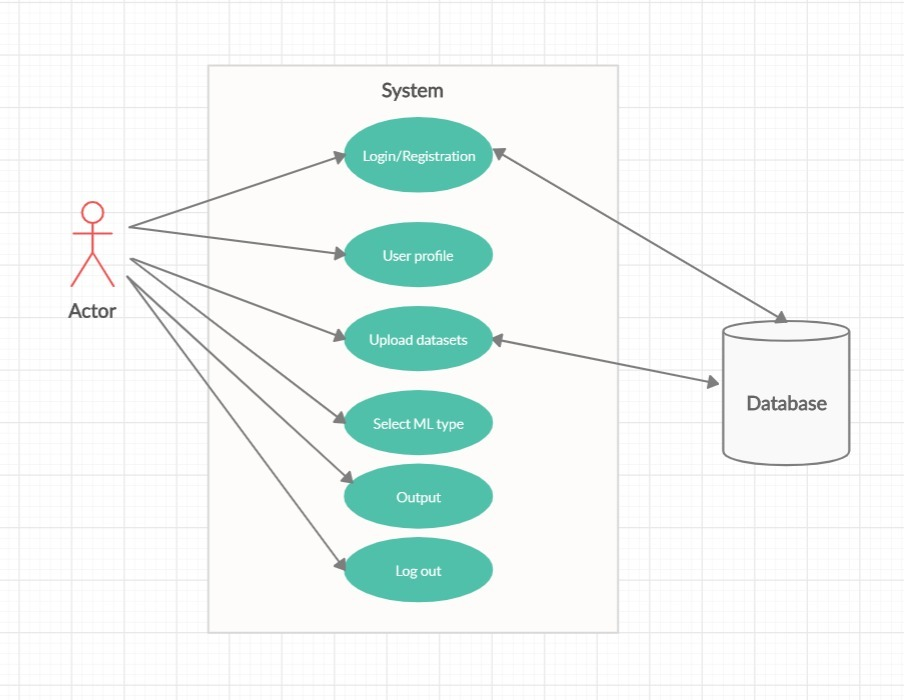
The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**Goals:** The Primary goals in the design of the UML are as follows:

* Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
* Provide extendibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development process.
* Provide a formal basis for understanding the modelling language.
* Encourage the growth of tools market.
* Support higher level development concepts such as collaborations, frameworks, patterns and components.
* Integrate best practices.

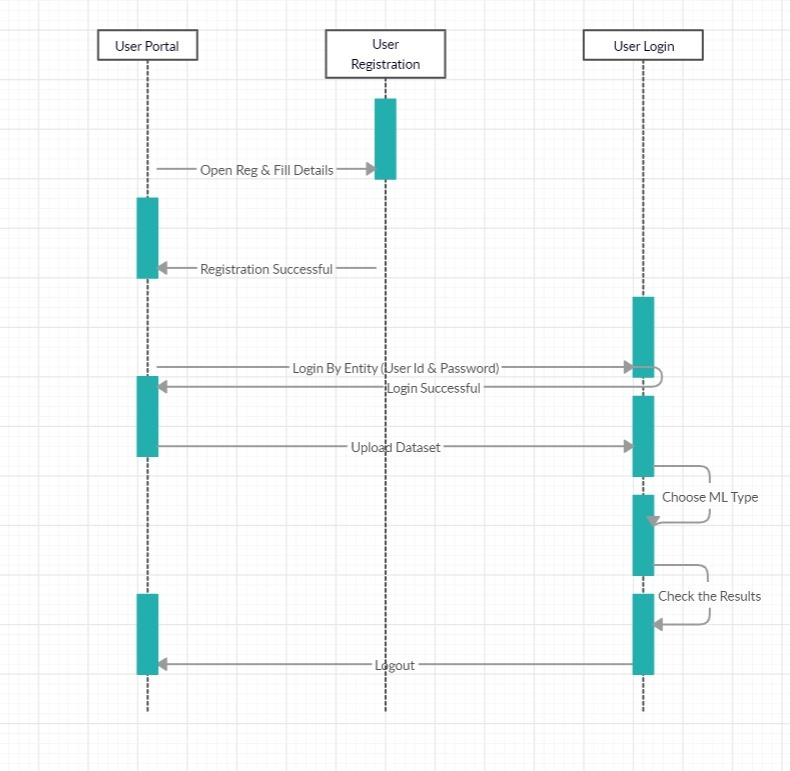
**4.2.1 Use Case Diagram:**

UML provides the use case diagram to facilitate the process of requirements gathering. The use case diagram models the interactions between the system’s external clients and the use cases of the system. Each use case represents a different capability that the system provides the client.



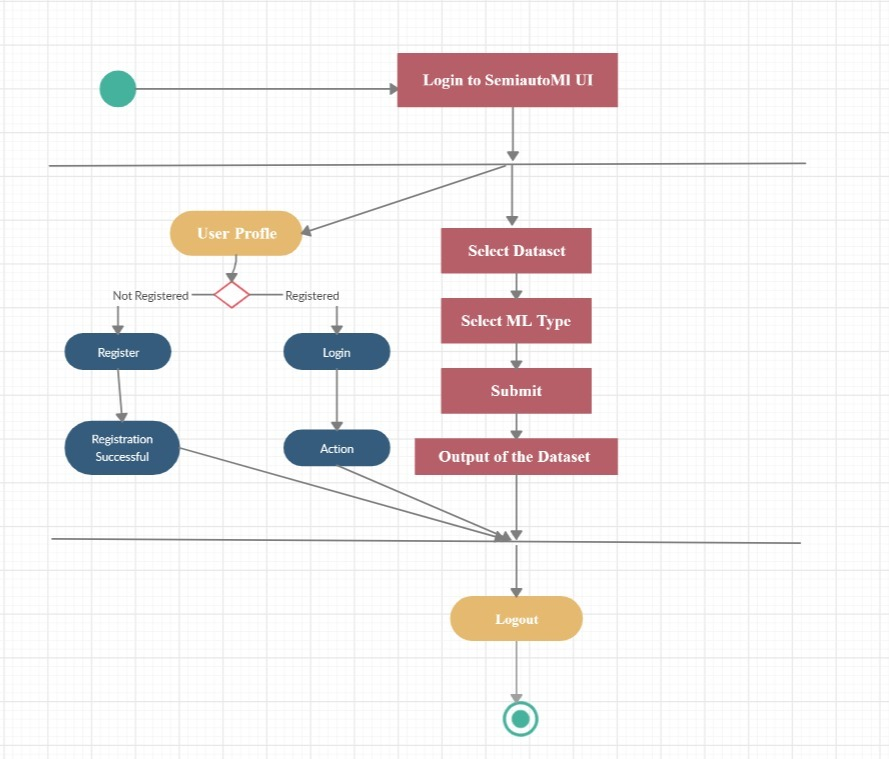
**4.2.2 Sequence Diagram:**

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

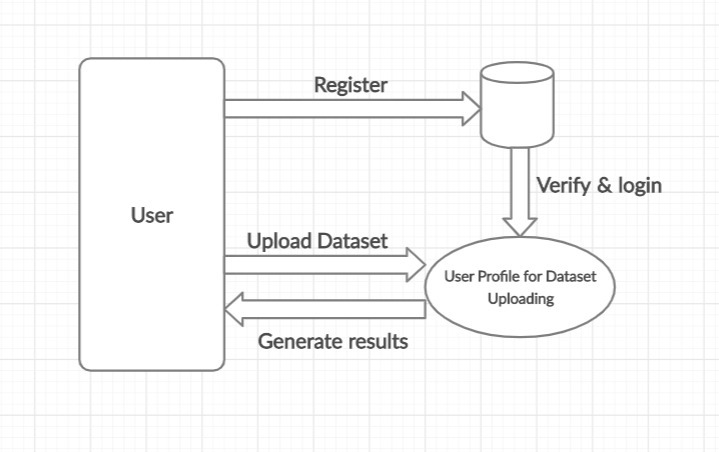


**4.2.3 Activity Diagram:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



**4.3 DATA FLOW DIAGRAMS**



**SYSTEM IMPLEMENTATION**

**5. SYSTEM IMPLEMENTATION**

**5. IMPLEMENTATION AND TESTING**

**5.1 METHOD OF IMPLEMENTATION**

**5.1.1 PYTHON**

**A Brief History of Python:**

Python is a widely used general-purpose, high-level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

In the late 1980s, history was about to be written. It was that time when working on Python started. Soon after that, Guido Van Rossum began doing its application based work in December of 1989 by at Centrum Wiskunde & Informatica (CWI) which is situated in Netherland. It was started firstly as a hobby project because he was looking for an interesting project to keep him occupied during Christmas. The programming language which Python is said to have succeeded is ABC Programming Language, which had the interfacing with the Amoeba Operating System and had the feature of exception handling. He had already helped to create ABC earlier in his career and he had seen some issues with ABC but liked most of the features. After that what he did as really very clever. He had taken the syntax of ABC, and some of its good features. It came with a lot of complaints too, so he fixed those issues completely and had created a good scripting language which had removed all the flaws. The inspiration for the name came from BBC’s TV Show – ‘Monty Python’s Flying Circus’, as he was a big fan of the TV show and also he wanted a short, unique and slightly mysterious name for his invention and hence he named it Python! He was the “Benevolent dictator for life” (BDFL) until he stepped down from the position as the leader on 12th July 2018. For quite some time he used to work for Google, but currently, he is working at Dropbox.

The language was finally released in 1991. When it was released, it used a lot fewer codes to express the concepts, when we compare it with Java, C++ & C. Its design philosophy was quite good too. Its main objective is to provide code readability and advanced developer productivity. When it was released it had more than enough capability to provide classes with inheritance, several core data types exception handling and functions.

**Features of Python:**

Python is known for its general purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development.

Here, we are specifing applications areas where python can be applied.

**1) Web Applications**

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautifulSoup, Feedparser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and delelop web based applications. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc.

**2) Desktop GUI Applications**

Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

**3) Software Development**

Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.

**4) Scientific and Numeric**

Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.

**5) Business Applications**

Python is used to build Bussiness applications like ERP and e-commerce systems. Tryton is a high level application platform.

**6) Console Based Application**

We can use Python to develop console based applications. For example: IPython.

**7) Audio or Video based Applications**

Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.

**8) 3D CAD Applications**

To create CAD application Fandango is a real application which provides full features of CAD.

**9) Enterprise Applications**

Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.

**10) Applications for Images**

Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc.

There are several such applications which can be developed using Python

**5.1.2 PYTHON FLASK**

Flask Tutorial provides the basic and advanced concepts of the Python Flask framework. Our Flask tutorial is designed for beginners and professionals.

Flask is a web framework that provides libraries to build lightweight web applications in python. It is developed by Armin Ronacher who leads an international group of python enthusiasts (POCCO).

**What is Flask?**

Flask is a web framework that provides libraries to build lightweight web applications in python. It is developed by Armin Ronacher who leads an international group of python enthusiasts (POCCO). It is based on WSGI toolkit and jinja2 template engine. Flask is considered as a micro framework.

**What is WSGI?**

It is an acronym for web server gateway interface which is a standard for python web application development. It is considered as the specification for the universal interface between the web server and web application.

**What is Jinja2?**

Jinja2 is a web template engine which combines a template with a certain data source to render the dynamic web pages.

**FLASK ENVIRONMENT SETUP**

To install flask on the system, we need to have python 2.7 or higher installed on our system. However, we suggest using python 3 for the development in the flask.

*Install virtual environment (virtualenv)*

virtualenv is considered as the virtual python environment builder which is used to create the multiple python virtual environment side by side. It can be installed by using the following command.

*$ pip install virtualenv*

Once it is installed, we can create the new virtual environment into a folder as given below.

*$ mkdir new*

*$ cd new*

*$ virtualenv venv*

To activate the corresponding environment, use the following command on the Linux operating system.

*$ venv/bin/activate*

On windows, use the following command.

*$ venv\scripts\activate*

We can now install the flask by using the following command.

*$ pip install flask*

**5.1.3 SQLite**

SQLite is embedded relational database management system. It is self-contained, serverless, zero configuration and transactional SQL database engine.

Our SQLite Tutorial includes all topics of SQLite such as SQLite with history, features, advantages, installation, commands, syntax, datatypes, operators, expressions, databases, table, crud operations, clauses, like, glob, limit, and clause, advance sqlite etc.

**What is SQLite**

SQLite is embedded relational database management system. It is self-contained, serverless, zero configuration and transactional SQL database engine.

SQLite is free to use for any purpose commercial or private. In other words, "SQLite is an open source, zero-configuration, self-contained, stand alone, transaction relational database engine designed to be embedded into an application".

SQLite is different from other SQL databases because unlike most other SQL databases, SQLite does not have a separate server process. It reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file.

**SQLite Features/ Why to use SQLite**

Following is a list of features which makes SQLite popular among other lightweight databases:

**SQLite is totally free:** SQLite is open-source. So, no license is required to work with it.

**SQLite is serverless:** SQLite doesn't require a different server process or system to operate.

**SQLite is very flexible:** It facilitates you to work on multiple databases on the same session on the same time.

**Configuration Not Required:** SQLite doesn't require configuration. No setup or administration required.

**SQLite is a cross-platform DBMS:** You don't need a large range of different platforms like Windows, Mac OS, Linux, and Unix. It can also be used on a lot of embedded operating systems like Symbian, and Windows CE.

**Storing data is easy:** SQLite provides an efficient way to store data.

**Variable length of columns:** The length of the columns is variable and is not fixed. It facilitates you to allocate only the space a field needs. For example, if you have a varchar(200) column, and you put a 10 characters' length value on it, then SQLite will allocate only 20 characters' space for that value not the whole 200 space.

Provide large number of API's: SQLite provides API for a large range of programming languages. For example: .Net languages (Visual Basic, C#), PHP, Java, Objective C, Python and a lot of other programming language.

SQLite is written in ANSI-C and provides simple and easy-to-use API.

SQLite is available on UNIX (Linux, Mac OS-X, Android, iOS) and Windows (Win32, WinCE, WinRT).

**5.1.4 HTML:**

HTML, an acronym of Hyper Text Markup Language, is the predominant markup language for web pages. It provides a means to describe the structure of text-based information in a document by denoting certain text as links, headings, paragraphs, lists, and so on and to supplement that text with interactive forms, embedded images, and other objects.

HTML is written in the form of tags, surrounded by angle brackets. HTML can also describe, to some degree, the appearance and semantics of a document, and can include embedded scripting language code (such as JavaScript) which can affect the behaviour of Web browsers and other HTML processors.

Web pages are built with the help of this HTML which are called the Web Documents. We used the following tags in our project.

**TABLE:**

Tables are so popular with web page authors is that they let you arrange the elements of a web page in such a way that the browser won‘t rearrange them web page authors frequently use tables to structure web pages.

**TR:**

TR is used to create a row in a table encloses <TH> and <TD> elements. <TR> contain many attributes. Some of them are,

● ALIGN: specifies the horizontal alignment of the text in the table row.

● BGCOLOR: Specifies the background color for the row.

● BORDERCOLOR: Sets the external border color for the row.

● VALIGN: Sets the vertical alignment of the data in this row.

**TH:**

TH is used to create table heading.

**●** ALIGN: Sets the horizontal alignment of the content in the table cell. Sets LEFT, RIGHT, CENTER.

**●** BACKGROUND: Species the background image for the table cell.

**●** BGCOLOR: Specifies the background color of the table cell.

**●** VALIGN: Sets the vertical alignment of the data. Sets to TOP, MIDDLE, BOTTOM or BASELINE.

**●** WIDTH: Specifies the width of the cell. Set to a pixel width or a percentage of the display area.

**TD:**

TD is used to create table data that appears in the cells of a table.

● ALIGN: Species the horizontal alignment of content in the table cell. Sets to LEFT, CENTER, RIGHT.

● BGCOLOR: Specifies the background image for the table cell.

● BGCOLOR: sets the background color of the table cells.

● WIDTH: Species the width of the cell

**FRAMES:**

Frames are used for either run off the page or display only small slices of what are supposed to be shown and to configure the frame we can use <FRAMESET>

There are two important points to consider when working with <FRAMESET>.

● <FRAMESET> element actually takes the place of the <BODY> element in a document.

● Specifying actual pixel dimensions for frames.

<FRAME> Elements are used to create actual frames.

From the frame set point of view dividing the browser into two vertical frames means creating two columns using the <FRAMESET> elements COLS attribute.

The syntax for vertical fragmentation is,

<FRAMESET COLS =‖50%, 50%‖>

</FRAMESET>

Similarly, if we replace COLS with ROWS then we get horizontal fragmentation.

The syntax for horizontal fragmentation is,

<FRAMESET ROWS=‖50%, 50%‖>

</FRAMESET>

**FORM:** The purpose of FORM is to create an HTML form, used to enclose HTML controls, like buttons and text fields.

**ATTRIBUTES:**

● ACTION: Gives the URL that will handle the form data.

● NAME: Gives the name to the form so you can reference it in code set to an alphanumeric string.

● METHOD: method or protocol is used to sending data to the target action URL. The GET method is the default, it is used to send all form name/value pair information in an URL. Using the POST method, the content of the form are encoded as with the GET method, but are sent in environment variables.

**CONTROLS IN HTML:**

➢ <INPUT TYPE =BUTTON>:

Creates an html button in a form.

**Attributes:**

● NAME: gives the element a name. Set to alphanumeric characters.

● SIZE: sets the size.

● VALUE: sets the caption of the element.

➢ <INPUT TYPE = PASSWORD>:

Creates a password text field, which makes typed input.

**Attributes:**

● NAME: gives the element a name, set to alphanumeric characters.

● VALUE: sets the default content of the element.

➢ <INPUT TYPE=RADIO>:

Creates a radio button in a form

. **Attributes:**

● NAME: Gives the element a name. Set to alphanumeric character.

● VALUE: Sets the default content of the element.

➢ <INPUT TYPE=SUBMIT>:

Creates a submit button that the user can click to send data in the form back to the web server.

**Attributes:**

● NAME: Gives the element a name. Set to alphanumeric characters.

● VALUE: Gives this button another label besides the default, Submit Query. Set to alphanumeric characters.

➢ <INPUT TYPE=TEXT>:

Creates a text field that the user can enter or edit text in

**Attributes:**

● NAME: Gives the element a name. Set to alphanumeric characters.

● VALUE: Gives this button another label besides the default, Submit Query. Set to alphanumeric characters.

**5.1.5 JAVA SCRIPT**

Java Script is Netscape’s cross–platform, object-based scripting language for client server application. JavaScript is mainly used as a client side scripting language. This means that JavaScript code is written into an HTML page. When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it's up to the browser to do something with it. JavaScript can be used in other contexts than a Web browser. Netscape created server-side JavaScript as a CGI-language that can do roughly the same as Perl or ASP.

Fortunately most browsers can handle JavaScript nowadays, but of course some browsers do not support some bits of script.

**Types of Java Script:**

a. Navigator Java Script also called client-side Java Script.

b. Live Wire Java Script also called server-side Java Script.

Using Java Script, dynamic HTML pages can be created that process user input and maintain persistent data using special objects, files and relational databases. Browser interprets JavaScript statements embedded in an HTML page. Netscape Navigator 2.0 and Internet Explorer 3.0 versions and later recognize Java Script. Through JavaScript Live Connect functionally, application can access Java and CORBA distributed-object applications. Navigator 3.0 and later versions supports Live Connect.

**Features of JavaScript (JS):**

1. Browser interprets JavaScript.
2. JavaScript is object based and uses built-in, extensible objects and have no classes or inheritance
3. JavaScript is loosely typed language
4. In JavaScript object reference are checked at runtime
5. JavaScript is designed to supplement the capabilities of HTML with script that are capable of responding to web pages events. JSP has access to some extent of aspects of the web browser window.
6. JavaScript control browser and content but cannot draw graphics or perform networking.

**Client side JavaScript features:**

Client–side JavaScript has expressly been developed for use in a web browser in conjunction with HTML pages. This has certain consequences for security.

* JavaScript cannot read files from or write them to the file system on the computer. This would be a clear security hazard
* JavaScript cannot execute any other programs. This would also be unacceptable.
* JavaScript cannot establish any connection to whatever computer, except to download a new HTML page or to send mail. This, too, would create unacceptable hazards.

The Client-Side JavaScript also has the following features:

* Controls Document’s appearance and content
* Control the browser
* Interact with the HTML forms
* Interact with the user
* Read and write client state with cookies

**Server- Side JavaScript Features:**

1. Embedded in HTML page
2. Executed at the server
3. Pre-complied for faster response
4. Access to Server-side objects
5. Encapsulation of the request

**5.2.1 Python Pandas**

Python Pandas is defined as an open-source library that provides high-performance data manipulation in Python. This tutorial is designed for both beginners and professionals.It is used for data analysis in Python and developed by Wes McKinney in 2008. Our Tutorial provides all the basic and advanced concepts of Python Pandas, such as Numpy, Data operation and Time Series

**Python Pandas Introduction**

Pandas is defined as an open-source library that provides high-performance data manipulation in Python. The name of Pandas is derived from the word Panel Data, which means an Econometrics from Multidimensional data. It is used for data analysis in Python and developed by Wes McKinney in 2008.

Data analysis requires lots of processing, such as restructuring, cleaning or merging, etc. There are different tools are available for fast data processing, such as Numpy, Scipy, Cython, and Panda. But we prefer Pandas because working with Pandas is fast, simple and more expressive than other tools.

Pandas is built on top of the Numpy package, means Numpy is required for operating the Pandas.

Before Pandas, Python was capable for data preparation, but it only provided limited support for data analysis. So, Pandas came into the picture and enhanced the capabilities of data analysis. It can perform five significant steps required for processing and analysis of data irrespective of the origin of the data, i.e., load, manipulate, prepare, model, and analyze.

**Key Features of Pandas**

* It has a fast and efficient DataFrame object with the default and customized indexing.
* Used for reshaping and pivoting of the data sets.
* Group by data for aggregations and transformations.
* It is used for data alignment and integration of the missing data.
* Provide the functionality of Time Series.
* Process a variety of data sets in different formats like matrix data, tabular heterogeneous, time series.
* Handle multiple operations of the data sets such as subsetting, slicing, filtering, groupBy, re-ordering, and re-shaping.
* It integrates with the other libraries such as SciPy, and scikit-learn.
* Provides fast performance, and If you want to speed it, even more, you can use the Cython.

**Benefits of Pandas**

The benefits of pandas over using other language are as follows:

* Data Representation: It represents the data in a form that is suited for data analysis through its DataFrame and Series.
* Clear code: The clear API of the Pandas allows you to focus on the core part of the code. So, it provides clear and concise code for the user.

**Python Pandas Data Structure**

The Pandas provides two data structures for processing the data, i.e., Series and DataFrame, which are discussed below:

**1) Series**

It is defined as a one-dimensional array that is capable of storing various data types. The row labels of series are called the index. We can easily convert the list, tuple, and dictionary into series using "series' method. A Series cannot contain multiple columns. It has one parameter:

Data: It can be any list, dictionary, or scalar value.

Creating Series from Array:

Before creating a Series, Firstly, we have to import the numpy module and then use array() function in the program.

import pandas as pd

import numpy as np

info = np.array(['P','a','n','d','a','s'])

a = pd.Series(info)

print(a)

Output

0 P

1 a

2 n

3 d

4 a

5 s

dtype: object

Explanation: In this code, firstly, we have imported the pandas and numpy library with the pd and np alias. Then, we have taken a variable named "info" that consist of an array of some values. We have called the info variable through a Series method and defined it in an "a" variable. The Series has printed by calling the print(a) method.

**Python Pandas DataFrame**

It is a widely used data structure of pandas and works with a two-dimensional array with labeled axes (rows and columns). DataFrame is defined as a standard way to store data and has two different indexes, i.e., row index and column index. It consists of the following properties:

The columns can be heterogeneous types like int, bool, and so on.

It can be seen as a dictionary of Series structure where both the rows and columns are indexed. It is denoted as "columns" in case of columns and "index" in case of rows.

Create a DataFrame using List:

We can easily create a DataFrame in Pandas using list.

import pandas as pd

# a list of strings

x = ['Python', 'Pandas']

# Calling DataFrame constructor on list

df = pd.DataFrame(x)

print(df)

Output

0

0 Python

1 Pandas

Explanation: In this code, we have defined a variable named "x" that consist of string values. The DataFrame constructor is being called on a list to print the values.

**Pandas DataFrame.describe()**

The describe() method is used for calculating some statistical data like percentile, mean and std of the numerical values of the Series or DataFrame. It analyzes both numeric and object series and also the DataFrame column sets of mixed data types.

Syntax

DataFrame.describe(percentiles=None, include=None, exclude=None)

Parameters

percentile: It is an optional parameter which is a list like data type of numbers that should fall between 0 and 1. Its default value is [.25, .5, .75], which returns the 25th, 50th, and 75th percentiles.

include: It is also an optional parameter that includes the list of the data types while describing the DataFrame. Its default value is None.

exclude: It is also an optional parameter that exclude the list of data types while describing DataFrame. Its default value is None.

Returns

It returns the statistical summary of the Series and DataFrame.

Example1

import pandas as pd

import numpy as np

a1 = pd.Series([1, 2, 3])

a1.describe()

Output

count 3.0

mean 2.0

std 1.0

min 1.0

25% 1.5

50% 2.0

75% 2.5

max 3.0

dtype: float64

**Pandas DataFrame.dropna()**

If your dataset consists of null values, we can use the dropna() function to analyze and drop the rows/columns in the dataset.

*Syntax:*

*DataFrameName.dropna(axis=0, how='any', thresh=None, subset=None, inplace=False)*

Parameters:

axis : {0 or 'index', 1 or 'columns'}, default value 0

It takes int or string values for rows/columns. The input can be 0 and 1 for the integers and index or columns for the string.

0, or 'index': Drop the rows which contain missing values.

1, or 'columns': Drop the columns which contain the missing value.

how :

It determines if row or column is removed from DataFrame when we have at least one NA or all NA.

It takes a string value of only two kinds ('any' or 'all').

any: It drops the row/column if any value is null.

all: It drops only if all values are null.

thresh:

It takes integer value that defines the minimum amount of NA values to drop.

subset:

It is an array that limits the dropping process to passed rows/columns through the list.

inplace:

It returns a boolean value that makes the changes in data frame itself if it is True.

Returns

It returns the DataFrame from which NA entries has been dropped.

For Demonstration, first, we are taking a csv file that will drop any column from the dataset.

**Pandas DataFrame.fillna()**

We can use the fillna() function to fill the null values in the dataset.

*Syntax:*

*DataFrame.fillna(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None, \*\*kwargs)*

Parameters:

value: It is a value that is used to fill the null values, alternately a Series/dict/DataFrame.

method: A method that is used to fill the null values in the reindexed Series.

axis: It takes int or string value for rows/columns. Axis along which we need to fill missing values.

inplace: If it is True, it fills values at an empty place.

limit: It is an integer value that specifies the maximum number of consecutive forward/backward NaN value fills.

downcast: It takes a dict that specifies what to downcast like Float64 to int64.

Returns:

It returns an object in which the missing values are being filled.

**Pandas NumPy**

Numerical Python (Numpy) is defined as a Python package used for performing the various numerical computations and processing of the multidimensional and single-dimensional array elements. The calculations using Numpy arrays are faster than the normal Python array.

This package is created by the Travis Oliphant in 2005 by adding the functionalities of the ancestor module Numeric into another module Numarray. It is also capable of handling a vast amount of data and convenient with Matrix multiplication and data reshaping.

NumPy is mostly written in C language, and it is an extension module of Python.

Pandas are built over numpy array; therefore, numpy helps us to use pandas more effectively.

Creating Arrays

The main task of arrays is to store multiple values in a single variable. It defines the multidimensional arrays that can be easily handled in numpy as shown in the below examples:

Example

# import the "array" for demonstrating array operations

import array

# initializing an array with array values and signed integers

arr = array.array('l', [2, 4, 6, 8, 10, 12])

# print the original array

print ("New created array: ",end="")

for l in range (0,5):

print (arr[l], end=" ")

print ("\r")

Output:

New created array: 2 4 6 8 10

Boolean indexing

Boolean indexing is defined as a vital tool of numpy, which is frequently used in pandas. Its main task is to use the actual values of the data in the DataFrame. We can filter the data in the boolean indexing in different ways that are as follows:

Access the DataFrame with a boolean index.

Apply the boolean mask to the DataFrame.

Masking the data based on column value.

Masking the data based on the index value.

Example1

This example shows how to access the DataFrame with a boolean index:

# importing pandas as pd

import pandas as pd

# dictionary of lists

dict = {'name':["Smith", "William", "Phill", "Parker"],

'age': ["28", "39", "34", "36"]}

info = pd.DataFrame(dict, index = [True, True, False, True])

print(info)

Output:

name age

True Smith 28

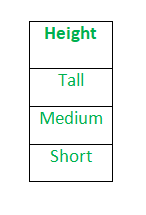
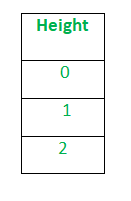
True William 39

False Phill 34

True Parker 36

Label Encoding

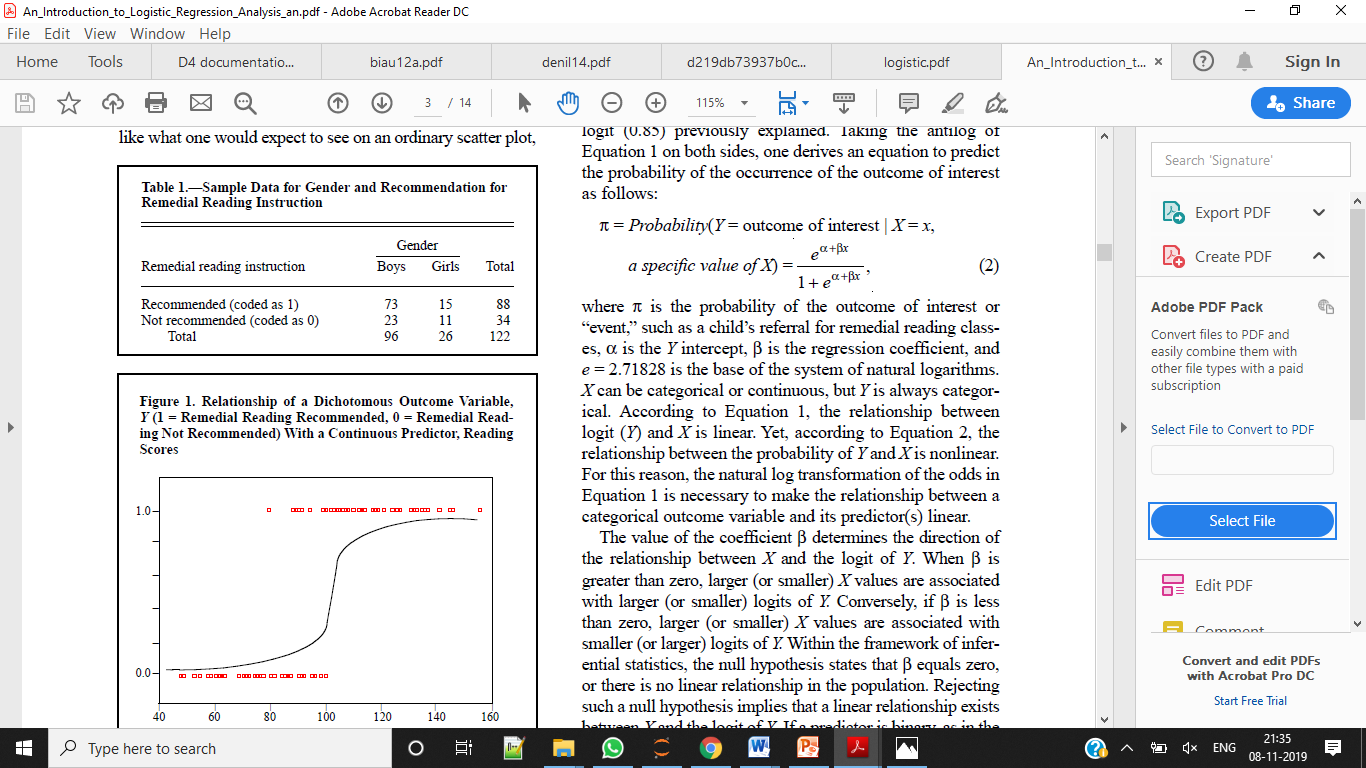
Label Encoding refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

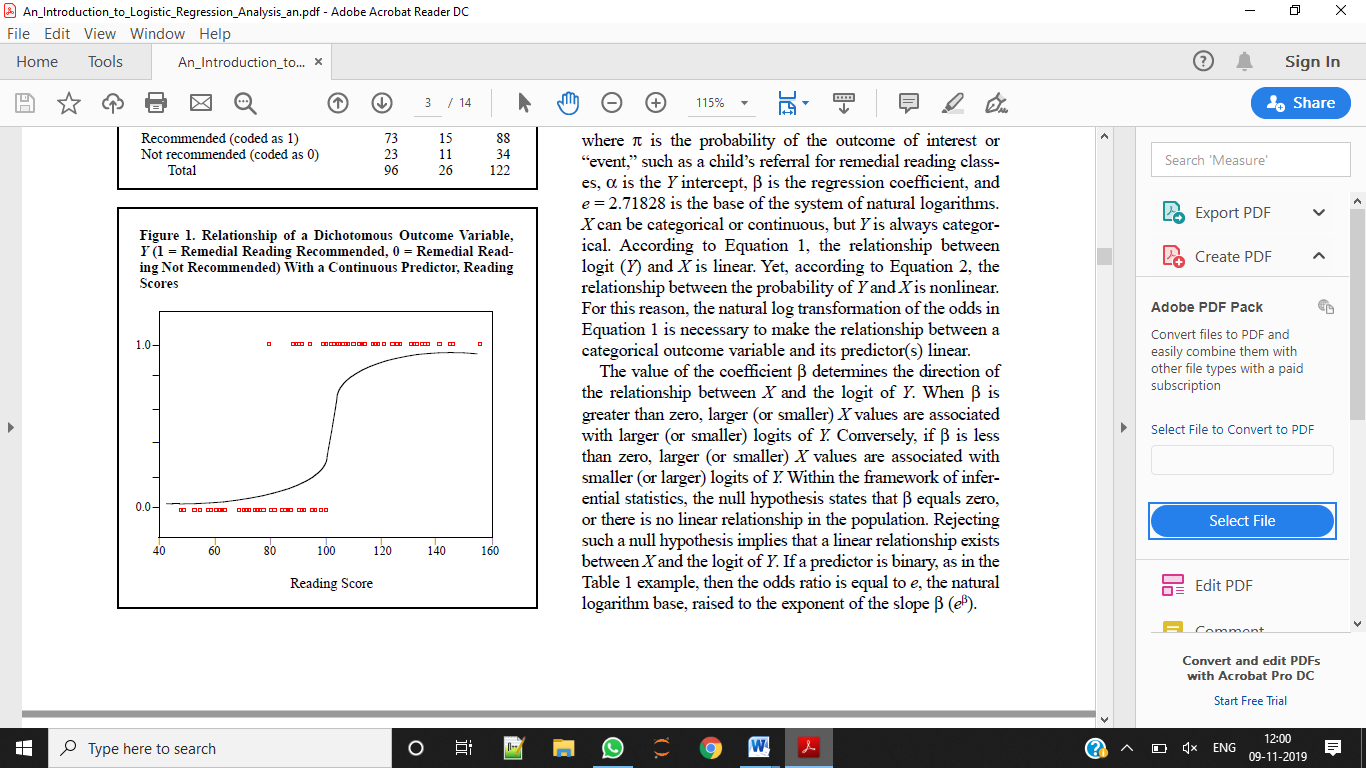
**Example :**  
Suppose we have a column*Height* in some dataset.  
  
After applying label encoding, the Height column is converted into:  
  
where 0 is the label for tall, 1 is the label for medium and 2 is label for short height.

**5.2.1 Logistic Regression**

The central mathematical concept that underlies logistic regression [5] is the logit—the natural logarithm of an odds ratio. The simplest example of a logit derives from a 2 × 2 contingency table. Consider an instance in which the distribution of a dichotomous outcome variable (a child from an inner city school who is recommended for remedial reading classes) is paired with a dichotomous predictor variable (gender). Example data are included in Table 1. A test of independence using chi-square could be applied. The results yield χ2(1) = 3.43. Alternatively, one might prefer to assessa boy’s odds of being recommended for remedial reading instruction relative to a girl’s odds. The result is an odds ratio of 2.33, which suggests that boys are 2.33 times more likely, than not, to be recommended for remedial reading classes compared with girls. The odds ratio is derived from two odds (73/23 for boys and 15/11 for girls); its natural logarithm [i.e., ln(2.33)] is a logit, which equals 0.85. The value of 0.85 would be the regression coefficient of the gender predictor if logistic regression were used to model the two outcomes of a remedial recommendation as it relates to gender.

Generally, logistic regression is well suited for describing and testing hypotheses about relationships between a categorical outcome variable and one or more categorical or continuous predictor variables. In the simplest case of linear regression for one continuous predictor X (a child’s reading score on a standardized test) and one dichotomous outcome variable Y (the child being recommended for remedial reading classes), the plot of such data results in two parallel lines, each corresponding to a value of the dichotomous outcome (Figure 1). Because the two parallel lines are difficult to be described with an ordinary least squares regression equation due to the dichotomy of outcomes, one may instead create categories for the predictor and compute the mean of the outcome variable for the respective categories. The resultant plot of categories’ means will appear linear in the middle, much like what one would expect to see on an ordinary scatter plot,

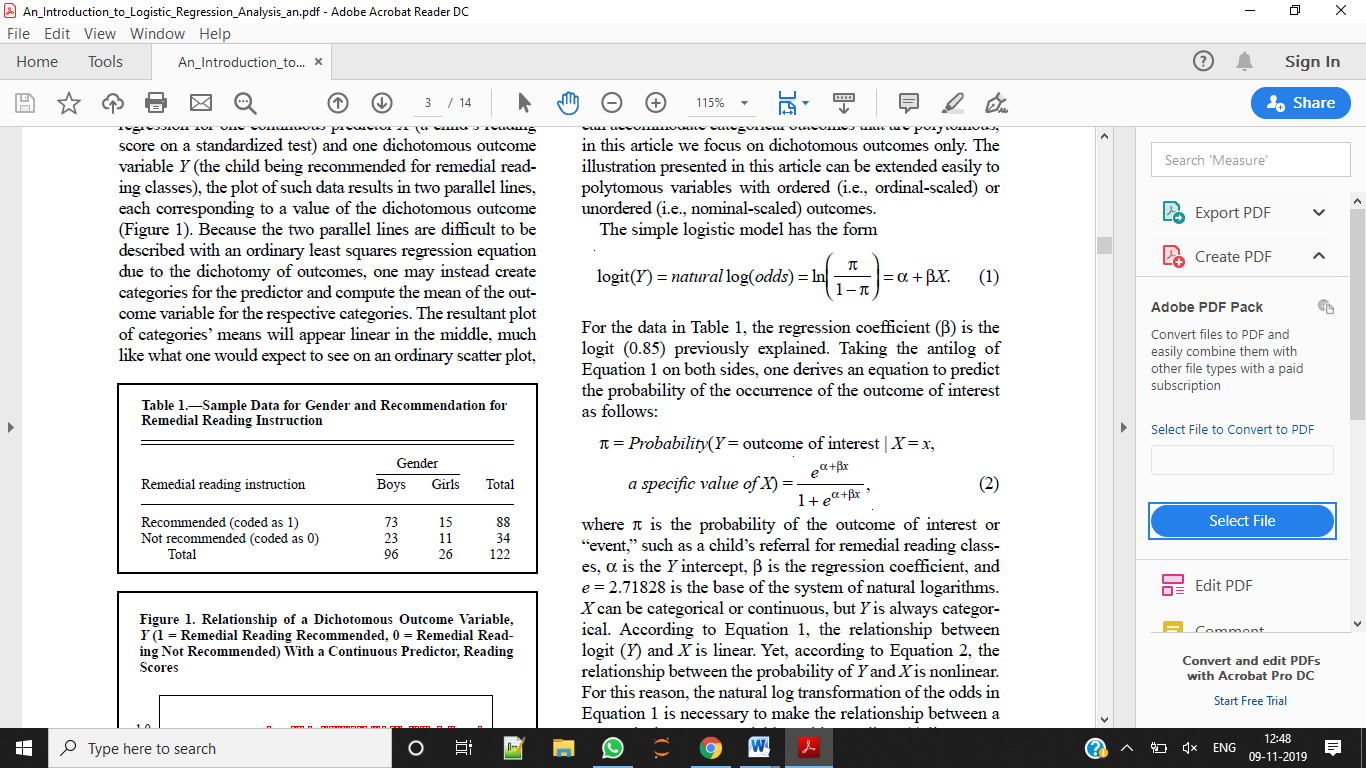


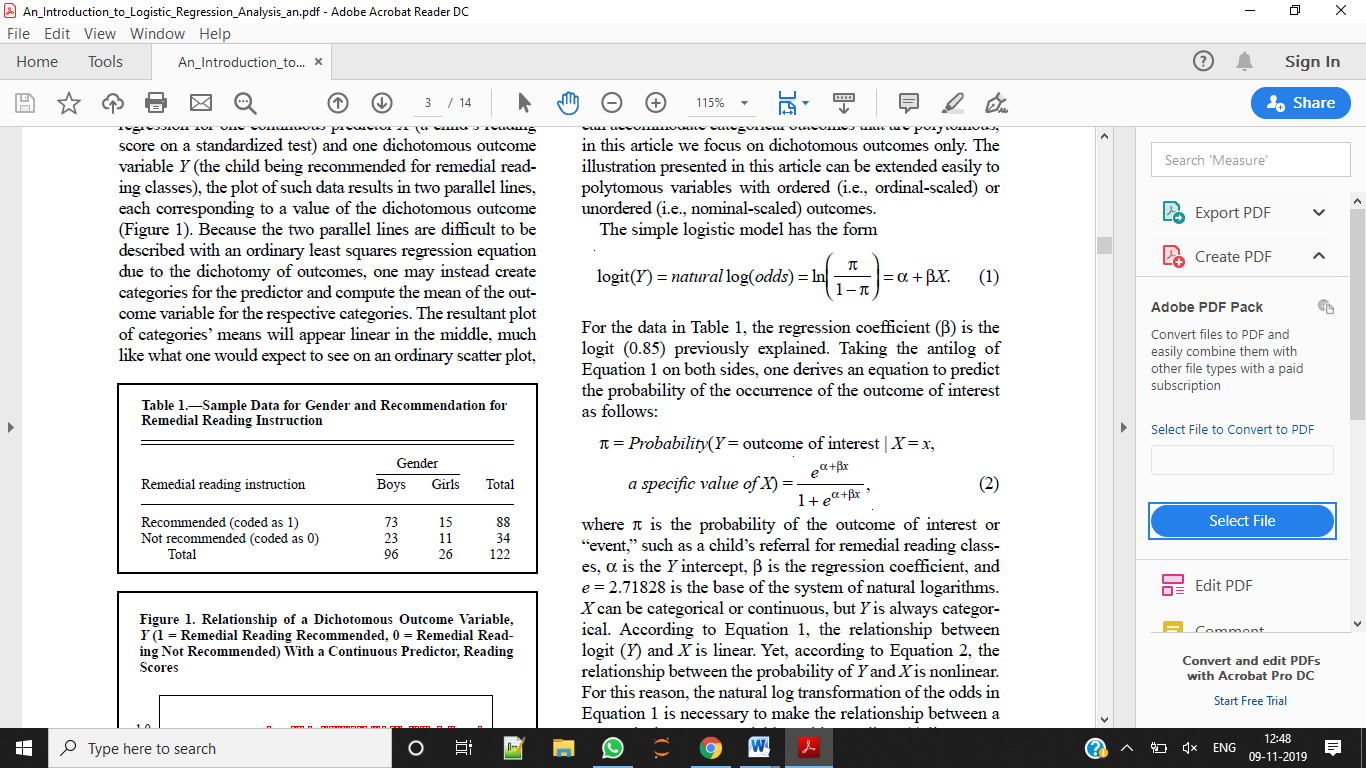


**Figure 5.8: Relationship of a Dichotomous Outcome Variable, *Y* (1 = Remedial Reading Recommended, 0 = Remedial Reading Not Recommended) With a Continuous Predictor, Reading Scores**

but curved at the ends (Figure 1, the S-shaped curve). Such a shape, often referred to as sigmoidal or S-shaped, is difficult to describe with a linear equation for two reasons. First, the extremes do not follow a linear trend. Second, the errors are neither normally distributed nor constant across the entire range of data (Peng, Manz, & Keck, 2001). Logistic regression solves these problems by applying the logit transformation to the dependent variable. In essence, the logistic model predicts the logit of Y from X. As stated earlier, the logit is the natural logarithm (ln) of odds of Y, and odds are ratios of probabilities (π) of Y happening (i.e., a student is recommended for remedial reading instruction) to probabilities (1 – π) of Y not happening (i.e., a student is not recommended for remedial reading instruction). Although logistic regression can accommodate categorical outcomes that are polytomous, in this article we focus on dichotomous outcomes only. The illustration presented in this article can be extended easily to polytomous variables with ordered (i.e., ordinal-scaled) or unordered (i.e., nominal-scaled) outcomes.

The simple logistic model has the form

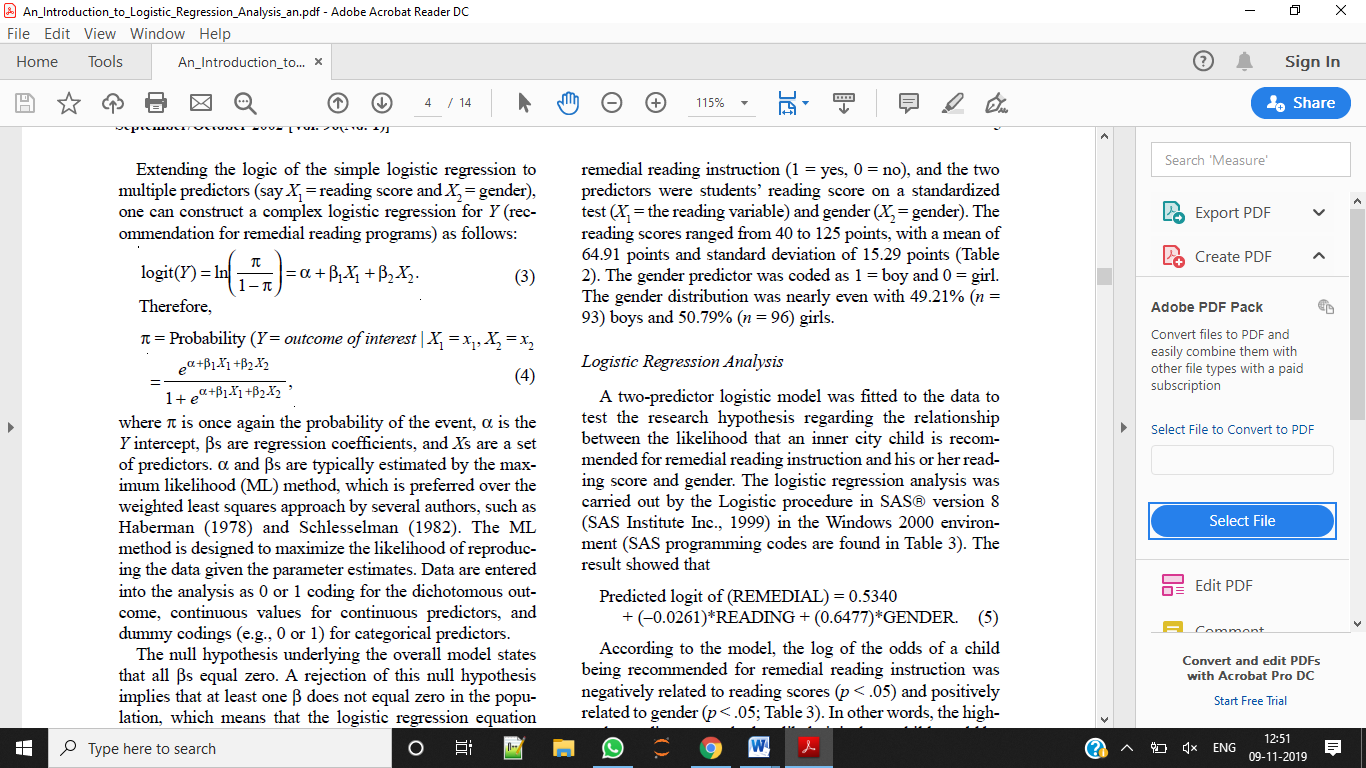
 For the data in Table 1, the regression coefficient (β) is the logit (0.85) previously explained. Taking the antilog of Equation 1 on both sides, one derives an equation to predict the probability of the occurrence of the outcome of interest as follows:

 where π is the probability of the outcome of interest or “event,” such as a child’s referral for remedial reading classes, α is the Y intercept, β is the regression coefficient, and e = 2.71828 is the base of the system of natural logarithms.

X can be categorical or continuous, but Y is always categorical. According to Equation 1, the relationship between logit (Y) and X is linear. Yet, according to Equation 2, the relationship between the probability of Y and X is nonlinear.

For this reason, the natural log transformation of the odds in Equation 1 is necessary to make the relationship between a categorical outcome variable and its predictor(s) linear. The value of the coefficient β determines the direction of the relationship between X and the logit of Y. When β is greater than zero, larger (or smaller) X values are associated with larger (or smaller) logits of Y. Conversely, if β is less than zero, larger (or smaller) X values are associated with smaller (or larger) logits of Y. Within the framework of inferential statistics, the null hypothesis states that β equals zero, or there is no linear relationship in the population. Rejecting such a null hypothesis implies that a linear relationship exists between X and the logit of Y. If a predictor is binary, as in the Table 1 example, then the odds ratio is equal to e, the natural logarithm base, raised to the exponent of the slope β (eβ).

Extending the logic of the simple logistic regression to multiple predictors (say X1 = reading score and X2 = gender), one can construct a complex logistic regression for Y (recommendation for remedial reading programs) as follows

 where π is once again the probability of the event, α is the Y intercept, βs are regression coefficients, and Xs are a set of predictors. α and βs are typically estimated by the maximum likelihood (ML) method, which is preferred over the weighted least squares approach by several authors, such as Haberman (1978) and Schlesselman (1982). The ML method is designed to maximize the likelihood of reproducing the data given the parameter estimates. Data are entered into the analysis as 0 or 1 coding for the dichotomous outcome, continuous values for continuous predictors, and dummy codings (e.g., 0 or 1) for categorical predictors. The null hypothesis underlying the overall model states that all βs equal zero. A rejection of this null hypothesis implies that at least one β does not equal zero in the population, which means that the logistic regression equation predicts the probability of the outcome better than the mean of the dependent variable Y. The interpretation of results is rendered using the odds ratio for both categorical and continuous predictors.

**5.2.2 Decision Tree**

It is flow-chart like tree structure, where each internal node denotes a test on an attribute, each branch denotes an outcome of test, and each leaf node holds a class label. The topmost node in a tree is the root node [14]. Given a tuple, X, for which the associated class label is unknown, the attribute values of the tuple are tested against decision tree. A path is traced from the root to a leaf node, which holds the class prediction for that tuple. Decision tree is useful because construction of decision tree classifiers does not require any domain knowledge. It can handle hidimensional data. The learning and classification steps of decision tree induction are simple and fast. Their representation of acquired knowledge in tree form is easy to assimilate by users. Decision tree classifiers have good accuracy [15].

Mining Classification Rules Every data classification project is different but the projects have some common features. Data classification requires some rules [16]. This classification rules are given below

* The data must be available
* The data must be relevant, adequate, and clean
* There must be a well-defined problem The problem should not be solvable by means of ordinary query
* The result must be actionable

Proposed Decision Tree Algorithm

The decision tree algorithm is a top-down induction algorithm. The aim of this algorithm is to build a tree that has leaves that are homogeneous as possible. The major step of this algorithm is to continue to divide leaves that are not homogeneous into leaves that are as homogeneous as possible. Steps of this algorithm are given below.

Input:

* Data partition, D, which is a set of training tuples and their associated class labels.
* Attribute\_list, the set of candidate attributes.
* Attribute\_selection\_method, a procedure to determine the splitting criterion that “best” partitions the data tuples into individual classes.

Output: A decision tree.

Decision Tree Rules

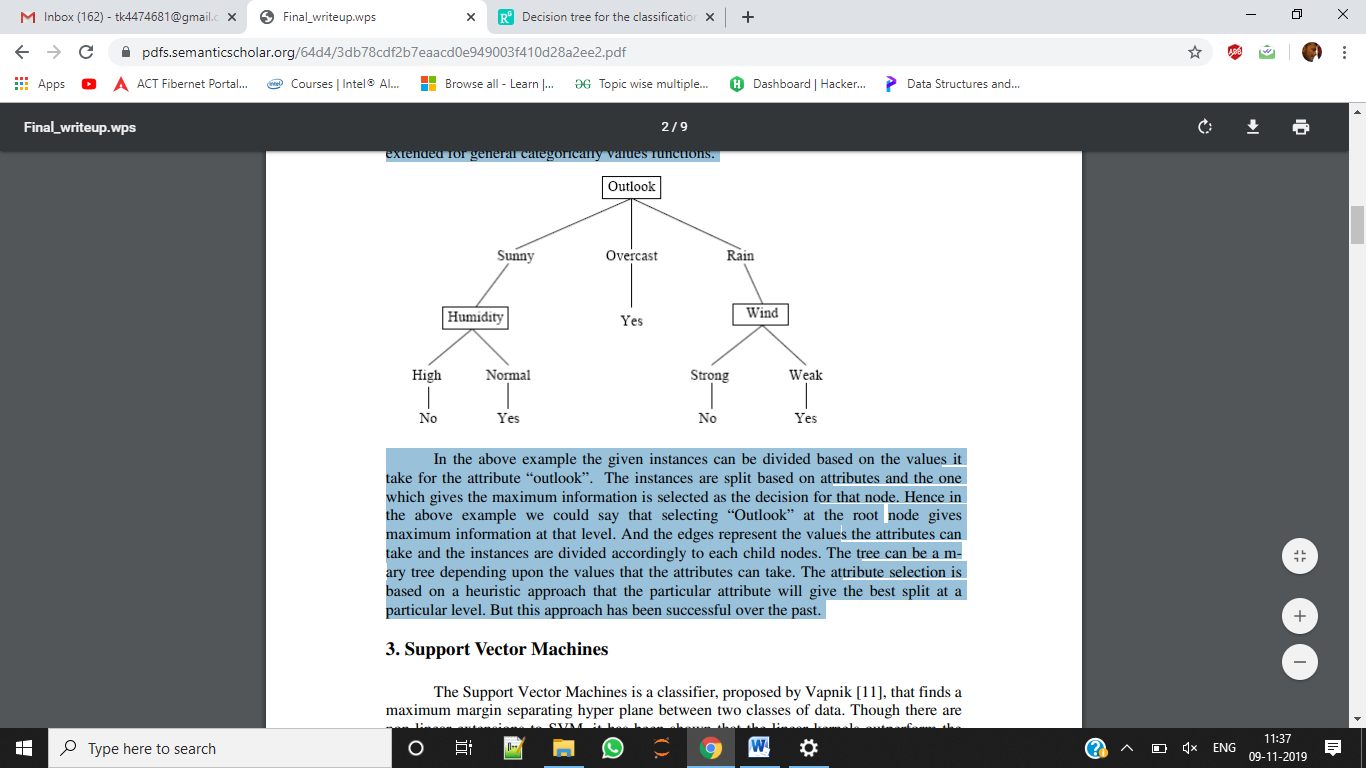
There are a number of advantages in converting a decision tree to rules. Decision tree make it easier to make pruning decisions. Since it is easier to see the context of each rule. Also, converting to rules removes the distinction between attribute tests that occur near the root of the tree and those that occur near the leaves. These rules are easier to read and to understand for people. The basic rules for decision tree are as below.

* Each path from the root to the leaf of the decision tree therefore consists of attribute tests, finally reaching a leaf that describes the class.
* If-then rules may be derived based on the various paths from the root to the leaf nodes.
* Rules can often be combined to produce a smaller set of rules. For example: If result = “distinction %” then credit rating = excellent
* If stream = “arts” and result = “70 %” then credit rating = average.
* Once all the rules have been generated, it may be possible to simplify the rules.
* Rules with only one antecedent (e.g. if result = “distinction”) can not be further simplified. So we only consider those with two or more antecedents.
* Eliminate unnecessary rule antecedents that have no effect on the conclusion reached by the rule.
* In some cases, a number of rules that lead to the same class may be combined.

Generation of Standard Decision Tree

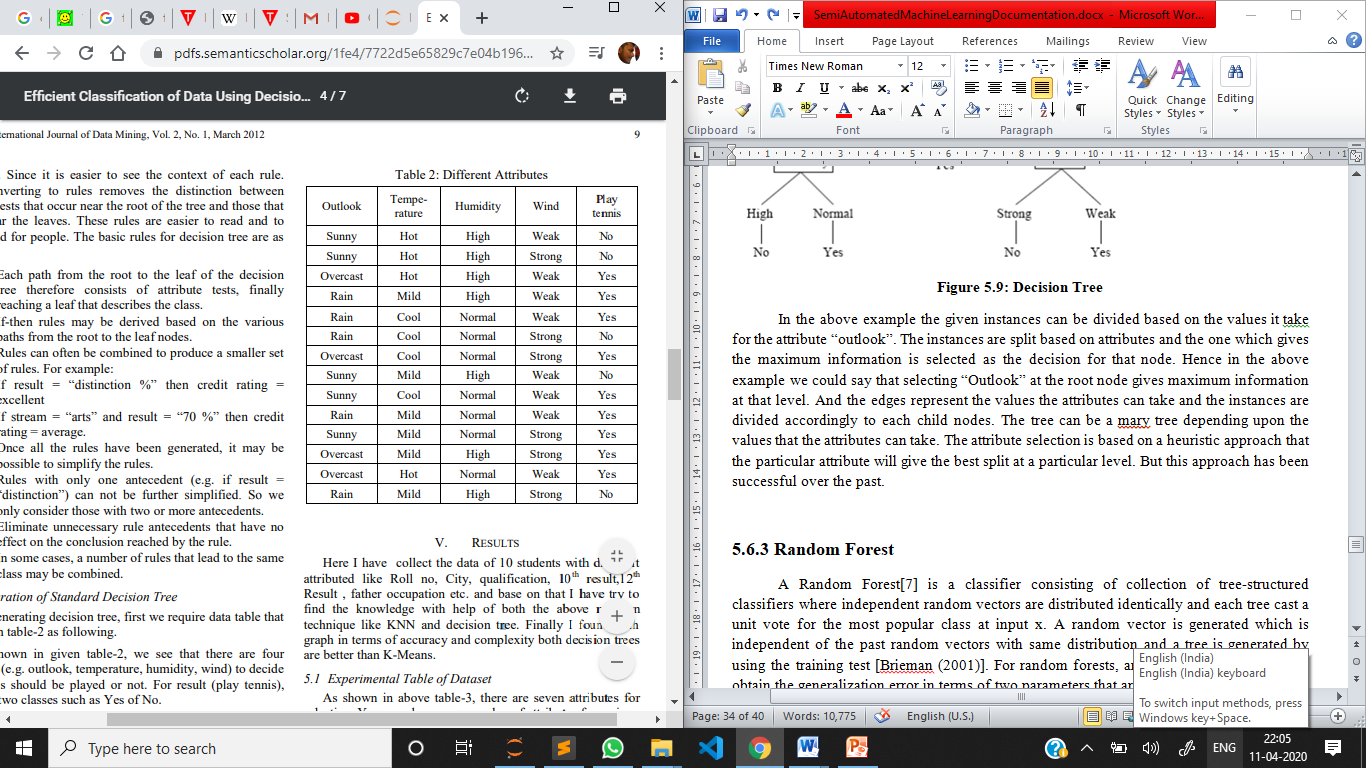
For generating decision tree, first we require data table that is given in table-2 as following.

As, shown in given table-2, we see that there are four attributes (e.g. outlook, temperature, humidity, wind) to decide that tennis should be played or not. For result (play tennis), there are two classes such as Yes of No.

These attributes may be increased or decreased. But if the numbers of attributes are more than data classification can be done with more accuracy. The decision tree for above data can be generated as shown in figure-1. 

**Figure 5.9: Decision Tree**

In the above example the given instances can be divided based on the values it take for the attribute “outlook”. The instances are split based on attributes and the one which gives the maximum information is selected as the decision for that node. Hence in the above example we could say that selecting “Outlook” at the root node gives maximum information at that level. And the edges represent the values the attributes can take and the instances are divided accordingly to each child nodes. The tree can be a mary tree depending upon the values that the attributes can take. The attribute selection is based on a heuristic approach that the particular attribute will give the best split at a particular level. But this approach has been successful over the past.



**5.2.3 Random Forest**

A Random Forest[7] is a classifier consisting of collection of tree-structured classifiers where independent random vectors are distributed identically and each tree cast a unit vote for the most popular class at input x. A random vector is generated which is independent of the past random vectors with same distribution and a tree is generated by using the training test [Brieman (2001)]. For random forests, an upper bound is derived to obtain the generalization error in terms of two parameters that are given below:

* The accuracy of individual classifiers
* The dependency between the individual classifiers

The generalization of error for random forest includes two segments. These segments are defined below:

* The strength of the individual classifiers in the forest.
* The correlation between them in terms of raw margin function

In order to improve accuracy of random forest, the correlation should be minimized while retaining their strength. Brieman (2001) studied that forests consist of randomly selected inputs or combination of inputs at each node to grow tree. The class of procedures has desirable characteristics that are listed below:

* Accuracy is good and sometimes better.
* Relatively robust to outliers and noise.
* Faster than Bagging and Boosting.
* Simple and can easily be parallelized.

Brieman has proposed a randomization approach that works better with bagging or random space method. To generate each tree of random forest, following steps are followed that are described below:

* Training dataset consist of N number of records.
* Samplings of N number of records are performed randomly but with replacement.
* This sample of dataset is named as bootstrap sample.
* If this training set would consist of M number of input variables, m<<M number of inputs are selected randomly out of M and the best split on these m attributes is used to split the node.
* The value of m will remain constant during forest growing.
* The tree will be grow to the largest possible level.

There are two reasons for using Bagging approach. They are given below: [Brieman (1994)]

* It seems that the use of bagging along with the random features generates more accurate results.
* Bagging can be used to provide on-going estimation of generalization error as well as the estimation of strength and correlation.

When the forest is trained to classify a new instance, this whole process is executed across all the trees included in the forest. Each tree of the forest has to provide a vote which is recorded as a classification for the new instance. The votes of all trees are combined together and the votes are counted, the class having maximum number of votes are declared as classification of new instance. This process is known as Forest RI process. The description of process for building forest is given below:

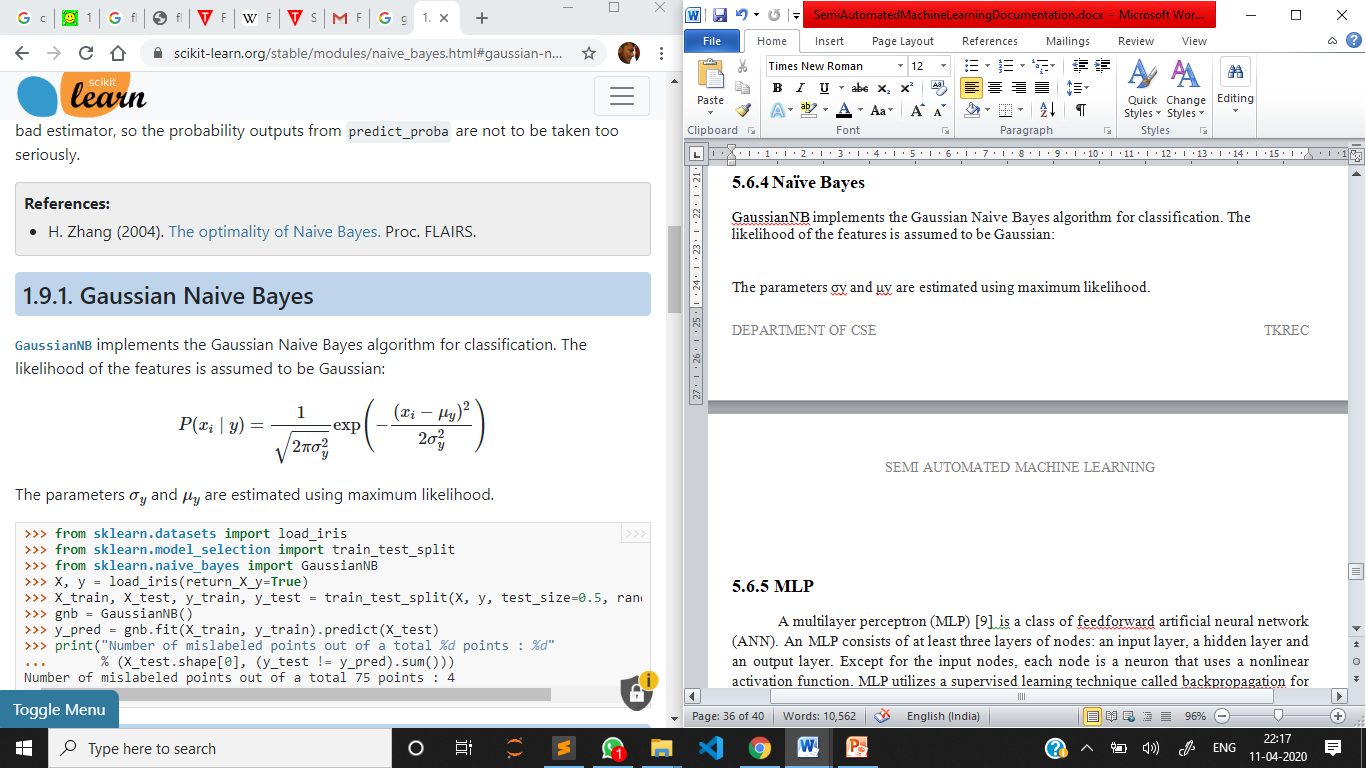
* When bootstrap sample is built by sampling the data with replacement of each tree, then one-third of the instances are left out.
* The left out instances are known as OOB (Out of Bag) data.
* Each tree of the forest has its own OOB data which is used for the error estimation of individual trees known as OOB error estimation.
* Random forest also consist of in-built facilities for calculating variable importance and proximities.
* The proximities are used for removing and replacing missing values and outliers.



**Figure 5.10: Random Forest**

**5.2.4 Naïve Bayes**

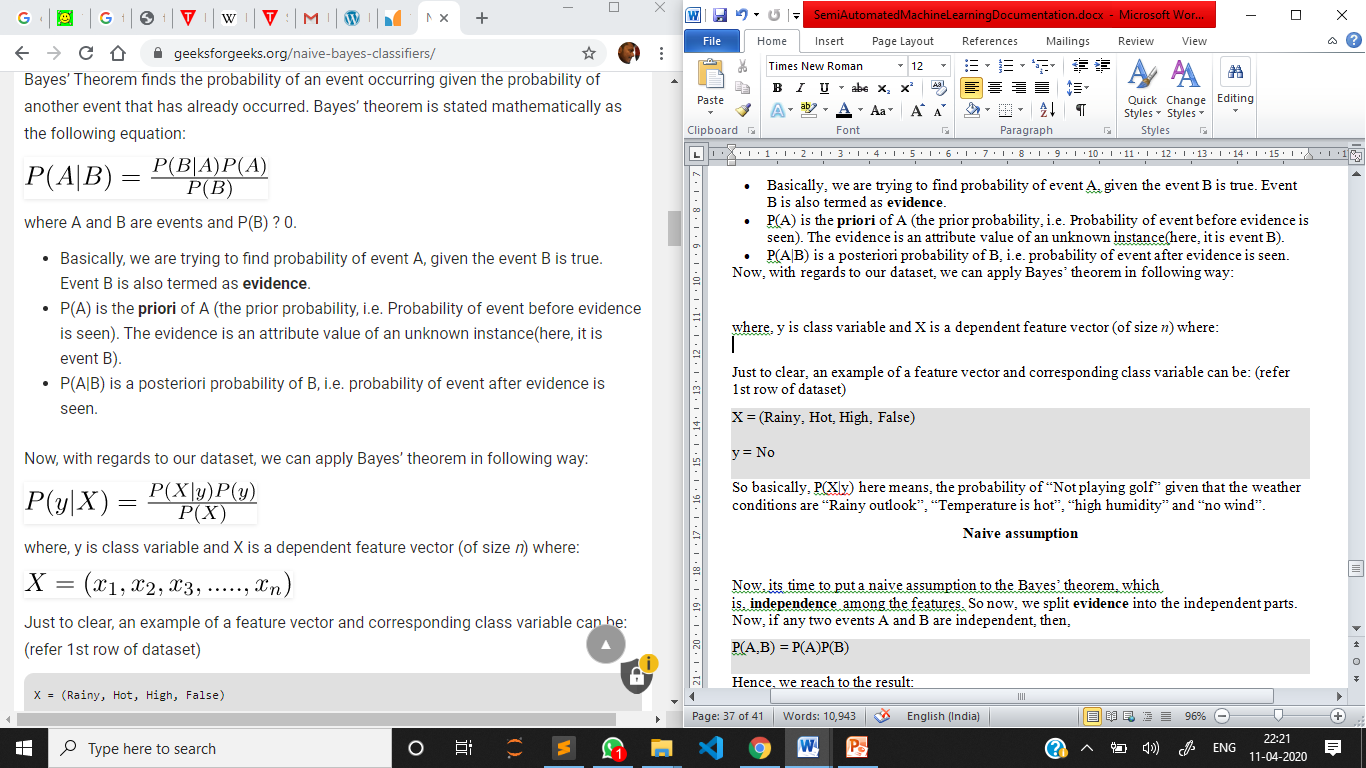
[GaussianNB](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB) implements the Gaussian Naive Bayes algorithm for classification. The likelihood of the features is assumed to be Gaussian:



The parameters σy and μy are estimated using maximum likelihood.

**Bayes’ Theorem**

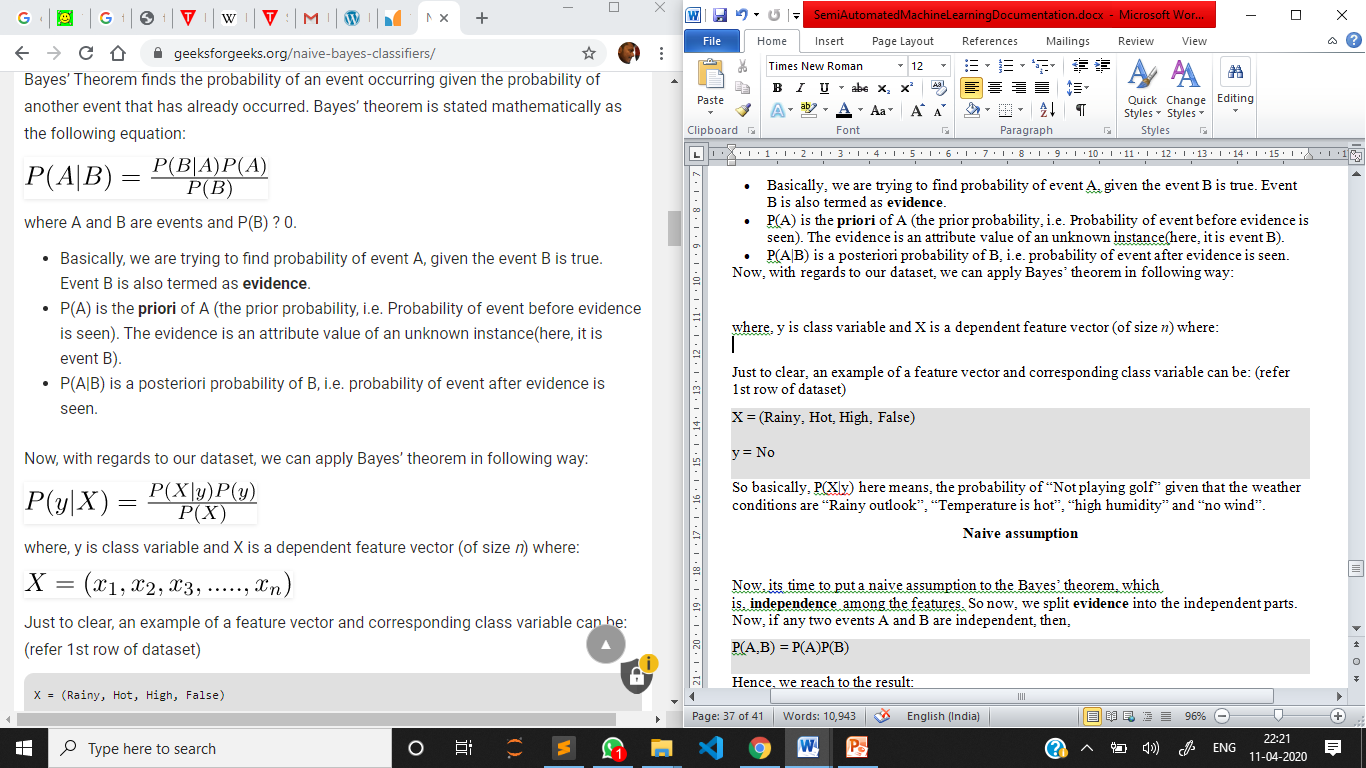
Bayes’ Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes’ theorem is stated mathematically as the following equation:



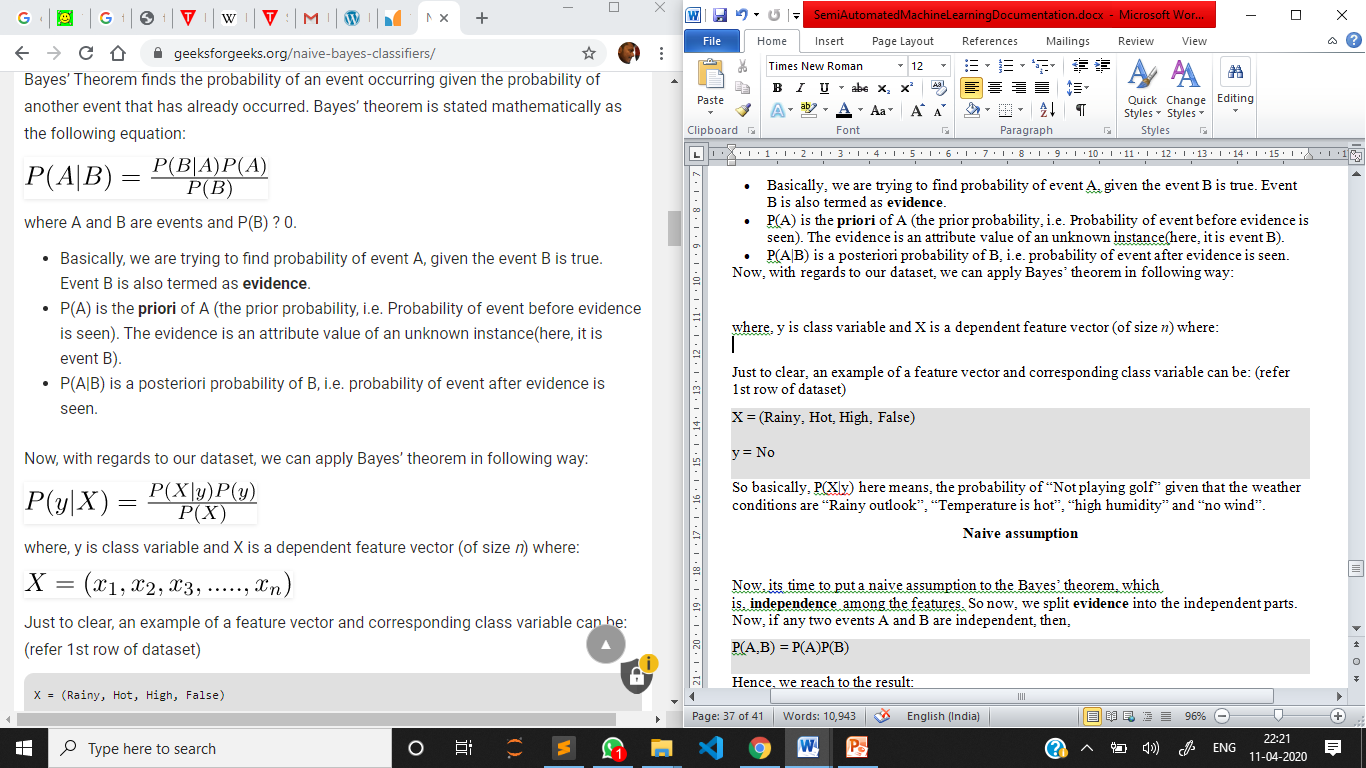
where A and B are events and P(B) ? 0.

* Basically, we are trying to find probability of event A, given the event B is true. Event B is also termed as **evidence**.
* P(A) is the **priori** of A (the prior probability, i.e. Probability of event before evidence is seen). The evidence is an attribute value of an unknown instance(here, it is event B).
* P(A|B) is a posteriori probability of B, i.e. probability of event after evidence is seen.

Now, with regards to our dataset, we can apply Bayes’ theorem in following way:



where, y is class variable and X is a dependent feature vector (of size *n*) where:



Just to clear, an example of a feature vector and corresponding class variable can be: (refer 1st row of dataset)

X = (Rainy, Hot, High, False)

y = No

So basically, P(X|y) here means, the probability of “Not playing golf” given that the weather conditions are “Rainy outlook”, “Temperature is hot”, “high humidity” and “no wind”.

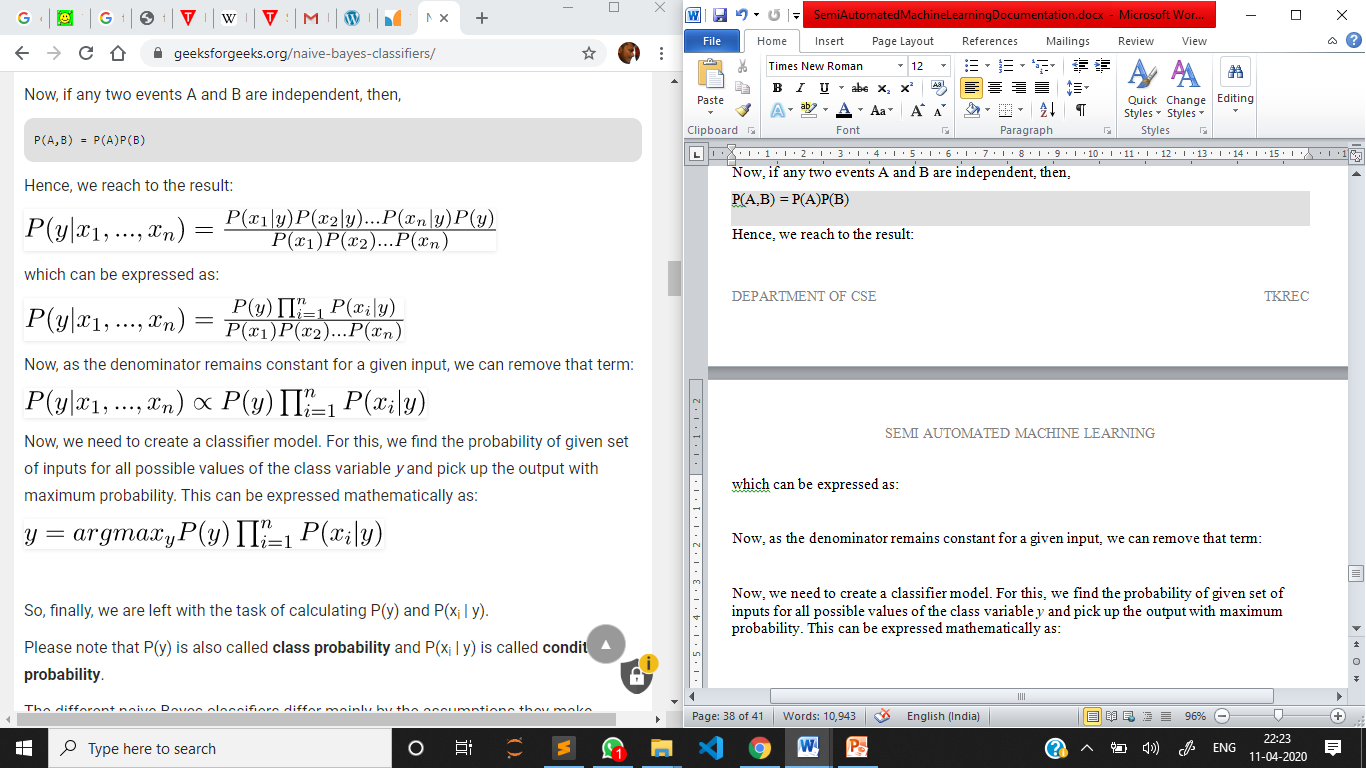
**Naive assumption**

Now, its time to put a naive assumption to the Bayes’ theorem, which is, **independence** among the features. So now, we split **evidence** into the independent parts.

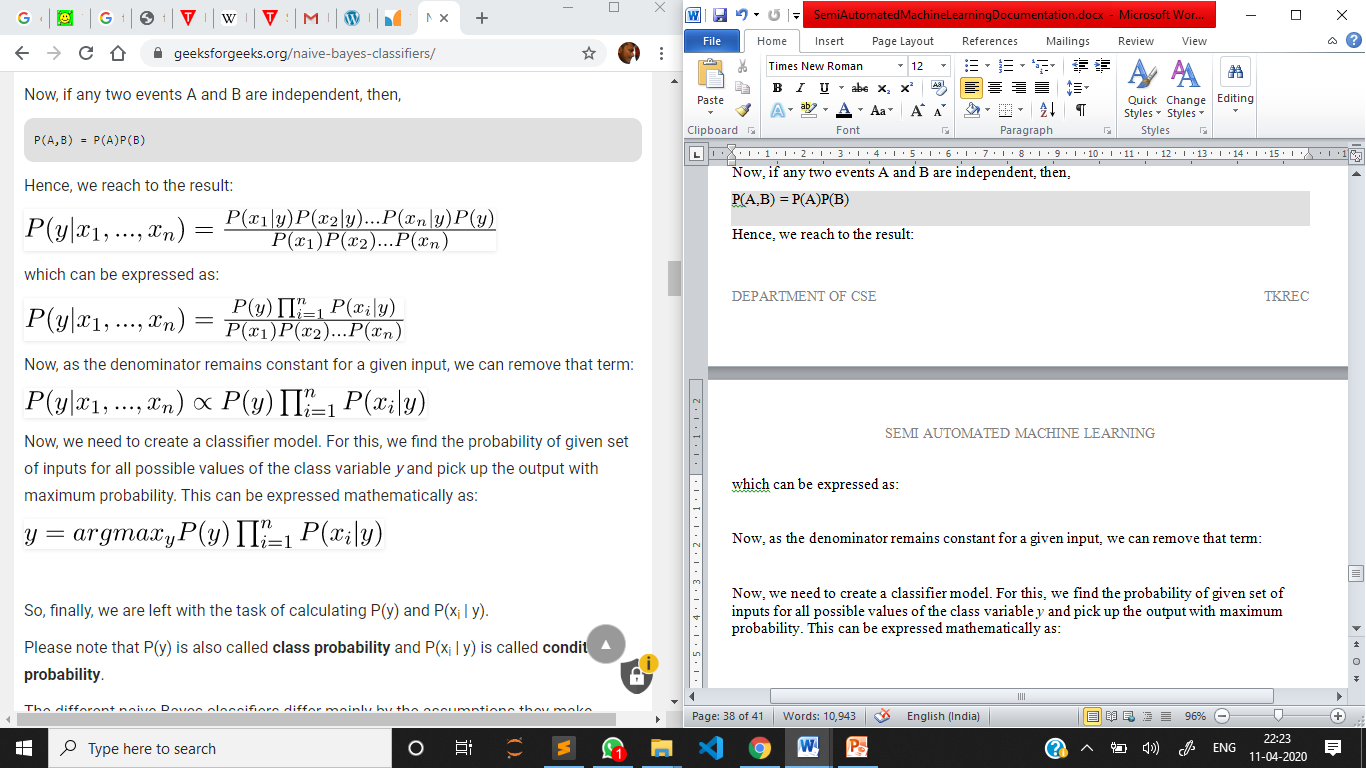
Now, if any two events A and B are independent, then,

P(A,B) = P(A)P(B)

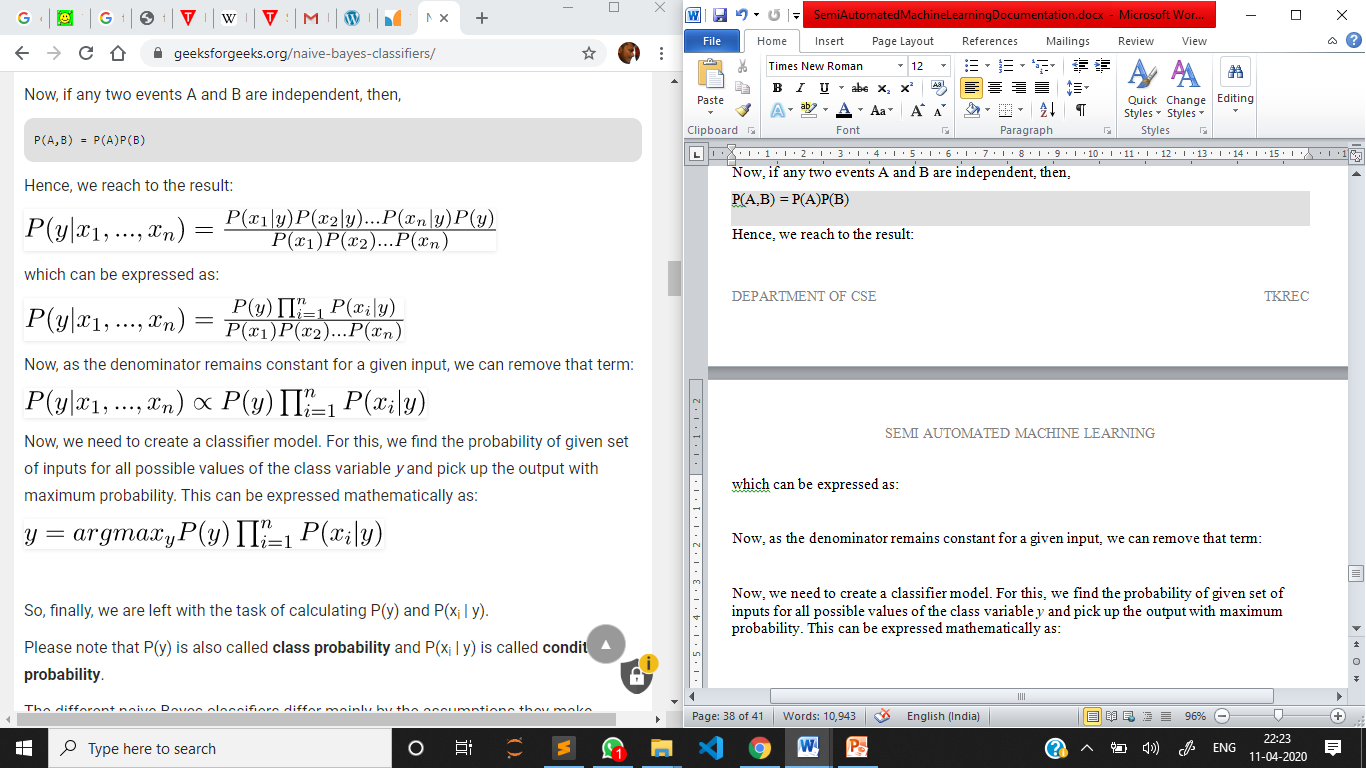
Hence, we reach to the result:



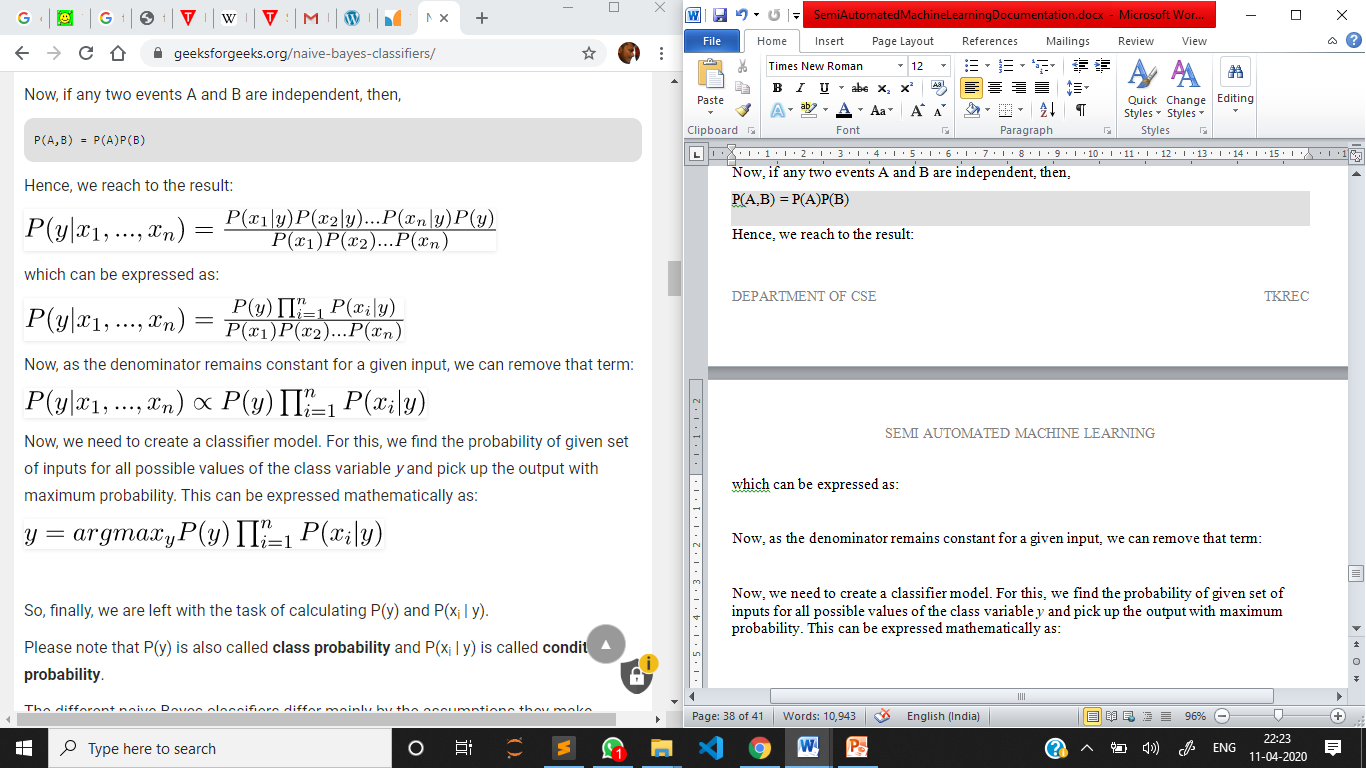
which can be expressed as:



Now, as the denominator remains constant for a given input, we can remove that term:



Now, we need to create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable *y* and pick up the output with maximum probability. This can be expressed mathematically as:



So, finally, we are left with the task of calculating P(y) and P(xi | y).

Please note that P(y) is also called **class probability** and P(xi | y) is called **conditional probability**.

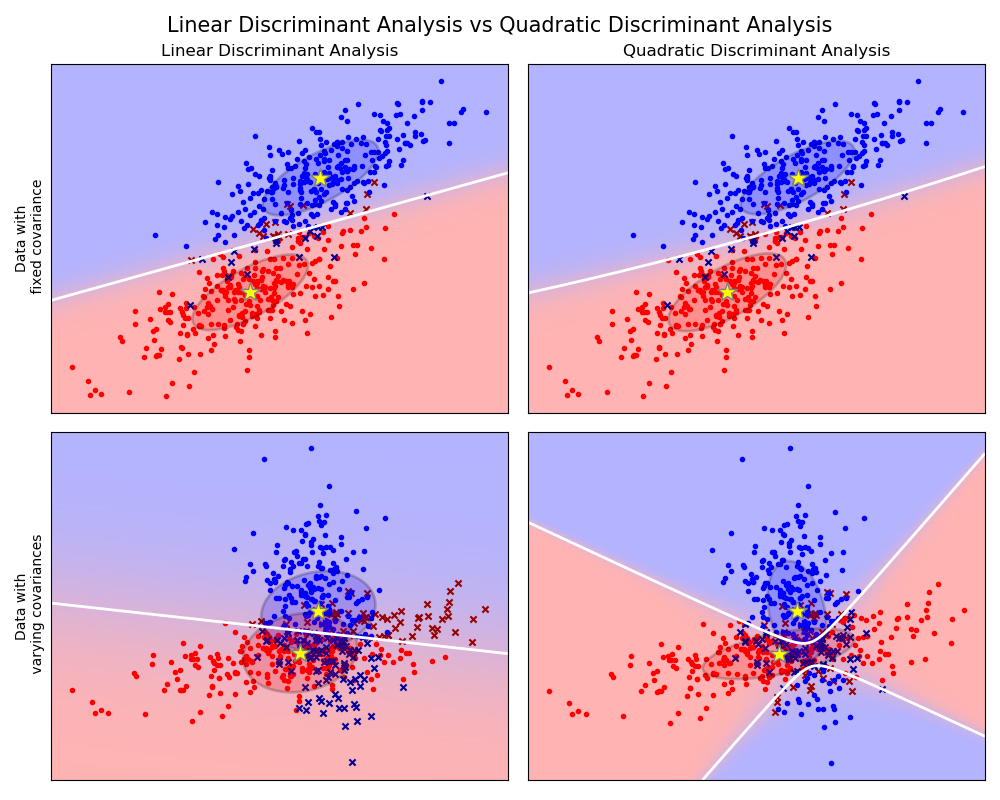
The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of P(xi | y).

Let us try to apply the above formula manually on our weather dataset. For this, we need to do some precomputations on our dataset.

**5.2.5 Linear Discriminant Analysis**

Linear Discriminant Analysis (**[discriminant\_analysis.LinearDiscriminantAnalysis](https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html" \l "sklearn.discriminant_analysis.LinearDiscriminantAnalysis" \o "sklearn.discriminant_analysis.LinearDiscriminantAnalysis)**) and Quadratic Discriminant Analysis (**[discriminant\_analysis.QuadraticDiscriminantAnalysis](https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis.html" \l "sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis" \o "sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis)**) are two classic classifiers, with, as their names suggest, a linear and a quadratic decision surface, respectively.

These classifiers are attractive because they have closed-form solutions that can be easily computed, are inherently multiclass, have proven to work well in practice, and have no hyperparameters to tune.

**[](https://scikit-learn.org/stable/auto_examples/classification/plot_lda_qda.html)**

The plot shows decision boundaries for Linear Discriminant Analysis and Quadratic Discriminant Analysis. The bottom row demonstrates that Linear Discriminant Analysis can only learn linear boundaries, while Quadratic Discriminant Analysis can learn quadratic boundaries and is therefore more flexible.

Both LDA and QDA can be derived from simple probabilistic models which model the class conditional distribution of the data P(X|y=k) for each class k. Predictions can then be obtained by using Bayes’ rule:

*P(y=k|X)=P(X|y=k)P(y=k)P(X)=P(X|y=k)P(y=k)∑lP(X|y=l)⋅P(y=l)*

and we select the class k which maximizes this conditional probability.

More specifically, for linear and quadratic discriminant analysis, P(X|y) is modeled as a multivariate Gaussian distribution with density:

*P(X|y=k)=1(2π)d/2|Σk|1/2exp⁡(−12(X−μk)tΣk−1(X−μk))*

where d is the number of features.

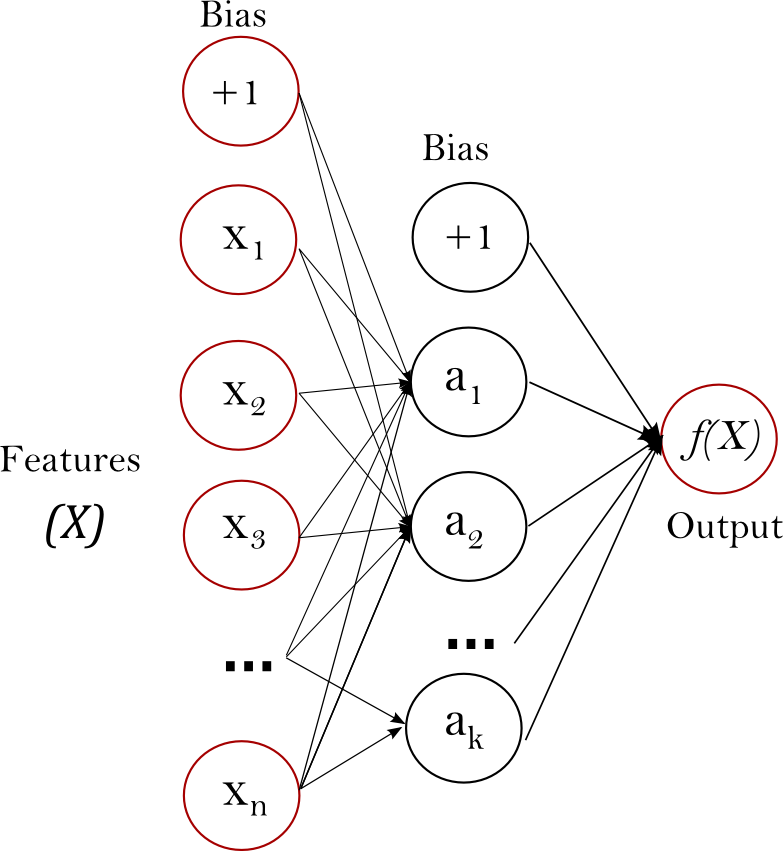
To use this model as a classifier, we just need to estimate from the training data the class priors P(y=k) (by the proportion of instances of class k), the class means μk (by the empirical sample class means) and the covariance matrices (either by the empirical sample class covariance matrices, or by a regularized estimator: see the section on shrinkage below).

In the case of LDA, the Gaussians for each class are assumed to share the same covariance matrix: Σk=Σ for all k. This leads to linear decision surfaces, which can be seen by comparing the log-probability ratios *log⁡[P(y=k|X)/P(y=l|X)]:*

*log⁡(P(y=k|X)P(y=l|X))=log⁡(P(X|y=k)P(y=k)P(X|y=l)P(y=l))=0⇔(μk−μl)tΣ−1X=12(μktΣ−1μk−μltΣ−1μl)−log⁡P(y=k)P(y=l)*

**5.2.6 MLP**

**Multi-layer Perceptron (MLP)** is a supervised learning algorithm that learns a function f(⋅):Rm→Ro by training on a dataset, where m is the number of dimensions for input and o is the number of dimensions for output. Given a set of features X=x1,x2,...,xm and a target y, it can learn a non-linear function approximator for either classification or regression. It is different from logistic regression, in that between the input and the output layer, there can be one or more non-linear layers, called hidden layers. Figure 1 shows a one hidden layer MLP with scalar output.

[](https://scikit-learn.org/stable/_images/multilayerperceptron_network.png)

**Figure 1 : One hidden layer MLP.**

The leftmost layer, known as the input layer, consists of a set of neurons {xi|x1,x2,...,xm} representing the input features. Each neuron in the hidden layer transforms the values from the previous layer with a weighted linear summation w1x1+w2x2+...+wmxm, followed by a non-linear activation function g(⋅):R→R - like the hyperbolic tan function. The output layer receives the values from the last hidden layer and transforms them into output values.

The module contains the public attributes coefs\_ and intercepts\_. coefs\_ is a list of weight matrices, where weight matrix at index i represents the weights between layer i and layer i+1. intercepts\_ is a list of bias vectors, where the vector at index i represents the bias values added to layer i+1.

The advantages of Multi-layer Perceptron are:

* Capability to learn non-linear models.
* Capability to learn models in real-time (on-line learning) using partial\_fit.

The disadvantages of Multi-layer Perceptron (MLP) include:

* MLP with hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore different random weight initializations can lead to different validation accuracy.
* MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations.
* MLP is sensitive to feature scaling.

**5.2.7 KNN**

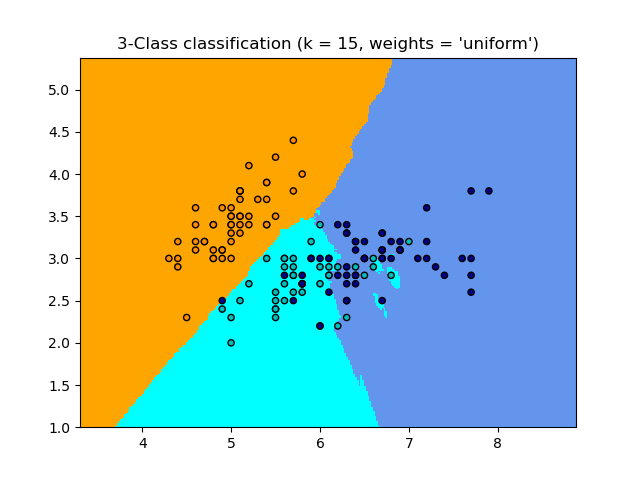
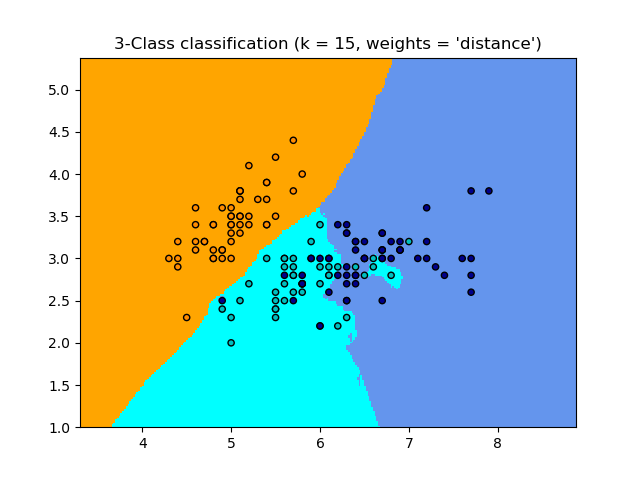
Neighbors-based classification is a type of *instance-based learning* or *non-generalizing learning*: it does not attempt to construct a general internal model, but simply stores instances of the training data. Classification is computed from a simple majority vote of the nearest neighbors of each point: a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

scikit-learn implements two different nearest neighbors classifiers: **[KNeighborsClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html" \l "sklearn.neighbors.KNeighborsClassifier" \o "sklearn.neighbors.KNeighborsClassifier)** implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user. **[RadiusNeighborsClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.RadiusNeighborsClassifier.html" \l "sklearn.neighbors.RadiusNeighborsClassifier" \o "sklearn.neighbors.RadiusNeighborsClassifier)** implements learning based on the number of neighbors within a fixed radius r of each training point, where r is a floating-point value specified by the user.

The k-neighbors classification in **[KNeighborsClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html" \l "sklearn.neighbors.KNeighborsClassifier" \o "sklearn.neighbors.KNeighborsClassifier)** is the most commonly used technique. The optimal choice of the value k is highly data-dependent: in general a larger k suppresses the effects of noise, but makes the classification boundaries less distinct.

In cases where the data is not uniformly sampled, radius-based neighbors classification in **[RadiusNeighborsClassifier](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.RadiusNeighborsClassifier.html" \l "sklearn.neighbors.RadiusNeighborsClassifier" \o "sklearn.neighbors.RadiusNeighborsClassifier)** can be a better choice. The user specifies a fixed radius r, such that points in sparser neighborhoods use fewer nearest neighbors for the classification. For high-dimensional parameter spaces, this method becomes less effective due to the so-called “curse of dimensionality”.

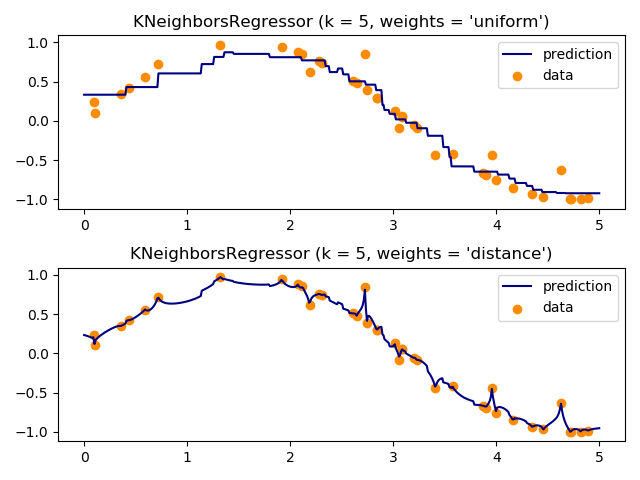
The basic nearest neighbors classification uses uniform weights: that is, the value assigned to a query point is computed from a simple majority vote of the nearest neighbors. Under some circumstances, it is better to weight the neighbors such that nearer neighbors contribute more to the fit. This can be accomplished through the weights keyword. The default value, weights = 'uniform', assigns uniform weights to each neighbor. weights = 'distance' assigns weights proportional to the inverse of the distance from the query point. Alternatively, a user-defined function of the distance can be supplied to compute the weights.

**[](https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html) [](https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html)**

Neighbors-based regression can be used in cases where the data labels are continuous rather than discrete variables. The label assigned to a query point is computed based on the mean of the labels of its nearest neighbors.

scikit-learn implements two different neighbors regressors: **[KNeighborsRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html" \l "sklearn.neighbors.KNeighborsRegressor" \o "sklearn.neighbors.KNeighborsRegressor)** implements learning based on the k nearest neighbors of each query point, where k is an integer value specified by the user. **[RadiusNeighborsRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.RadiusNeighborsRegressor.html" \l "sklearn.neighbors.RadiusNeighborsRegressor" \o "sklearn.neighbors.RadiusNeighborsRegressor)** implements learning based on the neighbors within a fixed radius r of the query point, where r is a floating-point value specified by the user.

The basic nearest neighbors regression uses uniform weights: that is, each point in the local neighborhood contributes uniformly to the classification of a query point. Under some circumstances, it can be advantageous to weight points such that nearby points contribute more to the regression than faraway points. This can be accomplished through the weights keyword. The default value, weights = 'uniform', assigns equal weights to all points. weights = 'distance' assigns weights proportional to the inverse of the distance from the query point. Alternatively, a user-defined function of the distance can be supplied, which will be used to compute the weights.

[](https://scikit-learn.org/stable/auto_examples/neighbors/plot_regression.html)

The use of multi-output nearest neighbors for regression is demonstrated in [Face completion with a multi-output estimators](https://scikit-learn.org/stable/auto_examples/plot_multioutput_face_completion.html#sphx-glr-auto-examples-plot-multioutput-face-completion-py). In this example, the inputs X are the pixels of the upper half of faces and the outputs Y are the pixels of the lower half of those faces.

**5.3 Cross validation**

**Validation**

This process of deciding whether the numerical results quantifying hypothesized relationships between variables, are acceptable as descriptions of the data, is known as validation. Generally, an error estimation for the model is made after training, better known as evaluation of residuals. In this process, a numerical estimate of the difference in predicted and original responses is done, also called the training error. However, this only gives us an idea about how well our model does on data used to train it. Now its possible that the model is underfitting or overfitting the data. So, the problem with this evaluation technique is that it does not give an indication of how well the learner will generalize to an independent/ unseen data set. Getting this idea about our model is known as Cross Validation.

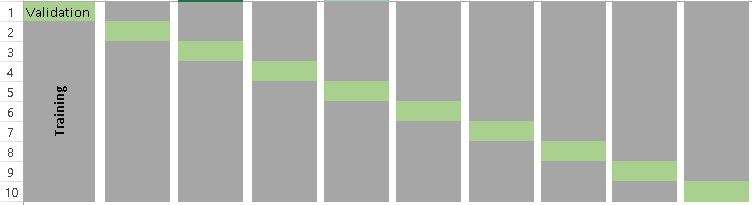
**Holdout Method**

Now a basic remedy for this involves removing a part of the training data and using it to get predictions from the model trained on rest of the data. The error estimation then tells how our model is doing on unseen data or the validation set. This is a simple kind of cross validation technique, also known as the holdout method. Although this method doesn’t take any overhead to compute and is better than traditional validation, it still suffers from issues of high variance. This is because it is not certain which data points will end up in the validation set and the result might be entirely different for different sets.

**K-Fold Cross Validation**

As there is never enough data to train your model, removing a part of it for validation poses a problem of underfitting. By reducing the training data, we risk losing important patterns/ trends in data set, which in turn increases error induced by bias. So, what we require is a method that provides ample data for training the model and also leaves ample data for validation. K Fold cross validation does exactly that.

In K Fold cross validation, the data is divided into k subsets. Now the holdout method is repeated k times, such that each time, one of the k subsets is used as the test set/ validation set and the other k-1 subsets are put together to form a training set. The error estimation is averaged over all k trials to get total effectiveness of our model. As can be seen, every data point gets to be in a validation set exactly once, and gets to be in a training set k-1 times. This significantly reduces bias as we are using most of the data for fitting, and also significantly reduces variance as most of the data is also being used in validation set. Interchanging the training and test sets also adds to the effectiveness of this method. As a general rule and empirical evidence, K = 5 or 10 is generally preferred, but nothing’s fixed and it can take any value.



**Stratified K-Fold Cross Validation**

In some cases, there may be a large imbalance in the response variables. For example, in dataset concerning price of houses, there might be large number of houses having high price. Or in case of classification, there might be several times more negative samples than positive samples. For such problems, a slight variation in the K Fold cross validation technique is made, such that each fold contains approximately the same percentage of samples of each target class as the complete set, or in case of prediction problems, the mean response value is approximately equal in all the folds. This variation is also known as Stratified K Fold.

Above explained validation techniques are also referred to as Non-exhaustive cross validation methods. These do not compute all ways of splitting the original sample, i.e. you just have to decide how many subsets need to be made. Also, these are approximations of method explained below, also called Exhaustive Methods, that computes all possible ways the data can be split into training and test sets.



**Leave-P-Out Cross Validation**

This approach leaves p data points out of training data, i.e. if there are n data points in the original sample then, n-p samples are used to train the model and p points are used as the validation set. This is repeated for all combinations in which original sample can be separated this way, and then the error is averaged for all trials, to give overall effectiveness.

This method is exhaustive in the sense that it needs to train and validate the model for all possible combinations, and for moderately large p, it can become computationally infeasible.

A particular case of this method is when p = 1. This is known as Leave one out cross validation. This method is generally preferred over the previous one because it does not suffer from the intensive computation, as number of possible combinations is equal to number of data points in original sample or n.

Cross Validation is a very useful technique for assessing the effectiveness of your model, particularly in cases where you need to mitigate overfitting. It is also of use in determining the hyper parameters of your model, in the sense that which parameters will result in lowest test error. This is all the basic you need to get started with cross validation. You can get started with all kinds of validation techniques using Scikit-Learn, that gets you up and running with just a few lines of code in python.

**5.4 TESTING TECHNIQUES**

Testing is a process, which reveals errors in the program.  It is the major quality measure employed during software development. During testing, the program is executed with a set of conditions known as test cases and the output is evaluated to determine whether the program is performing as expected. In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

**5.4.1 Unit Testing:**

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

**EACH MODULE CAN BE TESTED USING THE FOLLOWING TWO STRATEGIES:**

**5.4.2 Black Box Testing:**

In this strategy some test cases are generated as input conditions that fully     execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structure or external database access
* Performance errors
* Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

**5.4.3 White Box Testing:**

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

* Guarantee that all independent paths have been executed.
* Execute all logical decisions on their true and false sides.
* Execute all loops at their boundaries and within their operational
* Execute internal data structures to ensure their validity.

**5.4.4 Integration Testing:**

Integration testing ensures that software and subsystems work together as a whole.  It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

**5.4.5 System Testing:**

Involves in-house testing of the entire system before delivery to the user.  Its aim is to satisfy the user the system meets all requirements of the client's specifications.

**5.4.6 Acceptance Testing:**

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

**5.4.7 Validation Testing:**

          The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled.  In case of erroneous input corresponding error messages are displayed.

**5.4.8 Compile Testing:**

It was a good idea to do our stress testing early on, because it gave us time to fix some of the unexpected deadlocks and stability problems that only occurred when components were exposed to very high transaction volumes.

**5.4.9 Execution Test:**

This program was successfully loaded and executed. Because of good programming there was no execution error.

**5.4.10 Output Test:**

The successful output screens are placed in the output screens section.  
Testing is the process of finding differences between the expected behavior specified by system models and the observed behavior of the system.

**5.5 Code**

# Flask Packages

from flask import Flask,render\_template,request,url\_for

from flask\_bootstrap import Bootstrap

from flask\_uploads import UploadSet,configure\_uploads,IMAGES,DATA,ALL

from flask\_sqlalchemy import SQLAlchemy

from werkzeug.utils import secure\_filename

import os

import datetime

import time

# EDA Packages

import pandas as pd

import numpy as np

# ML Packages

from sklearn import model\_selection

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis

from sklearn.naive\_bayes import GaussianNB

from sklearn.svm import SVC

from sklearn.neural\_network import MLPClassifier

from sklearn.neural\_network import MLPRegressor

from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import RandomForestRegressor

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import GradientBoostingRegressor

from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()

# ML Packages For Vectorization of Text For Feature Extraction

from sklearn.feature\_extraction.text import CountVectorizer

from sklearn.feature\_extraction.text import TfidfVectorizer

import warnings

warnings.simplefilter('ignore')

app = Flask(\_\_name\_\_)

Bootstrap(app)

@app.route('/')

def index():

return render\_template('index.html')

# Route for our Processing and Details Page

@app.route('/dataupload',methods=['GET','POST'])

def dataupload():

if request.method == 'POST' and 'csv\_data' in request.files:

model\_type = request.form['model\_type']

file = request.files['csv\_data']

filename = secure\_filename(file.filename)

# os.path.join is used so that paths work in every operating system

# file.save(os.path.join("wherever","you","want",filename))

file.save(os.path.join('static/uploadsDB',filename))

fullfile = os.path.join('static/uploadsDB',filename)

# For Time

date = str(datetime.datetime.fromtimestamp(time.time()).strftime("%Y-%m-%d %H:%M:%S"))

# EDA function

df = pd.read\_csv(os.path.join('static/uploadsDB',filename))

df\_size = df.size

df\_shape = df.shape

df\_columns = list(df.columns)

df.fillna(-999, inplace=True)

df=df.apply(le.fit\_transform)

df\_targetname = df[df.columns[-1]].name

df\_featurenames = df\_columns[0:-1] # select all columns till last column

df\_Xfeatures = df.iloc[:,0:-1]

df\_Ylabels = df[df.columns[-1]] # Select the last column as target

# same as above df\_Ylabels = df.iloc[:,-1]

# Model Building

X = df\_Xfeatures

Y = df\_Ylabels

seed = 7

models = []

# prepare models

if model\_type=='classification':

models.append(('LogisticRegression', LogisticRegression()))

models.append(('LinearDiscriminantAnalysis', LinearDiscriminantAnalysis()))

models.append(('KNeighborsClassifier', KNeighborsClassifier()))

models.append(('DecisionTreeClassifier', DecisionTreeClassifier()))

models.append(('GaussianNB', GaussianNB()))

models.append(('MLPClassifier', MLPClassifier()))

models.append(('RandomForestClassifier', RandomForestClassifier()))

elif model\_type=='regression':

models.append(('LogisticRegression', LogisticRegression()))

#models.append(('GradientBoostingRegressor', GradientBoostingRegressor()))

models.append(('DecisionTreeRegressor', DecisionTreeRegressor()))

#models.append(('MLPRegressor', MLPRegressor()))

#models.append(('RandomForestRegressor', RandomForestRegressor()))

# evaluate each model in turn

results = []

names = []

allmodels = []

#prediction=[]

scoring = 'accuracy'

for name, model in models:

kfold = model\_selection.KFold(n\_splits=10, random\_state=seed)

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

results.append(cv\_results)

names.append(name)

msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())

allmodels.append(msg)

# Making predictions for the test data

model.fit(X, Y)

#pred=model.predict\_proba(X)[:,1]

model\_results = results

model\_names = names

#prediction.append(pred)

return render\_template('details.html',filename=filename,date=date,

df\_size=df\_size,

df\_shape=df\_shape,

df\_columns =df\_columns,

df\_targetname =df\_targetname,

model\_results = allmodels,

model\_names = names,

fullfile = fullfile,

dfplot = df,

models=models

#prediction=prediction

)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

Detail.html

{%extends "bootstrap/base.html" %}

{% block content %}

<!-- Main Content Is Here -->

<!-- Main jumbotron for a primary marketing message or call to action -->

<div class="jumbotron">

<div class="container">

<h1 class="display-3">Simplified MLApp</h1>

<p>Simplify Your Machine Learning and Data Exploration(Semi-Automated ML)</p>

<p><a href="{{ url\_for('login')}}" type="button" class="btn btn-danger" > Refresh</a></p>

</div>

</div>

<div class="container">

<!-- Details Page -->

<div class="row">

<div class="col-md-5" style="background-color:#3f76ef">

<h2>Details of Dataset</h2>

<div class="alert alert-info" role="alert">

<p><span style="color:red">Filename</span>: {{ filename }}</p>

</div>

<div class="alert alert-danger" role="alert">

<p><span style="color:red">Time Stamp</span>: {{ date }} </p>

</div>

<h2>Exploratory Data Analysis</h2>

<div class="alert alert-info" role="alert"><p><span style="color:red">Size</span>: {{ df\_size }}</p></div>

<div class="alert alert-danger" role="alert"><p><span style="color:red">Shape</span>: <br/>

(Row,Columns)<br/>

{{ df\_shape }}</p></div>

<div class="alert alert-info" role="alert">

<p><span style="color:red">Columns</span>:

{% for i in df\_columns %}

<li>{{ i}}</li>

{% endfor %}

</p>

</div>

<div class="alert alert-info" role="alert">

<p><span style="color:red">Data Types</span>:

{% for j in df\_dtypes %}

<li>{{ j}}</li>

{% endfor %}

</p>

</div>

<div class="alert alert-danger" role="alert">

<p><span style="color:red">Target</span>: {{ df\_targetname }}</p>

</div>

<div class="alert alert-danger" role="alert">

<p><span style="color:red">Number of Unique Values</span>: {{ df\_nunique }}</p>

</div>

<div class="alert alert-danger" role="alert">

<p><span style="color:red">Number of Missing Values</span>: {{ df\_missing }}</p>

</div>

<h2>Model Score</h2>

{% for mlaccuracy in model\_results %}

<div class="list-group">

<a href="#" class="list-group-item">

<span style="color:red">ML Algorithm | Mean Accuracy | Standard Deviation</span> </a>

<a href="#" class="list-group-item">{{ mlaccuracy }}</a>

</div>

{% endfor %}

{% for para in models %}

<div class="list-group">

<a href="#" class="list-group-item">

<span style="color:red">ML Algorithm | Parameters</span> </a>

<a href="#" class="list-group-item">{{ model\_names }}</a>

<a href="#" class="list-group-item">{{ para }}</a>

</div>

{% endfor %}

<div class="list-group">

<a href="#" class="list-group-item">

<span style="color:red">Model and Accuracy Comparing</span> </a>

<a href="#" class="list-group-item">{{ res }}</a>

</div>

</div>

<!-- Viewing Dataset As a Table-->

<div class="col-md-6">

<h2>Details of Dataset</h2>

<p>{{ fullfile }} </p>

<button onclick="displayDataset()" class="btn btn-primary">View Dataset</button>

<br/>

<br/>

<div id="myDIV">

{{ dfplot.to\_html(classes="table table-striped table-hover",na\_rep="-",index=False) | safe}}

</div>

<h2>Description of Dataset</h2>

<div class="table table-striped table-hover" >

{{ df\_describe.to\_html(classes="table striped",na\_rep="-",index=True) | safe}}

</div>

</div>

</div>

</div>

{% endblock %}

<!-- Main External 3-Party JS Is Here -->

{% block scripts %}

{{ super() }}

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>

<!-- Function for Displaying Table -->

<script>

function displayDataset()

{

var x = document.getElementById("myDIV");

if (x.style.display === "none") {

x.style.display = "block";

} else {

x.style.display = "none";

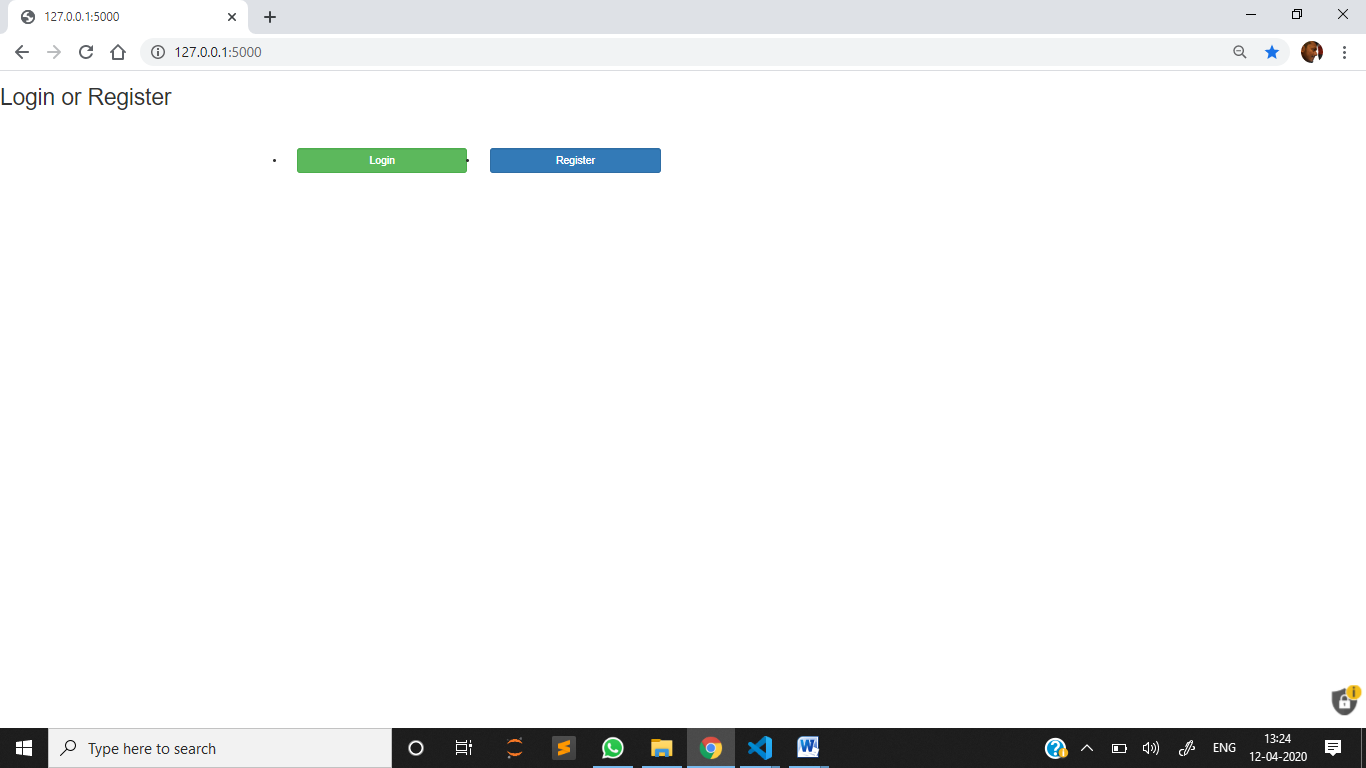
}

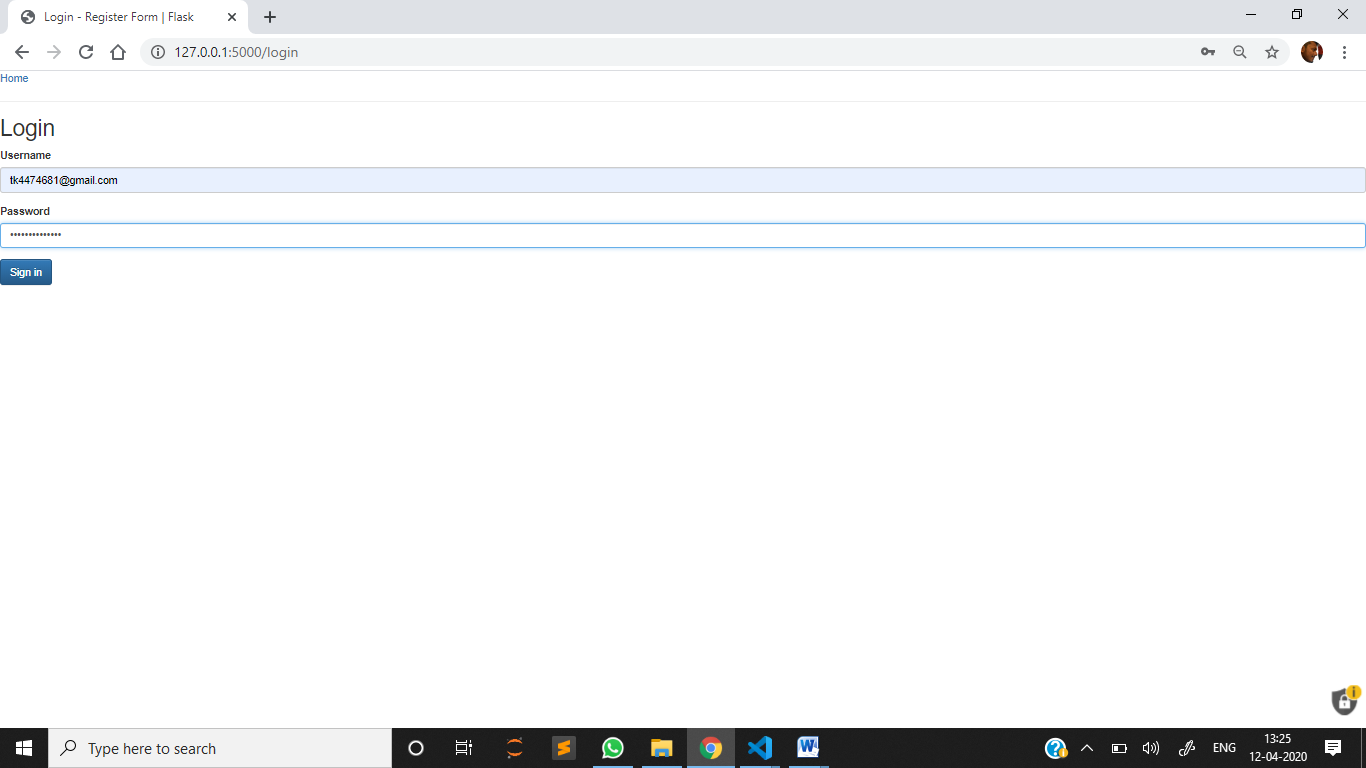
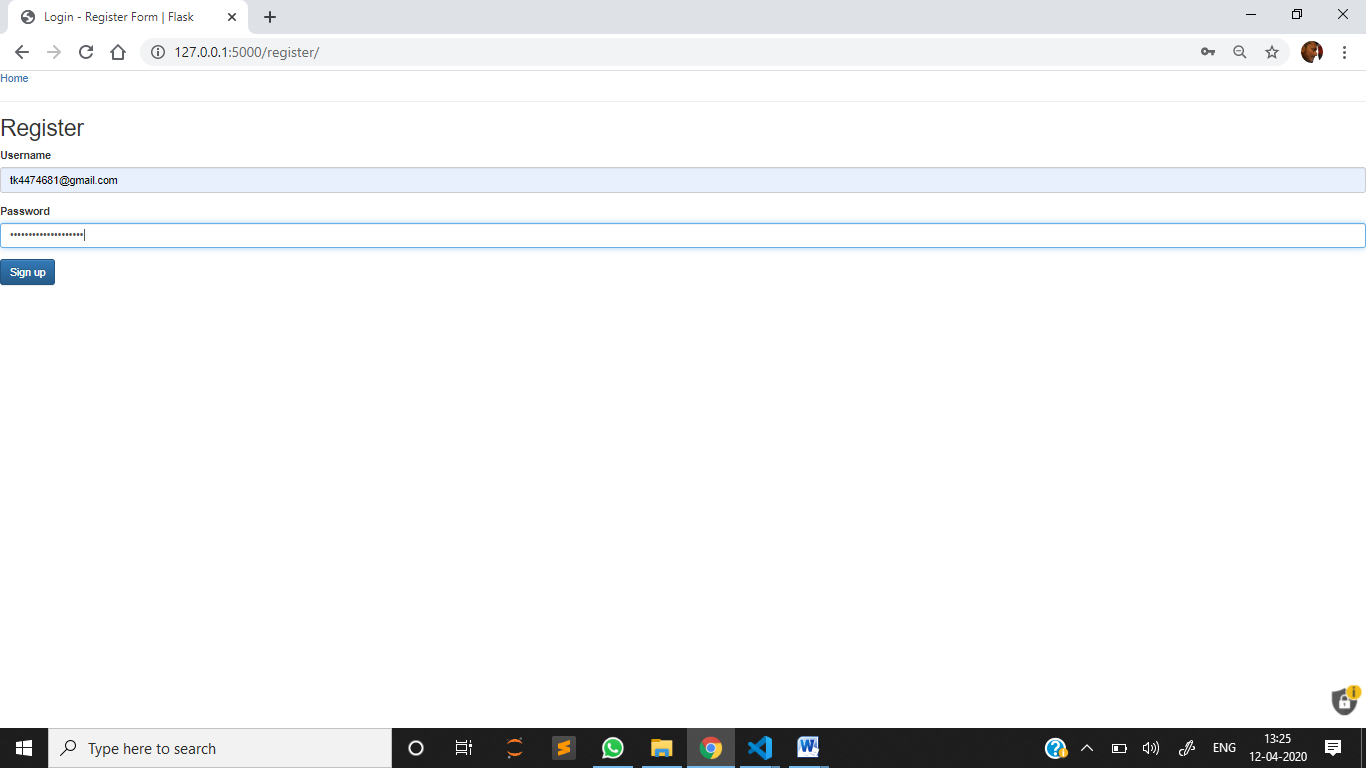
}

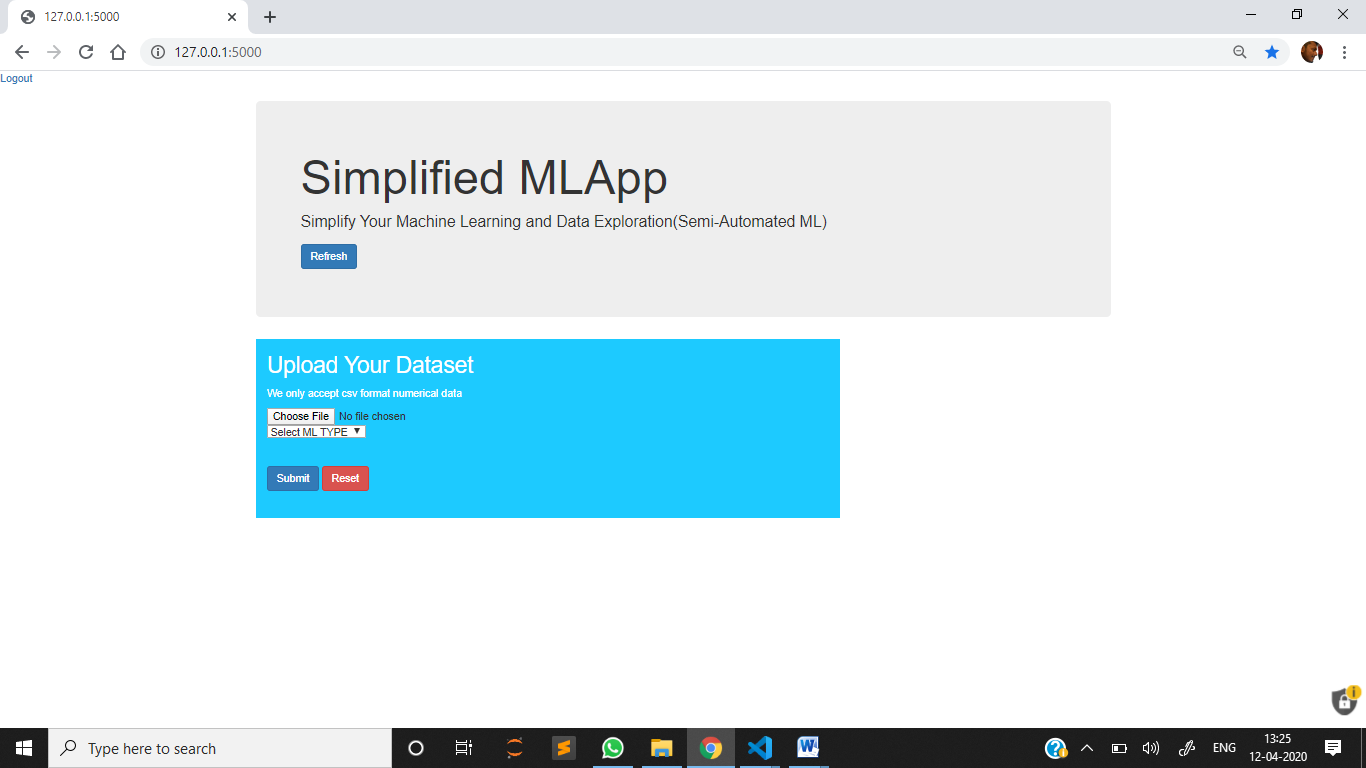
</script>

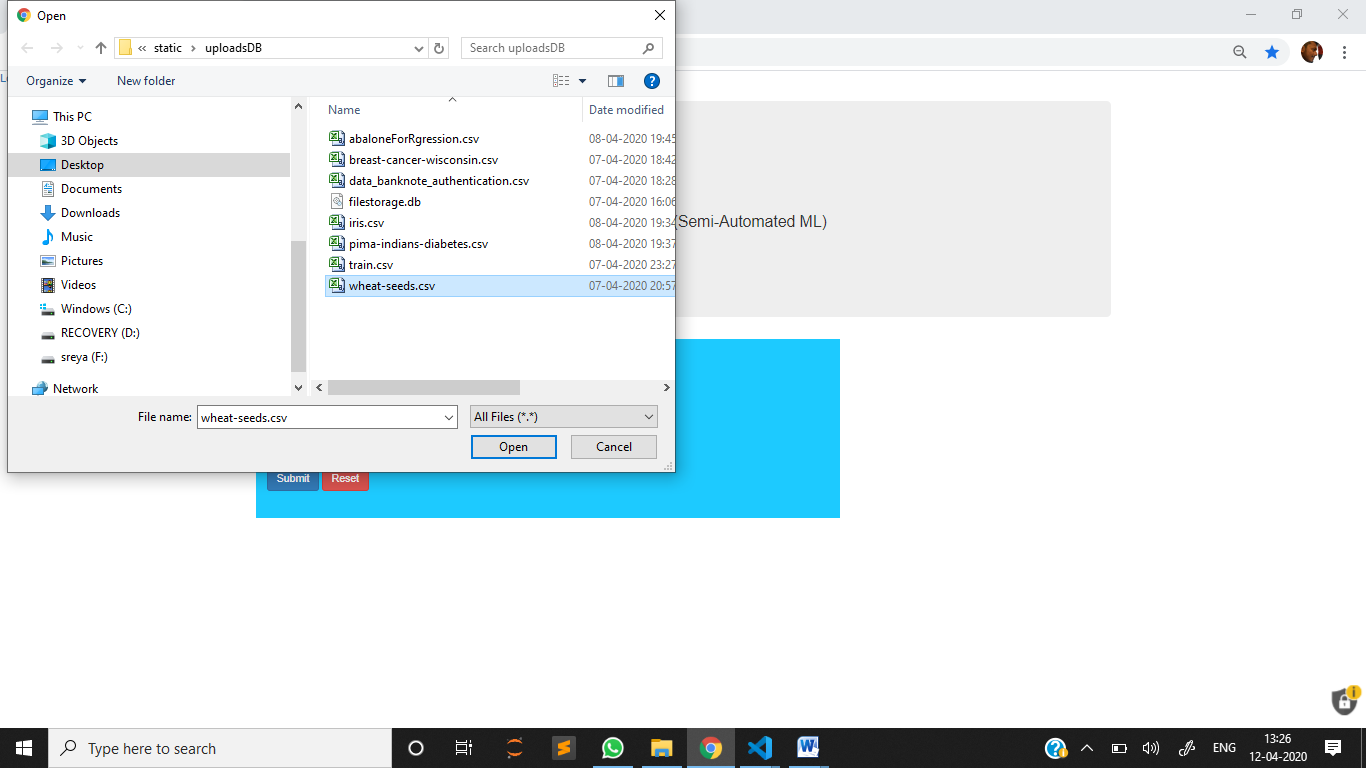
{% endblock%}

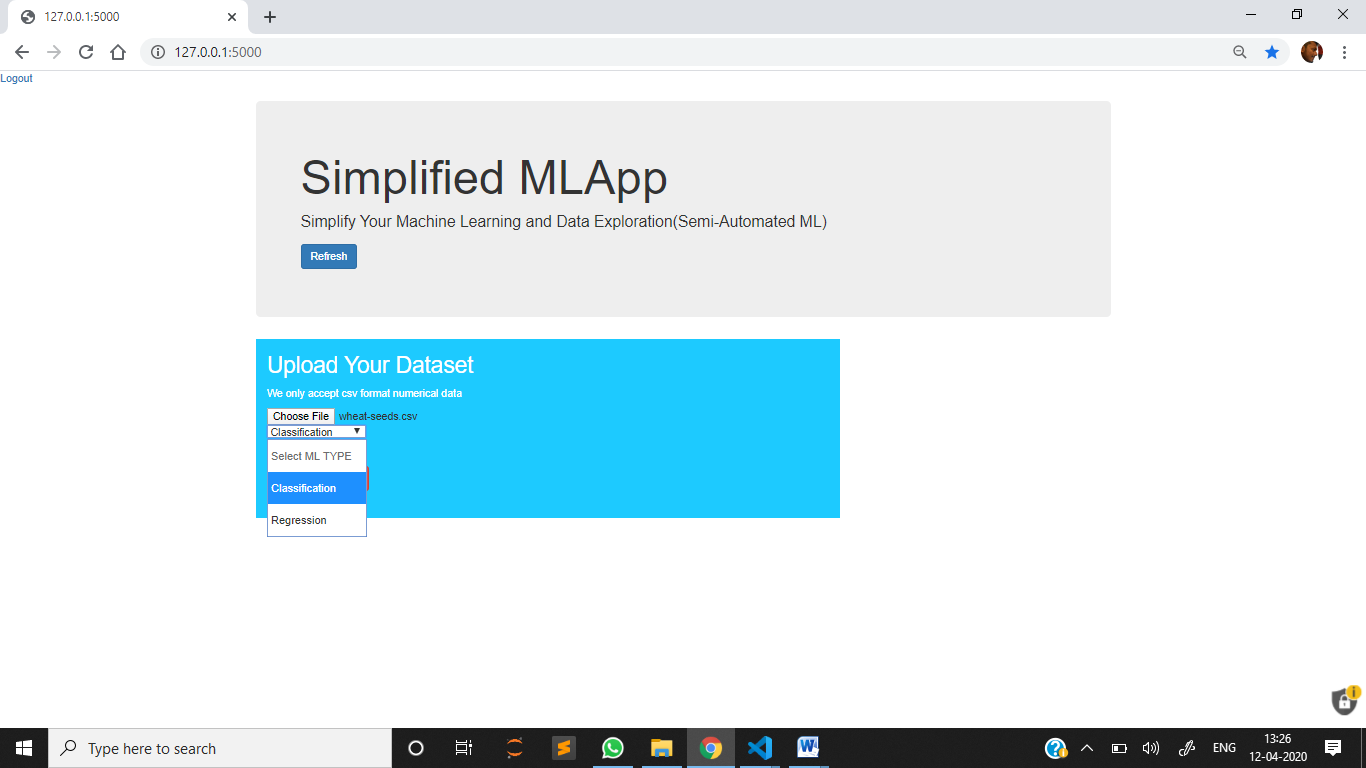
**5.5 OUTPUT SCREENSHOTS**

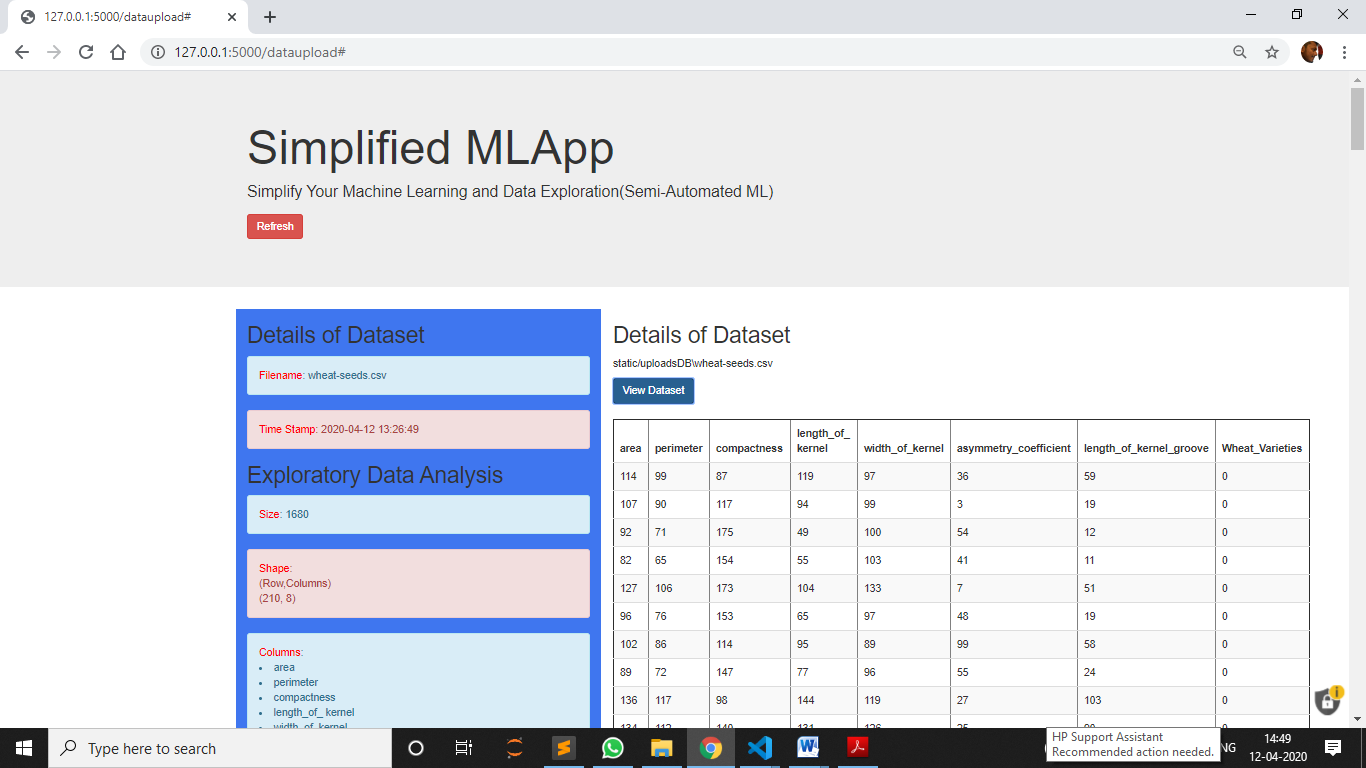


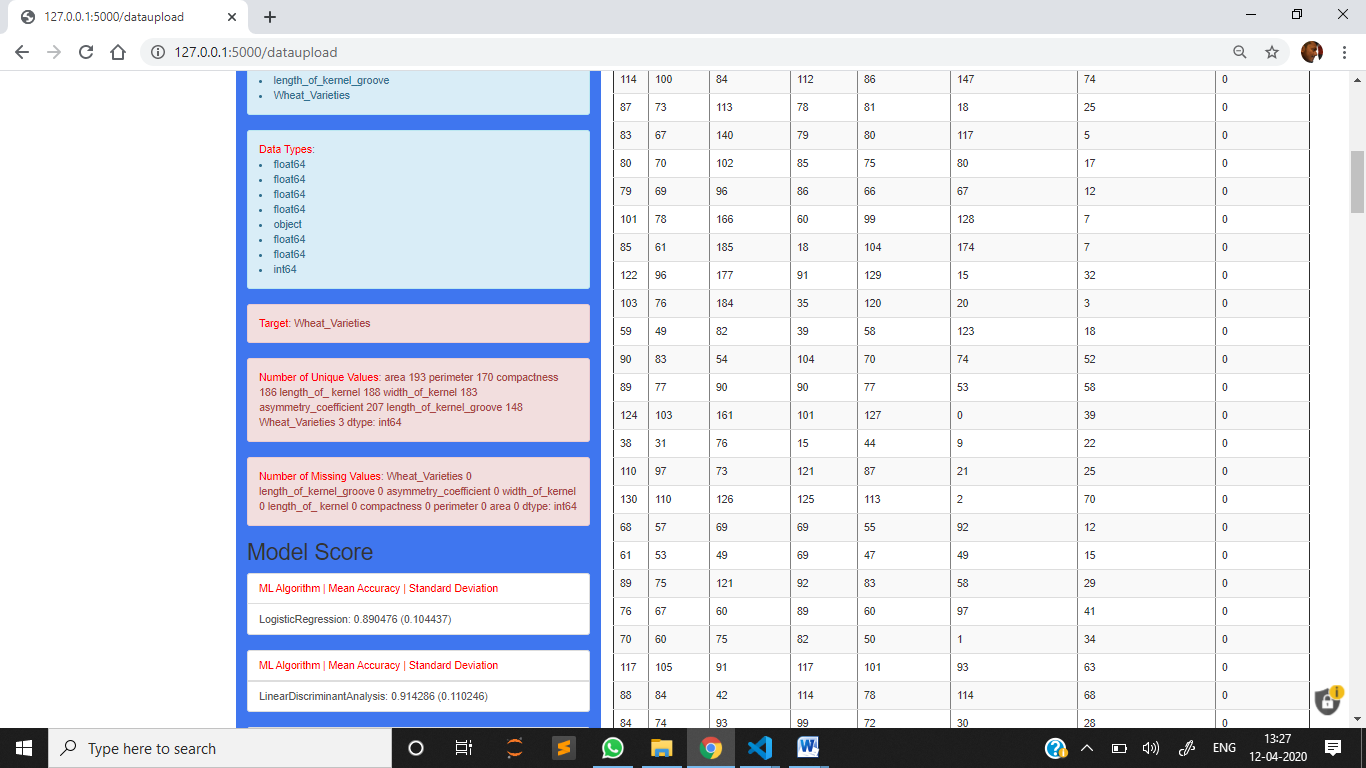
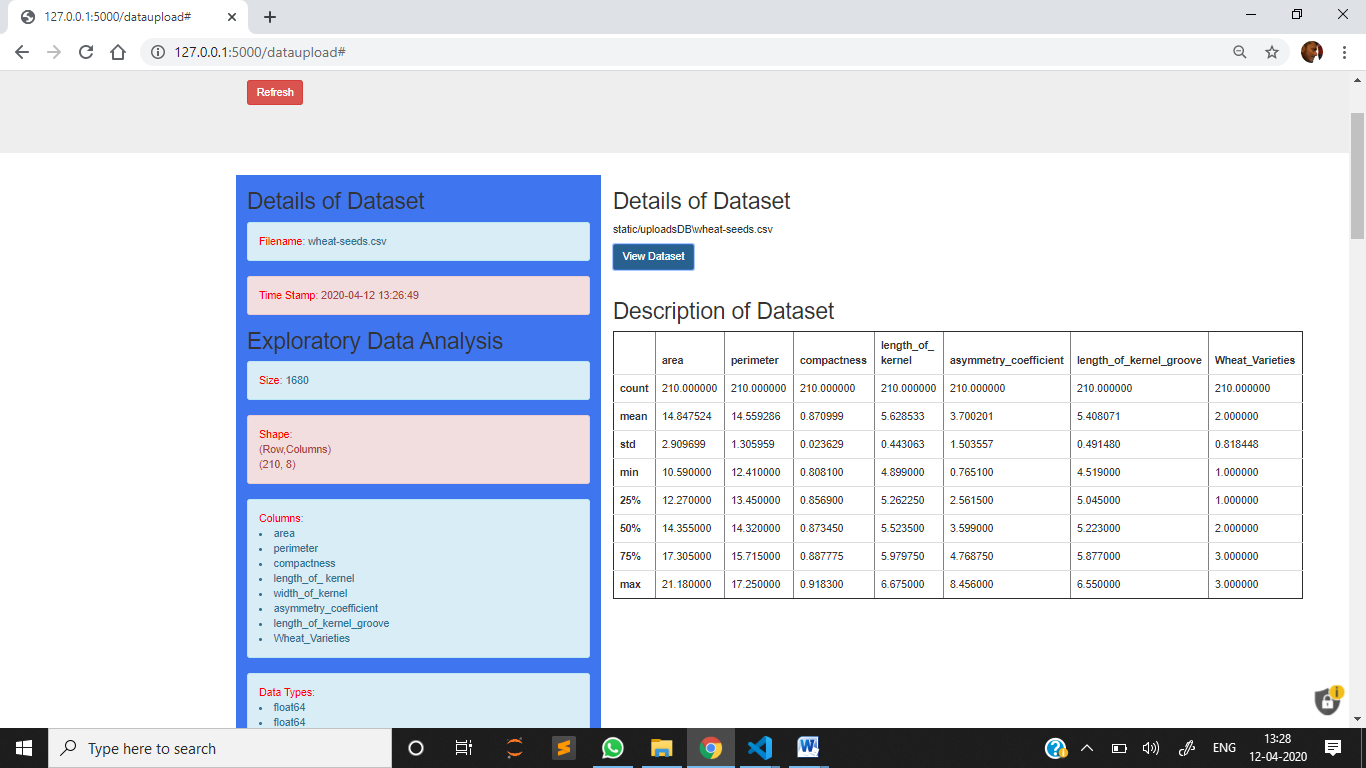
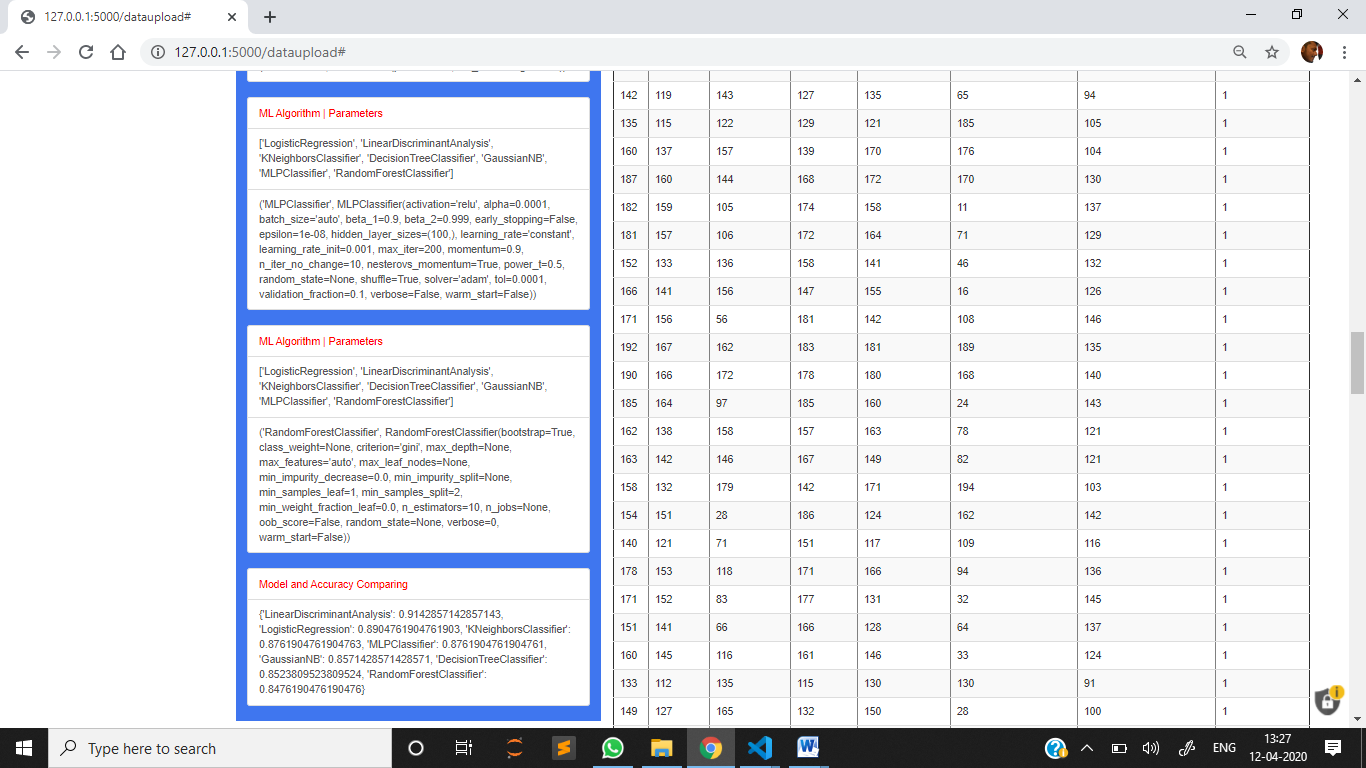
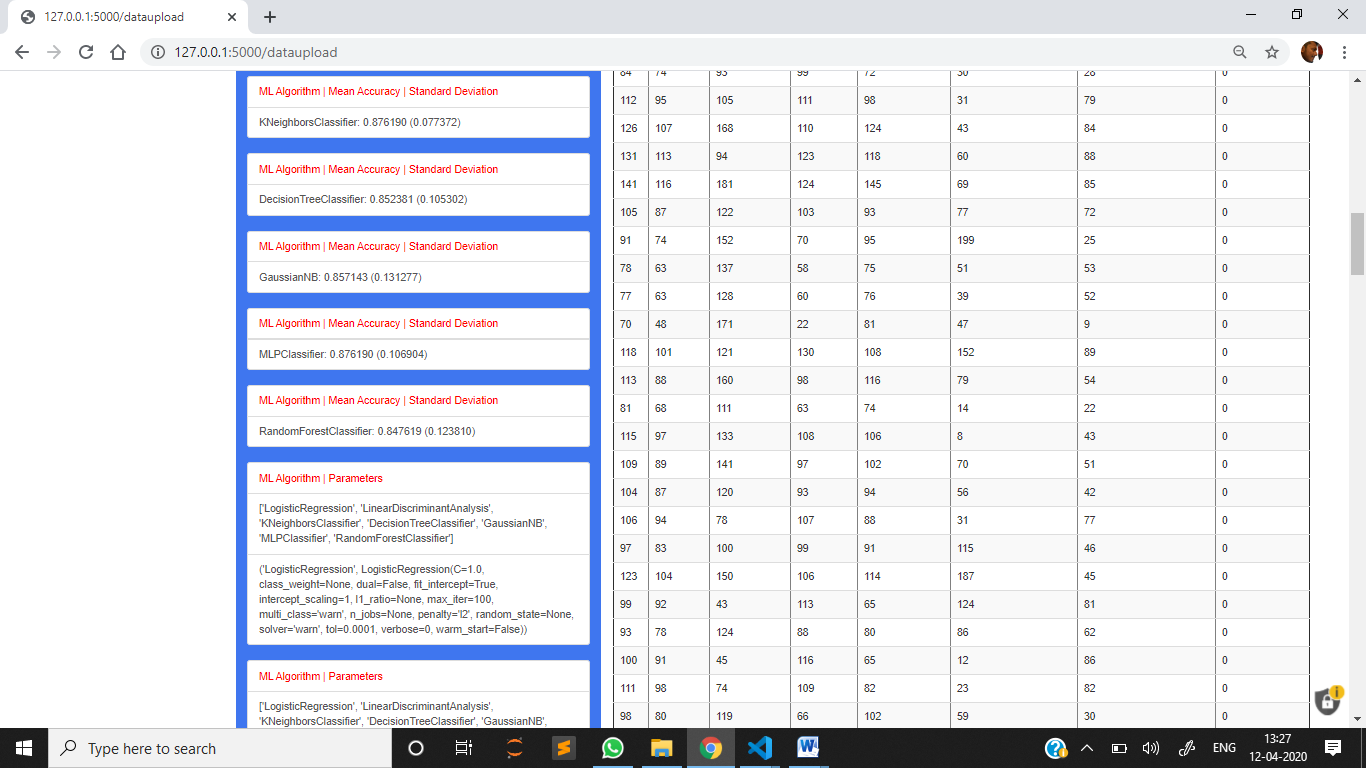










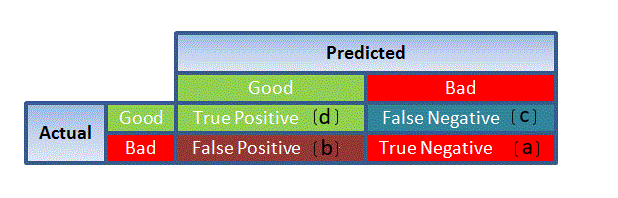
**SYSTEM TESTING**

**6. SYSTEM TESTING**

**6.1Model Evaluation**

The process of model building is not complete without evaluation of model’s performance. Suppose we have the predictions from the model, how can we decide whether the predictions are accurate? We can plot the results and compare them with the actual values, i.e. calculate the distance between the predictions and actual values. Lesser this distance more accurate will be the predictions. Since this is a classification problem, we can evaluate our models using any one of the following evaluation metrics:

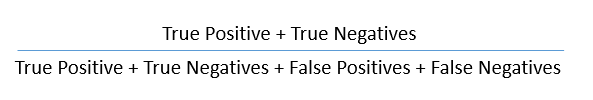
* **Accuracy**: Let us understand it using the confusion matrix which is a tabular representation of Actual vs Predicted values. This is how a confusion matrix looks like:



**Figure 6.1 : Accuracy**

* True Positive - Targets which are actually true(Y) and we have predicted them true(Y)
* True Negative - Targets which are actually false(N) and we have predicted them false(N)
* False Positive - Targets which are actually false(N) but we have predicted them true(T)
* False Negative - Targets which are actually true(T) but we have predicted them false(N)

Using these values, we can calculate the accuracy of the model. The accuracy is given by:



* **Precision**: It is a measure of correctness achieved in true prediction i.e. of observations labeled as true, how many are actually labeled true.

Precision = TP / (TP + FP)

* **Recall(Sensitivity)** - It is a measure of actual observations which are predicted correctly i.e. how many observations of true class are labeled correctly. It is also known as ‘Sensitivity’.

Recall = TP / (TP + FN)

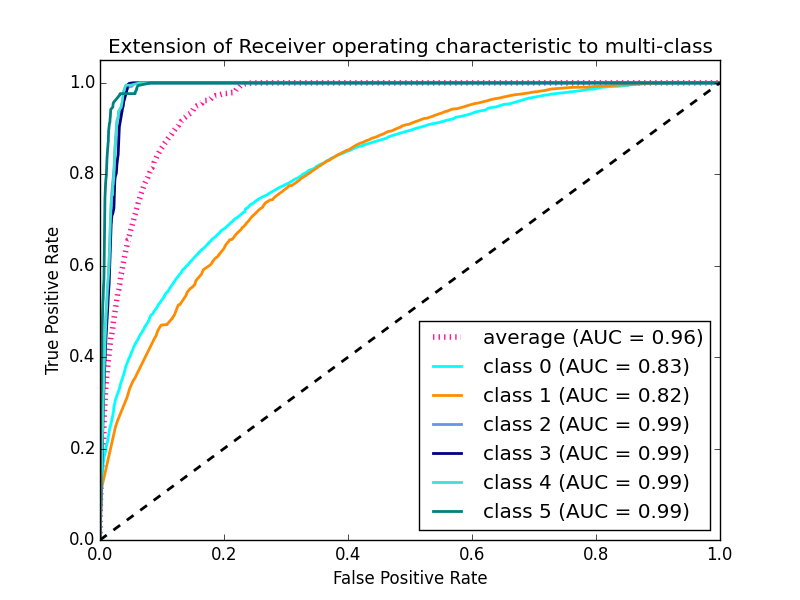
* **Specificity** - It is a measure of how many observations of false class are labeled correctly.

Specificity = TN / (TN + FP)

Specificity and Sensitivity plays a crucial role in deriving ROC curve.

* **ROC curve**
* Receiver Operating Characteristic(ROC) summarizes the model’s performance by evaluating the trade offs between true positive rate (sensitivity) and false positive rate(1- specificity).
* The area under curve (AUC), referred to as index of accuracy(A) or concordance index, is a perfect performance metric for ROC curve. Higher the area under curve, better the prediction power of the model.

This is how a ROC curve looks like:



**Figure 6.2 : ROC curve**

* The area of this curve measures the ability of the model to correctly classify true positives and true negatives. We want our model to predict the true classes as true and false classes as false.
* So it can be said that we want the true positive rate to be 1. But we are not concerned with the true positive rate only but the false positive rate too. For example in our problem, we are not only concerned about predicting the Y classes as Y but we also want N classes to be predicted as N.
* We want to increase the area of the curve which will be maximum for class 2,3,4 and 5 in the above example.
* For class 1 when the false positive rate is 0.2, the true positive rate is around 0.6. But for class 2 the true positive rate is 1 at the same false positive rate. So, the AUC for class 2 will be much more as compared to the AUC for class 1. So, the model for class 2 will be better.
* The class 2,3,4 and 5 model will predict more accurately as compared to the class 0 and 1 model as the AUC is more for those classes

**CONCLUSION**

**7. CONCLUSION**

This platform offers a one stop shop for applying multiple algorithms on a common dataset, with just a couple of mouse clicks the accuracies of some of the most commonly used algorithms pop on the screen, making it easy to compare of the end user.

**FUTURE ENHANCEMENTS**

**8. FUTURE ENHANCEMENTS**

We know there are a wide variety of algorithms available including deep learning modules, so these can be included as well for the future works. The feature of visualization is also a scope to be included in future

**BIBLIOGRAPHY**

**9. BIBLIOGRAPHY**

<http://people.scs.carleton.ca/~armyunis/projects/KAPI/porter.pdf>

An Introduction to Logistic Regression Analysis and Reporting

Article *in* The Journal of Educational Research · September 2002

<https://www.cs.princeton.edu/courses/archive/spr07/cos424/papers/mitchell-dectrees.pdf>

An Improved Random Forest Classifier for Text Categorization Baoxun Xu Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, China

<http://www.cs.cmu.edu/~knigam/papers/multinomial-aaaiws98.pdf>

Training Multilayer Perceptron Classifiers Based on a Modified Support Vector Method J. A. K. Suykens and J. Vandewalle

<https://scikit-learn.org/stable/supervised_learning.html#supervised-learning>