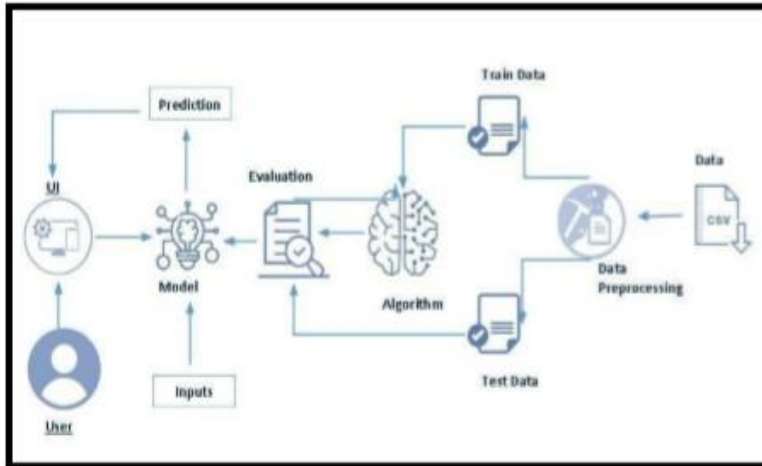


# **CHAPTER 1**

## **INTRODUCTION**

University admission is the process by which students are selected to attend a college or university. The process typically involves several steps, including submitting an application, taking entrance exams, and participating in interviews or other evaluations. Students are often worried about their chances of admission in University. the university admission process for students can be demanding, but by being well-informed, prepared, and organized, students can increase their chances of being admitted to the university of their choice. The aim of this project is to help students in short listing universities with their profiles. Machine learning algorithms are then used to train a model on this data, which can be used to predict the chances of future applicants being admitted. With this project, students can make more informed decisions about which universities to apply to, and universities can make more efficient use of their resources by focusing on the most promising applicants. The predicted output gives them a fair idea about their admission chances in a particular university. This analysis should also help students who are currently preparing or will be preparing to get a better idea.

## Technical Architecture:



### 1.1 Problem Understanding

The aim of this project is to help students in short listing universities with their profiles. Machine learning algorithms are then used to train a model on this data, which can be used to predict the chances of future applicants being admitted. With this project, students can make more informed decisions about which universities to apply to, and universities can make more efficient use of their resources by focusing on the most promising applicants. The predicted output gives them a fair idea about their admission chances in a particular university. This analysis should also help students who are currently preparing or will be preparing to get a better idea

### 1.2 Business Requirements

The business requirements for a machine learning model to predict chances of student admission in the university. A project aims to predict the chances of a student getting admitted to a particular university based on certain factors. The business value of this project is that it will help students make more informed decisions about which universities to apply to, and help university counselors to better advise students on the universities they are most likely to be admitted to the university.

### **1.3 Literature Survey**

The University Chances of Admission project is a well-researched topic in the field of education and machine learning. Many studies have been conducted to predict university admission using different machine learning techniques. One study by (Hsu and Chen, 2019) used decision tree, random forest, and logistic regression algorithms to predict the chance of university admission based on students' GPA, test scores, and personal information. The study found that the random forest algorithm performed the best with an accuracy of 85.5%. Another study by (Al-Shammari et al., 2018) used the k-nearest neighbor (KNN) algorithm to predict the chance of university admission based on students' GPA, test scores, and family income. The study found that the KNN algorithm performed well with an accuracy of 81.2%. A study by (Najafabadi et al., 2015) used a neural network to predict the chance of university admission based on students' GPA, test scores, and personal information. The study found that the neural network performed well with an accuracy of 94.3%. Overall, these studies suggest that various machine learning algorithms can be used to predict the chance of university admission with high accuracy.

### **1.4 Social or Business Impact**

#### **Social Impact:-**

The ability to accurately predict the chances of university admission can help students make more informed decisions about which universities to apply to, increasing their chances of being admitted and ultimately gaining access to higher education.

#### **Business Model/Impact:-**

1. Using machine learning models to predict university admission, the service can help universities more efficiently process and evaluate applications, potentially increasing the number of successful admissions.
2. An increase in the number of successful admissions can lead to an increase in revenue for universities, as well as for the company providing the prediction service

## CHAPTER 2

### SURVEY OF TECHNOLOGIES

#### 2.1 Introduction to Data Science

##### Data Science

Data Science is about data gathering, analysis and decision-making.

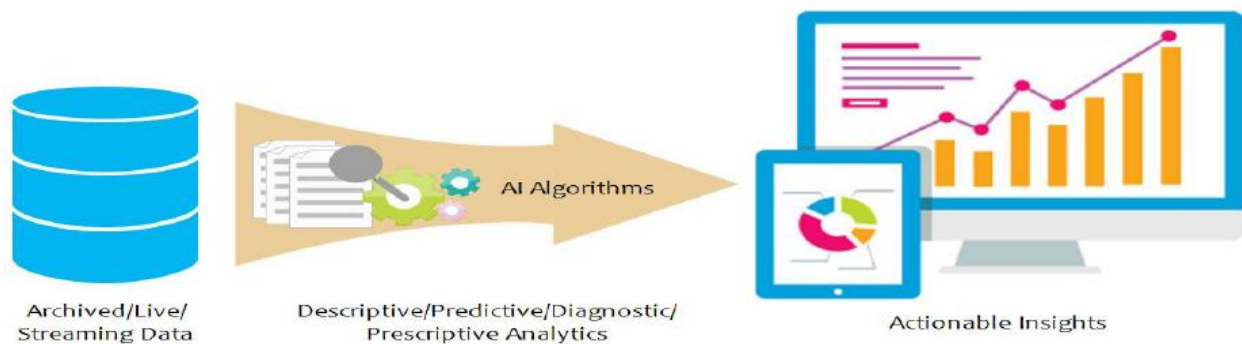
Data Science is about finding patterns in data, through analysis, and make future predictions.

By using Data Science, companies are able to make

Better decisions (should we choose A or B)

Predictive analysis (what will happen next?)

Pattern discoveries (find pattern, or maybe hidden information in the data)



#### Applications of Data science in Real-world:



## Components of Data Science:



## Tools For Data Science:



### 2.2 Google Colab

Colaboratory ("Colab" for short) is a data analysis and machine learning tool that allows you to combine executable Python code and rich text along with charts, images, HTML, LaTeX and more into a single document stored in Google Drive. It connects to powerful Google Cloud Platform runtimes and enables you to easily share your work and collaborate with others.

document stored in Google Drive.

Welcome To Colaboratory

File Edit View Insert Runtime Tools Help

+ Code + Text Copy to Drive

Connect Editing

## What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

### Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day

86400
```

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

## Use of Google Colab

The Basics. Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education

## 2.3 Kaggle

A subsidiary of Google, it is an online community of data scientists and machine learning engineers. Kaggle allows users to find datasets they want to use in building AI models, publish datasets, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.



### **Kaggle and how do you use it:**

Kaggle allows users to find datasets they want to use in building AI models, publish datasets, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

### **2.4 Python Basics**

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc)
  - Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

### **Python Operators:**

- Arithmetic operators
- Comparison operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

## Functions:

Functions are the most important aspect of an application. A function can be defined as the organized block of reusable code, which can be called whenever required.

There are mainly two types of functions.

### User-defined functions

The user-defined functions are those defined by the **user** to perform the specific task.

### Built-in functions

The built-in functions are those functions that are **pre-defined** in Python.

## Creating a Function:

Syntax:

```
def my_function(parameters):  
    function_block  
    return expression
```

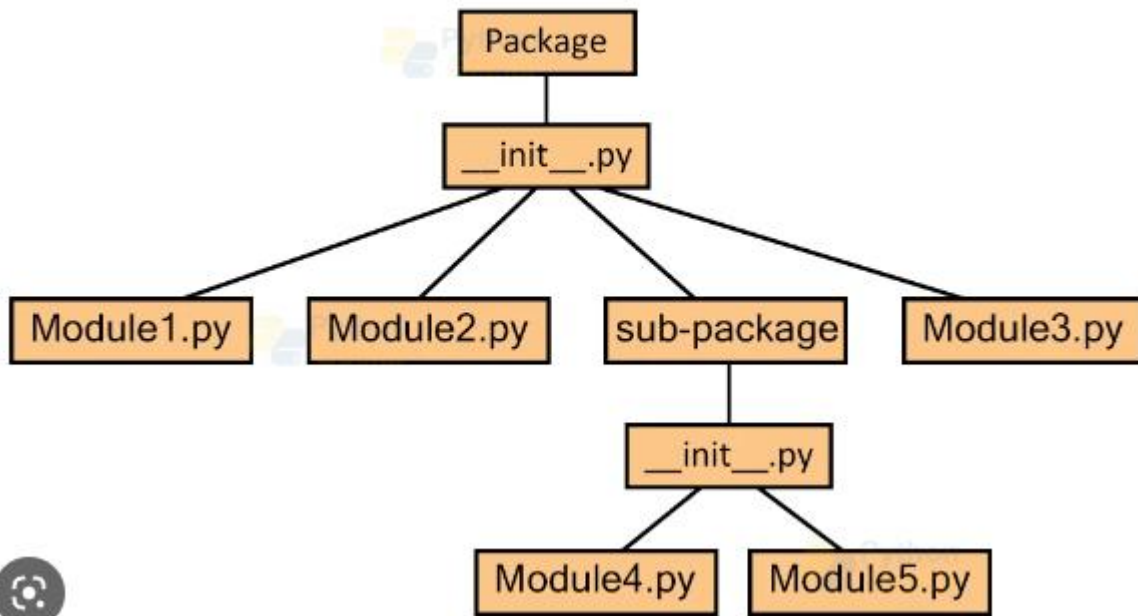
```
#function definition  
def hello_world():  
    print("hello world")  
# function calling  
hello_world()
```

## 2.5 Python Packages

A python package is a collection of modules. Modules that are related to each other are mainly put in the same package. When a module from an external package is required in a program, that package can be imported and its modules can be put to use.



# Structure of Packages



## 2.6 Numpy

NumPy (Numerical Python) is an open source Python library that's used in almost every field of science and engineering. It's the universal standard for working with numerical data in Python, and it's at the core of the scientific Python and PyData ecosystems.

### Why NumPy?

- In Python we have lists that serve the purpose of arrays, but they are slow to process.
- The NumPy library contains multidimensional array and matrix data. It provides **ndarray**, a homogeneous n-dimensional array object, with methods to efficiently operate on it.

## **Installing and Importing NumPy:**

```
py -m pip install numpy
```

## **Importing:**

To import numpy, run python3 on the terminal or any Notebook, and Type this  
**import numpy as np**

## **2.7 Pandas**

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.

## **Installing and Importing Pandas:**

```
py -m pip install adas
```

```
conda install pandas
```

Prefer pip  
pandas can be installed via pip from PyPI.

```
pip install pandas
```

## **Importing:**

To verify whether Pandas is installed correctly on your system, run python3 on the terminal or any Notebook, and Type this  
**import pandas as pd**

## **2.8 Data Visualization**

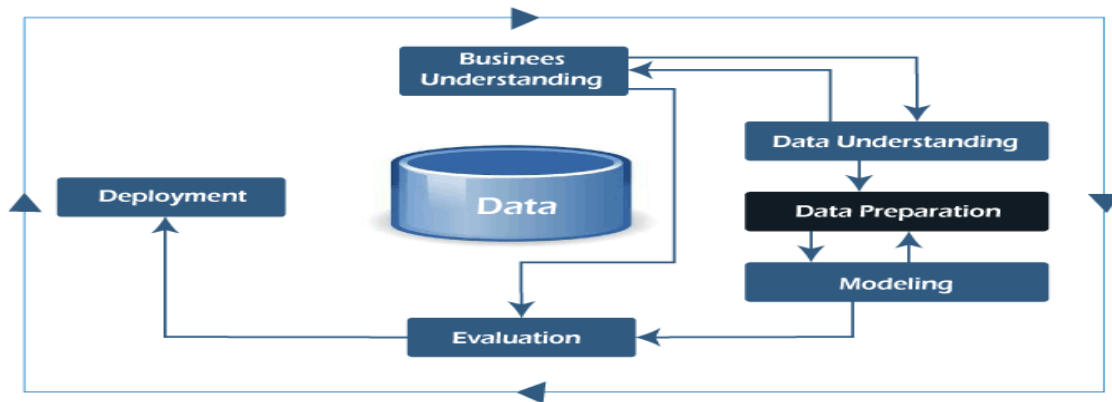
Data visualization is the representation of data through use of common graphics, such as charts, plots, infographics, and even animations. These visual displays of information communicate complex data relationships and data-driven insights in a way that is easy to understand.

Data visualization is a critical step in the data science process, helping teams and individuals convey data more effectively to colleagues and decision makers. Teams that manage reporting systems typically leverage defined template views to monitor performance. However, data visualization isn't limited to performance dashboards. For example, while text mining an analyst may use a word cloud to capture key concepts, trends, and hidden relationships within this unstructured data. Alternatively, they may utilize a graph structure to illustrate relationships between entities in a knowledge graph. There are a number of ways to represent different types of data, and it's important to remember that it is a skillset that should extend beyond your core analytics team.

## **2.9 Data Wrangling**

Data wrangling describes a series of processes designed to explore, transform, and validate raw datasets from their messy and complex forms into high-quality data. You can use your wrangled data to produce valuable insights and guide business decisions.

Data wrangling is the process of converting raw data into a usable form. It may also be called data munging or data remediation.



## 2.10 Supervised learning

Supervised learning, also known as supervised machine learning, is a subcategory of machine learning and artificial intelligence. It is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately. As input data is fed into the model, it adjusts its weights until the model has been fitted appropriately, which occurs as part of the cross validation process. Supervised learning helps organizations solve for a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox.

### 2.10.1 Regression

Regression is a statistical method used in finance, investing, and other disciplines that attempts to determine the strength and character of the relationship between one dependent variable (usually denoted by Y) and a series of other variables (known as independent variables).

Also called simple regression or ordinary least squares (OLS), linear regression is the most common form of this technique. Linear regression establishes the linear relationship between two variables based on a line of best fit. Linear regression is thus graphically depicted using a straight line with the slope defining how the change in one variable impacts a change in the other. The y-intercept of a linear regression relationship represents the value of one variable when the value of the other is zero. Non-linear regression models also exist, but are far more complex.

Regression analysis is a powerful tool for uncovering the associations between variables observed in data, but cannot easily indicate causation. It is used in several contexts in business, finance, and economics. For instance, it is used to help investment managers value assets and understand the relationships between factors such as commodity prices and the stocks of businesses dealing in those commodities.

Regression as a statistical technique should not be confused with the concept of regression to the mean (mean reversion)

### 2.10.2 Classification

The Classification algorithm is a Supervised Learning technique that is used to identify the category of new observations on the basis of training data. In Classification, a program learns from the given dataset or observations and then classifies new observation into a number of classes or groups. Such as, **Yes or No, 0 or 1, Spam or Not Spam, cat or dog**, etc. Classes can be called as targets/labels or categories.

Unlike regression, the output variable of Classification is a category, not a value, such as "Green or Blue", "fruit or animal", etc. Since the Classification algorithm is a Supervised learning technique, hence it takes labeled input data, which means it contains input with the corresponding output.

In classification algorithm, a discrete output function( $y$ ) is mapped to input variable( $x$ ).

$$y=f(x), \text{ where } y = \text{categorical output}$$

The best example of an ML classification algorithm is **Email Spam Detector**.

The main goal of the Classification algorithm is to identify the category of a given dataset, and these algorithms are mainly used to predict the output for the categorical data.

Classification algorithms can be better understood using the below diagram. In the below diagram, there are two classes, class A and Class B. These classes have features that are similar to each other and dissimilar to other classes.

### 2.10.3 Model Evaluation Metrics

Evaluation metric refers to a measure that we use to evaluate different models. Choosing an appropriate evaluation metric is a decision problem that requires a thorough understanding of the goal of a project and is a fundamental step before all modeling process that follows. So why is it so important, and how should we choose?

### 2.10.4 Hyper Parameter Optimization

Before I define hyperparameter optimization, you need to understand what a hyperparameter is.

In short, hyperparameters are different parameter values that are used to control the learning process and have a significant effect on the performance of machine learning models.

An example of hyperparameters in the Random Forest algorithm is the number of estimators (*n\_estimators*), maximum depth (*max\_depth*), and criterion. These parameters are **tunable** and can directly affect how well a model trains.

So then **hyperparameter optimization** is the process of finding the right combination of hyperparameter values to achieve maximum performance on the data in a reasonable amount of time.

This process plays a vital role in the prediction accuracy of a machine learning algorithm. Therefore Hyperparameter optimization is considered the **trickiest** part of building machine learning models.

Most of these machine learning algorithms come with the default values of their hyperparameters. But the default values do not always perform well on

different types of Machine Learning projects. This is why you need to optimize them in order to get the right combination that will give you the best performance.

## 2.11 Introduction to Unsupervised Learning

Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition.

Unsupervised learning models are utilized for three main tasks—clustering, association, and dimensionality reduction. Below we'll define each learning method and highlight common algorithms and approaches to conduct them effectively.

### Clustering

Clustering is a data mining technique which groups unlabeled data based on their similarities or differences. Clustering algorithms are used to process raw, unclassified data objects into groups represented by structures or patterns in the information. Clustering algorithms can be categorized into a few types, specifically exclusive, overlapping, hierarchical, and probabilistic

### Exclusive and Overlapping Clustering

Exclusive clustering is a form of grouping that stipulates a data point can exist only in one cluster. This can also be referred to as “hard” clustering. The K-means clustering algorithm is an example of exclusive clustering.

- **K-means clustering** is a common example of an exclusive clustering method where data points are assigned into K groups, where K represents the number of clusters based on the distance from each group's centroid. The data points closest to a given centroid will be clustered under the same category. A larger K value will be indicative of smaller groupings with more granularity whereas

a smaller K value will have larger groupings and less granularity. K-means clustering is commonly used in market segmentation, document clustering, image segmentation, and image compression. Overlapping clusters differs from exclusive clustering in that it allows data points to belong to multiple clusters with separate degrees of membership. “Soft” or fuzzy k-means clustering is an example of overlapping clustering.

## **Hierarchical clustering**

Hierarchical clustering, also known as hierarchical cluster analysis (HCA), is an unsupervised clustering algorithm that can be categorized in two ways; they can be agglomerative or divisive. Agglomerative clustering is considered a “bottoms-up approach.” Its data points are isolated as separate groupings initially, and then they are merged together iteratively on the basis of similarity until one cluster has been achieved. Four different methods are commonly used to measure similarity:

1. **Ward’s linkage:** This method states that the distance between two clusters is defined by the increase in the sum of squared after the clusters are merged.
2. **Average linkage:** This method is defined by the mean distance between two points in each cluster
3. **Complete (or maximum) linkage:** This method is defined by the maximum distance between two points in each cluster
4. **Single (or minimum) linkage:** This method is defined by the minimum distance between two points in each cluster

Euclidean distance is the most common metric used to calculate these distances; however, other metrics, such as Manhattan distance, are also cited in clustering literature.

Divisive clustering can be defined as the opposite of agglomerative clustering; instead it takes a “top-down” approach. In this case, a single data cluster is divided based on the differences between data points. Divisive clustering is not commonly used, but it is still worth noting in the context of hierarchical clustering. These



clustering processes are usually visualized using a dendrogram, a tree-like diagram that documents the merging or splitting of data points at each iteration.

## **Probabilistic clustering**

A probabilistic model is an unsupervised technique that helps us solve density estimation or “soft” clustering problems. In probabilistic clustering, data points are clustered based on the likelihood that they belong to a particular distribution. The Gaussian Mixture Model (GMM) is the one of the most commonly used probabilistic clustering methods.

- **Gaussian Mixture Models** are classified as mixture models, which means that they are made up of an unspecified number of probability distribution functions. GMMs are primarily leveraged to determine which Gaussian, or normal, probability distribution a given data point belongs to. If the mean or variance are known, then we can determine which distribution a given data point belongs to. However, in GMMs, these variables are not known, so we assume that a latent, or hidden, variable exists to cluster data points appropriately. While it is not required to use the Expectation-Maximization (EM) algorithm, it is a commonly used to estimate the assignment probabilities for a given data point to a particular data cluster.

An association rule is a rule-based method for finding relationships between variables in a given dataset. These methods are frequently used for market basket analysis, allowing companies to better understand relationships between different products. Understanding consumption habits of customers enables businesses to develop better cross-selling strategies and recommendation engines. Examples of this can be seen in Amazon’s “Customers Who Bought This Item Also Bought” or Spotify’s "Discover Weekly" playlist. While there are a few different algorithms used to generate association rules, such as Apriori, Eclat, and FP-Growth, the Apriori algorithm is most widely used.

## **Apriori algorithms**

Apriori algorithms have been popularized through market basket analyses, leading to different recommendation engines for music platforms and online retailers. They are used within transactional datasets to identify frequent itemsets, or collections of items, to identify the likelihood of consuming a product given the consumption of another product. For example, if I play Black Sabbath's radio on Spotify, starting with their song "Orchid", one of the other songs on this channel will likely be a Led Zeppelin song, such as "Over the Hills and Far Away." This is based on my prior listening habits as well as the ones of others. Apriori algorithms use a hash tree to count itemsets, navigating through the dataset in a breadth-first manner.

## **Dimensionality reduction**

While more data generally yields more accurate results, it can also impact the performance of machine learning algorithms (e.g. overfitting) and it can also make it difficult to visualize datasets. Dimensionality reduction is a technique used when the number of features, or dimensions, in a given dataset is too high. It reduces the number of data inputs to a manageable size while also preserving the integrity of the dataset as much as possible. It is commonly used in the preprocessing data stage, and there are a few different dimensionality reduction methods that can be used, such as:

### **Principal component analysis**

Principal component analysis (PCA) is a type of dimensionality reduction algorithm which is used to reduce redundancies and to compress datasets through feature extraction. This method uses a linear transformation to create a new data representation, yielding a set of "principal components." The first principal component is the direction which maximizes the variance of the dataset. While the second principal component also finds the maximum variance in the data, it is completely uncorrelated to the first principal component, yielding a direction that is perpendicular, or orthogonal, to the first component. This process repeats based on the number of dimensions, where a next principal component is the direction orthogonal to the prior components with the most variance.

## Singular value decomposition

Singular value decomposition (SVD) is another dimensionality reduction approach which factorizes a matrix,  $A$ , into three, low-rank matrices. SVD is denoted by the formula,  $A = USVT$ , where  $U$  and  $V$  are orthogonal matrices.  $S$  is a diagonal matrix, and  $S$  values are considered singular values of matrix  $A$ . Similar to PCA, it is commonly used to reduce noise and compress data, such as image files.

## Autoencoders

Autoencoders leverage neural networks to compress data and then recreate a new representation of the original data's input. Looking at the image below, you can see that the hidden layer specifically acts as a bottleneck to compress the input layer prior to reconstructing within the output layer. The stage from the input layer to the hidden layer is referred to as “encoding” while the stage from the hidden layer to the output layer is known as “decoding.”

Applications of unsupervised learning

Machine learning techniques have become a common method to improve a product user experience and to test systems for quality assurance. Unsupervised learning provides an exploratory path to view data, allowing businesses to identify patterns in large volumes of data more quickly when compared to manual observation. Some of the most common real-world applications of unsupervised learning are:

- **News Sections:** Google News uses unsupervised learning to categorize articles on the same story from various online news outlets. For example, the results of a presidential election could be categorized under their label for “US” news.
- **Computer vision:** Unsupervised learning algorithms are used for visual perception tasks, such as object recognition.
- **Medical imaging:** Unsupervised machine learning provides essential features to medical imaging devices, such as image detection, classification and segmentation, used in radiology and pathology to diagnose patients quickly and accurately.

- **Anomaly detection:** Unsupervised learning models can comb through large amounts of data and discover atypical data points within a dataset. These anomalies can raise awareness around faulty equipment, human error, or breaches in security.
- **Customer personas:** Defining customer personas makes it easier to understand common traits and business clients' purchasing habits. Unsupervised learning allows businesses to build better buyer persona profiles, enabling organizations to align their product messaging more appropriately.
- **Recommendation Engines:** Using past purchase behavior data, unsupervised learning can help to discover data trends that can be used to develop more effective cross-selling strategies. This is used to make relevant add-on recommendations to customers during the checkout process for online retailers.

## 2.12 System Requirements

Machine learning is a subset of artificial intelligence function that provides the system with the ability to learn from data without being programmed explicitly.

Machine learning is basically a mathematical and probabilistic model which requires tons of computations. It is very trivial for humans to do those tasks, but computational machines can perform similar tasks very easily.

Consumer hardware may not be able to do extensive computations very quickly as a model may require to calculate and update millions of parameters in run-time for a single iterative model like deep neural networks.

Thus, there is a scope for the hardware which works well with extensive calculation. But before we dive deep into hardware for ML, let's understand machine learning flow.

There are four steps for preparing a machine learning model:

1. Preprocessing input data
2. Training the deep learning model
3. Storing the trained deep learning model
4. Deployment of the model

Among all these, training the machine learning model is the most computationally intensive task.

Now if we talk about training the model, which generally requires a lot of computational power, the process could be frustrating if done without the right hardware. This intensive part of the neural network is made up of various matrix multiplications.

## CHAPTER 3

### DATA COLLECTION & PREPRATION

#### 3.1 Collect the dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com.

Please refer to the link given below to download the dataset. Link :

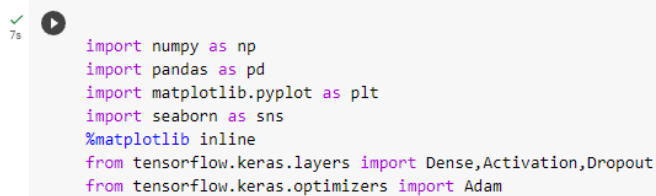
<https://www.kaggle.com/rishal005/admission-predict>

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

#### Importing the libraries

Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight.



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import Adam
```

#### Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read\_csv() to read the dataset. As a parameter we have to give the directory of the csv file

```
✓ [2] data = pd.read_csv('/content/Admission_Predict (1).csv')
```

## 3.2 Data Preparation

As we have understood how the data is, let's pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data
- Handling Imbalance Data

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

### Handling missing values

- Let's find the shape of our dataset first. To find the shape of our data, the `df.shape()` method is used. To find the data type, `df.info()` function is used.

```
✓ [3] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Serial No.            400 non-null   int64
1   GRE Score              400 non-null   int64
2   TOEFL Score            400 non-null   int64
3   University Rating      400 non-null   int64
4   SOP                    400 non-null   float64
5   LOR                    400 non-null   float64
6   CGPA                   400 non-null   float64
7   Research               400 non-null   int64
8   Chance of Admit        400 non-null   float64
dtypes: float64(4), int64(5)
memory usage: 28.2 KB
```

- For checking the null values, `df.isnull()` function is used. To sum those null values we use `.sum()` function. From the below image we found that there are no null values present in our dataset.

```
✓ [5] data.isnull().any()
0s
Serial No.      False
GRE Score       False
TOEFL Score     False
University Rating False
SOP             False
LOR             False
CGPA           False
Research        False
Chance of Admit False
dtype: bool
```

- Let us rename the column, in python have a inbuilt function rename( ). We can easily rename the column names.

```
✓ [6] data=data.rename(columns = {'Chance of Admit ':'Chance of Admit'})
0s
```



## CHAPTER 4

### EXPLORATORY DATA ANALYSIS

#### 4.1 Descriptive statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features

```
data.describe()
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000	400.000000
mean	200.500000	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	115.614301	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362	0.142609
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	100.750000	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	0.640000
50%	200.500000	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.730000
75%	300.250000	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000	0.830000
max	400.000000	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

#### 4.2 Visual Analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

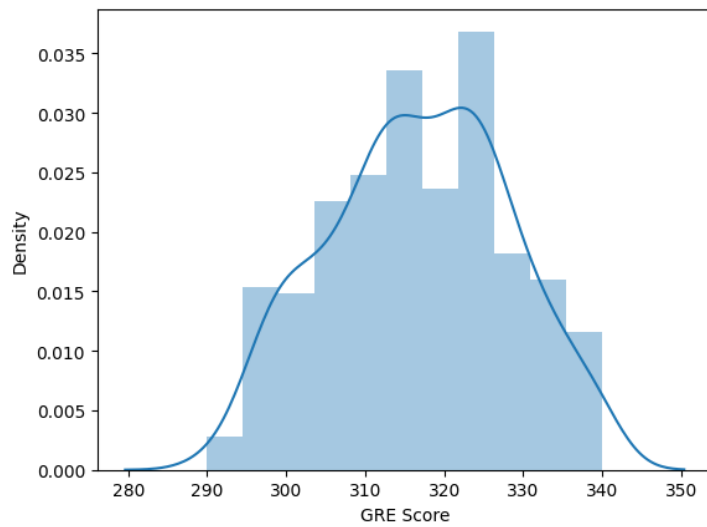
##### Univariate analysis

In simple words, univariate analysis is understanding the data with a single feature. Here we have displayed two different graphs such as distplot and countplot.

- The Seaborn package provides a wonderful function distplot. With the help of distplot, we can find the distribution of the feature. To make multiple graphs in a single plot, we use subplot.

```
sns.distplot(data['GRE Score'])
```

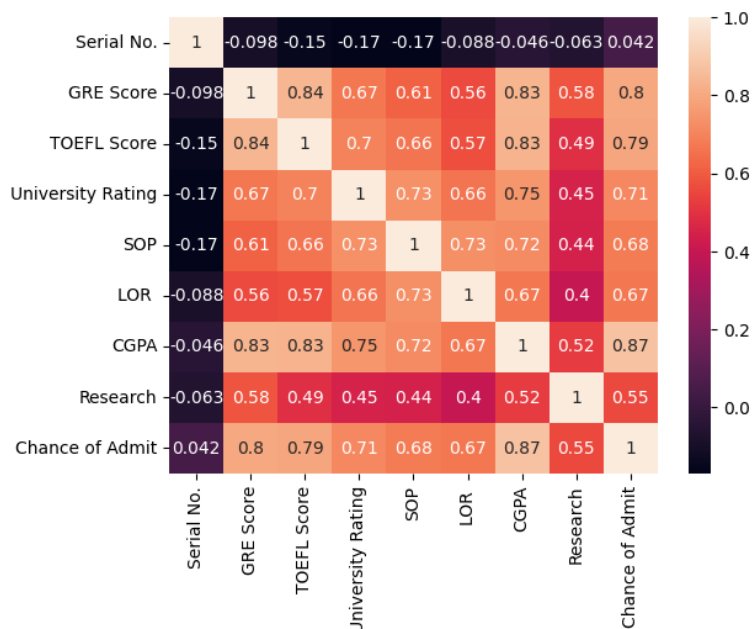
```
sns.distplot(data['GRE Score'])
<Axes: xlabel='GRE Score', ylabel='Density'>
```



## Bivariate analysis

```
15 ✓ sns.heatmap(data.corr(), annot=True)
```

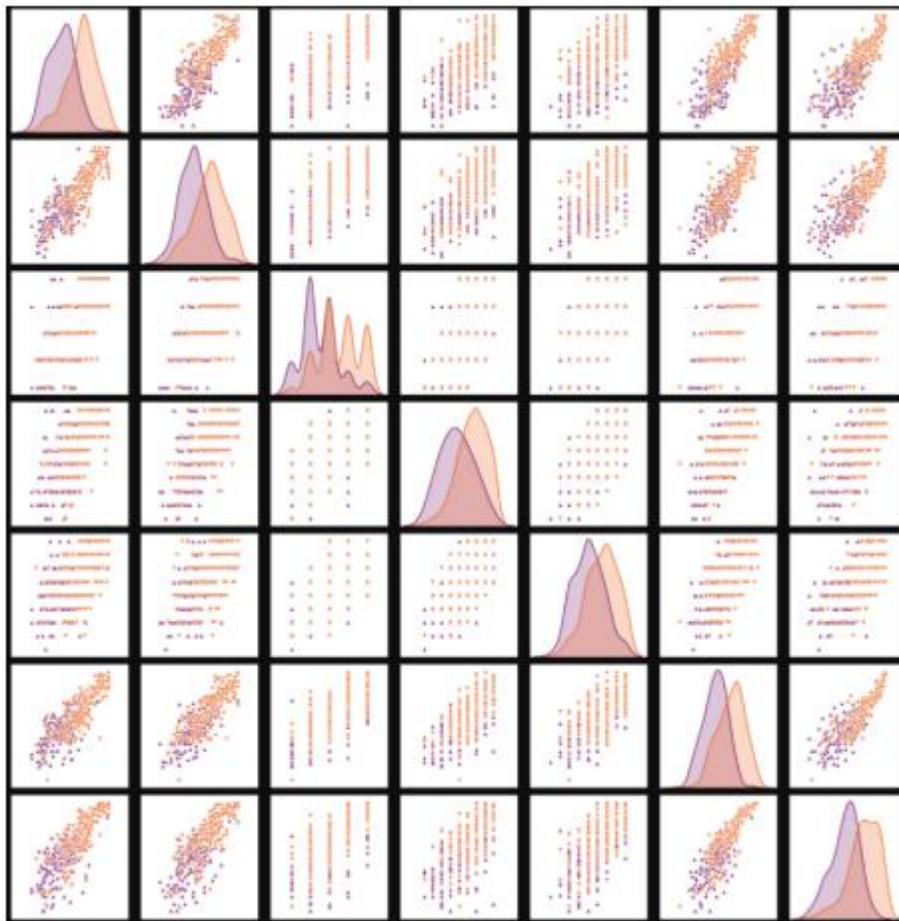
<Axes: >



We see that the output variable "Chance of Admit" depends on CGPA, GRE, TOEFL. The columns SOP, LOR and Research have less impact on university admission

**Pair Plot:** Plot pairwise relationships in a dataset

```
[10] sns.pairplot(data=data,hue='Research',markers=["^","v"],palette='inferno')
```

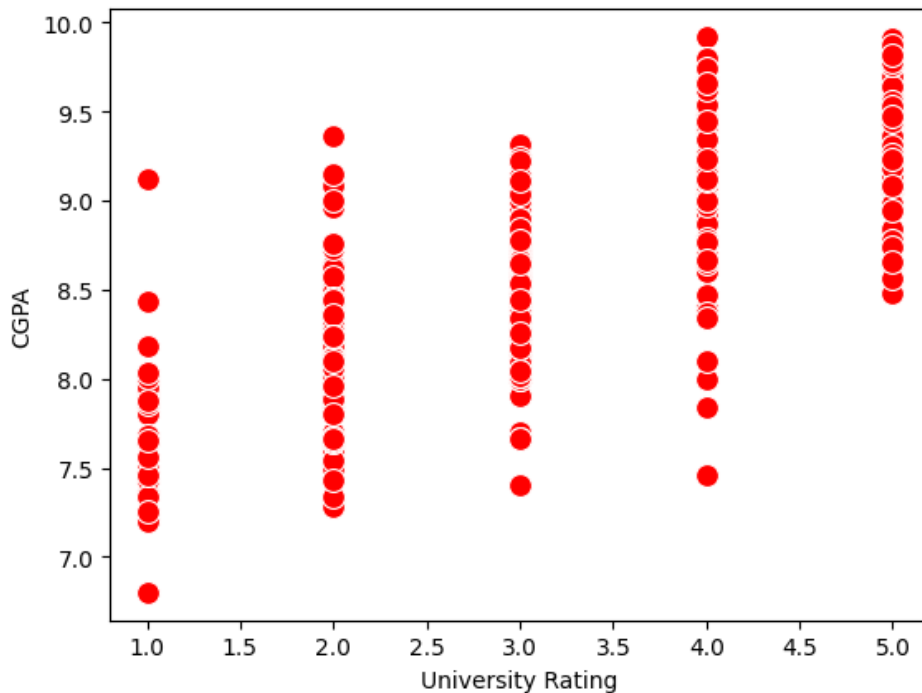


Pair plot usually gives pair wise relationships of the columns in the dataset

1. GRE score TOEFL score and CGPA all are linearly related to each other
2. Students in research score high in TOEFL and GRE compared to non research candidates

**Scatter Plot:** Matplot has a built-in function to create scatterplots called scatter(). A scatter plot is a type of plot that shows the data as a collection of points

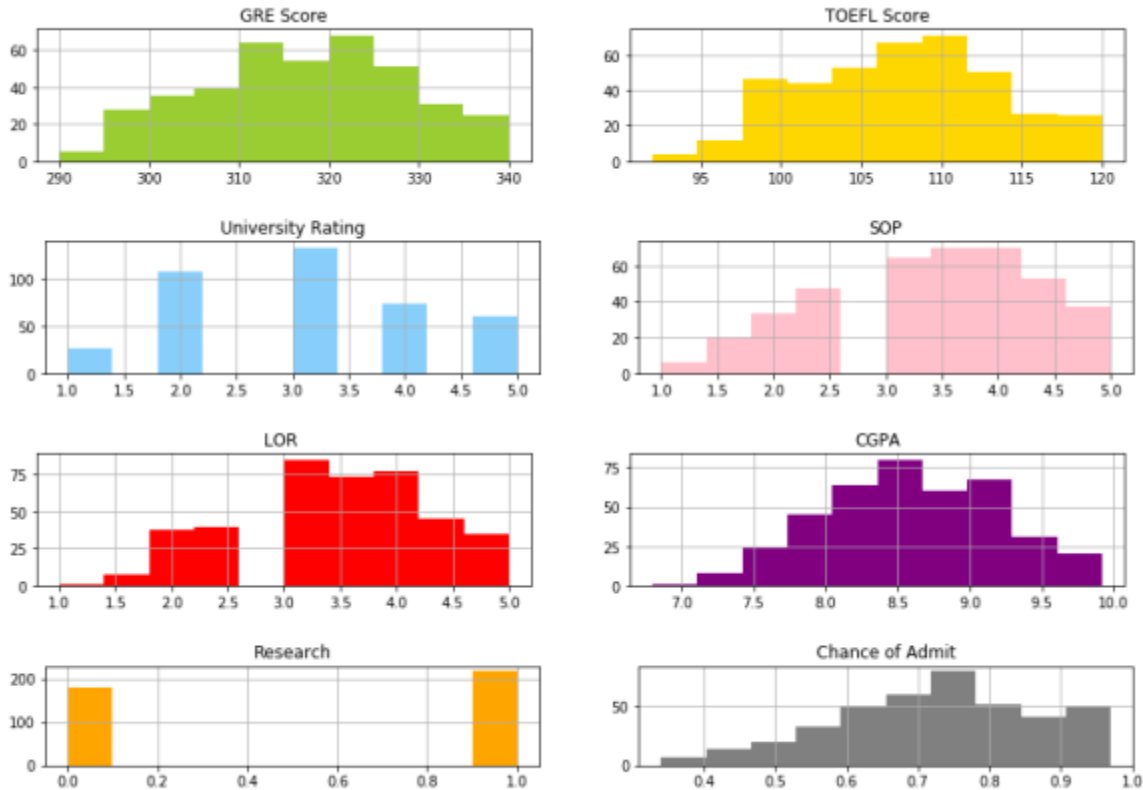
```
sns.scatterplot(x='University Rating',y='CGPA',data=data,color='Red', s=100)
```



Visualizing the Each column in a dataset using subplot( ).

```
category = ['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'CGPA', 'Research', 'Chance of Admit']
color = ['yellowgreen', 'gold', 'lightskyblue', 'pink', 'red', 'purple', 'orange', 'gray']
start = True
for i in np.arange(4):
    fig = plt.figure(figsize=(14,8))
    plt.subplot2grid((4,2),(i,0))
    data[category[2*i]].hist(color=color[2*i],bins=10)
    plt.title(category[2*i])
    plt.subplot2grid((4,2),(i,1))
    data[category[2*i+1]].hist(color=color[2*i+1],bins=10)
    plt.title(category[2*i+1])

plt.subplots_adjust(hspace = 0.7, wspace = 0.2)
plt.show()
```



## Scaling the Data

Scaling is one the important process, we have to perform on the dataset, because of data measures in different ranges can leads to mislead in prediction Models such as KNN, Logistic regression need scaled data, as they follow distance based method and Gradient Descent concept.

```
✓ [15] from sklearn.preprocessing import MinMaxScaler
0s sc = MinMaxScaler()
    x=sc.fit_transform(x)
    x
```

We will perform scaling only on the input values. Once the dataset is scaled, it will be converted into an array and we need to convert it back to a dataframe.

## Splitting data into x and y

Now let's split the Dataset into x and y

```
✓ 1s ▶ x=data.iloc[:,0:7].values
    x
```

```
✓ [14] y=data.iloc[:,7:].values
0s      y
```

Changes: first split the dataset into x and y and then split the data set

Here x and y variables are created. On x variable, df is passed with dropping the target variable. And on y target variable is passed. For splitting training and testing data we are using the train\_test\_split() function from sklearn. As parameters, we are passing x, y, test\_size, random\_state.

```
✓ [16] from sklearn.model_selection import train_test_split
0s      x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.30,random_state=101)
```

Let us convert it into classification problem

chance of admit>0.5 as true chance of admit<0.5 as false

```
✓ [17] y_train=(y_train>0.5)
0s      y_train
```

```
✓ [18] y_test=(y_test>0.5)
0s
```

## CHAPTER 5

### MODEL BUILDING

#### 5.1 Training the model in multiple algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

##### logistic Regression Model

A LogisticRegression algorithm is initialised and training data is passed to the model with the .fit() function. Test data is predicted with .predict() function and saved in a new variable. For evaluating the model, a confusion matrix and classification report is done

```
✓ [34] #Model building - Random Forest Classifier
def RandomForest(x_train,x_test,y_train,y_test):
    rf = RandomForestClassifier(criterion="entropy",n_estimators=10,random_state=0)
    rf.fit(x_train,y_train)
    y_rf_tr = rf.predict(x_train)
    return y_rf_tr
print(accuracy_score(y_rf_tr,y_train))
yPred_rf = rf.predict(x_test)
print(accuracy_score(yPred_rf,y_test))
print("***Random Forest***")
print("Confusion_Matrix")
print(confusion_matrix(y_test,yPred_rf))
print("Classification Report")
print(classification_report(y_test,yPred_rf))
```

##### ANN model

Building and training an Artificial Neural Network (ANN) using the Keras library with TensorFlow as the backend. The ANN is initialised as an instance of the Sequential class, which is a linear stack of layers. Then, the input layer and two hidden layers are added to the model using the Dense class, where the number of units and activation function are specified. The output layer is also added using the Dense class with a sigmoid activation function. The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy metric. Finally, the model is fit to the training data with a batch size of 100, 20% validation split, and 100 epochs.

```
✓ [23] import tensorflow as tf
0s from tensorflow import keras
from tensorflow.keras.layers import Dense,Activation,Dropout
from tensorflow.keras.optimizers import Adam
```

```

✓ [38] import keras
      from keras.models import Sequential
      from keras.layers import Dense

✓ [39] classifier = Sequential()

✓ ▶ classifier.add(Dense(units=7, activation='relu', input_dim = 7))

✓ [41] classifier.add(Dense(units=7, activation='relu'))

✓ [42] classifier.add(Dense(units=1, activation='linear'))

✓ [43] classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

✓ [44] model = classifier.fit(x_train, y_train, batch_size=10, validation_split=0.33, epochs=20)
3s
Epoch 1/20
22/22 ----- 2s 13ms/step - accuracy: 0.0706 - loss: 12.1798 - val_accuracy: 0.0849 - val_loss: 10.6656
Epoch 2/20
22/22 ----- 0s 5ms/step - accuracy: 0.0762 - loss: 7.9947 - val_accuracy: 0.0849 - val_loss: 2.5649
Epoch 3/20
22/22 ----- 0s 5ms/step - accuracy: 0.0743 - loss: 2.2759 - val_accuracy: 0.0849 - val_loss: 1.8789
Epoch 4/20
22/22 ----- 0s 8ms/step - accuracy: 0.0815 - loss: 1.8372 - val_accuracy: 0.0849 - val_loss: 1.6426
Epoch 5/20
22/22 ----- 0s 8ms/step - accuracy: 0.0748 - loss: 1.6282 - val_accuracy: 0.0849 - val_loss: 1.4740
Epoch 6/20
22/22 ----- 0s 8ms/step - accuracy: 0.0641 - loss: 1.4973 - val_accuracy: 0.0849 - val_loss: 1.3294
Epoch 7/20
22/22 ----- 0s 8ms/step - accuracy: 0.0492 - loss: 1.3411 - val_accuracy: 0.0849 - val_loss: 1.2018
Epoch 8/20
22/22 ----- 0s 8ms/step - accuracy: 0.0920 - loss: 1.2022 - val_accuracy: 0.0849 - val_loss: 1.0869
Epoch 9/20
22/22 ----- 0s 8ms/step - accuracy: 0.0973 - loss: 1.0875 - val_accuracy: 0.1698 - val_loss: 0.9813
Epoch 10/20
22/22 ----- 0s 9ms/step - accuracy: 0.1742 - loss: 0.9866 - val_accuracy: 0.3113 - val_loss: 0.8795
Epoch 11/20
22/22 ----- 0s 9ms/step - accuracy: 0.2459 - loss: 0.8765 - val_accuracy: 0.4340 - val_loss: 0.7864
Epoch 12/20
22/22 ----- 0s 7ms/step - accuracy: 0.4569 - loss: 0.7529 - val_accuracy: 0.5377 - val_loss: 0.6975
Epoch 13/20
22/22 ----- 0s 6ms/step - accuracy: 0.5443 - loss: 0.6902 - val_accuracy: 0.6321 - val_loss: 0.6145
Epoch 14/20
22/22 ----- 0s 6ms/step - accuracy: 0.5897 - loss: 0.6480 - val_accuracy: 0.7453 - val_loss: 0.5349
Epoch 15/20
22/22 ----- 0s 5ms/step - accuracy: 0.7050 - loss: 0.5491 - val_accuracy: 0.7830 - val_loss: 0.4599
Epoch 16/20
22/22 ----- 0s 5ms/step - accuracy: 0.7803 - loss: 0.4686 - val_accuracy: 0.8113 - val_loss: 0.3915

```

## 5.2 Testing the model

In ANN we first have to save the model to the test the inputs.

```

✓ [31] model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
0s

✓ [32] from sklearn.metrics import accuracy_score
0s

✓ [33] train_predictions = model.predict(x_train)
0s
9/9 [=====] - 0s 2ms/step

✓ [34] print(train_predictions)
0s

```



```
✓ [33] train_acc = model.evaluate(x_train, y_train, verbose=0)[1]  
2s
```

```
✓ [34] print(train_acc)  
0s  
0.2410714328289032
```

```
✓ [35] test_acc = model.evaluate(x_test, y_test, verbose=0)[1]  
0s
```

```
✓ [36] print(test_acc)  
0s  
0.3375000059604645
```

```
✓ [37] pred=model.predict(x_test)  
0s  
pred = (pred>0.5)  
pred
```

# CHAPTER 6

## PERFORMANCE TESTING & HYPERPARAMETER TUNING

### 6.1 Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

### 6.2 Comparing model accuracy before & after applying hyperparameter tuning:

#### Compare the model

For comparing the above four models, the compareModel function is defined.

#### Logistics Regression model

```
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score, confusion_matrix
print("\nAccuracy score: %f" %(accuracy_score(y_test, y_pred) * 100))
print("Recall score : %f" %(recall_score(y_test, y_pred) * 100))
print("ROC score : %f\n" %(roc_auc_score(y_test, y_pred) * 100))
print(confusion_matrix(y_test, y_pred))
```

#### ANN Model : Training Accuracy

```
[45] ann_pred = classifier.predict(x_test)
ann_pred = (ann_pred>0.5)
print(accuracy_score(ann_pred,y_test))
print("***ANN Model***")
print("Confusion Matrix")
print(confusion_matrix(y_test,ann_pred))
print("Classification Report")
print(classification_report(y_test,ann_pred))
```

3/3 0s 23ms/step

0.8625

\*\*\*ANN Model\*\*\*

Confusion Matrix

[[ 0 10]  
 [ 1 69]]

Classification Report

	precision	recall	f1-score	support
False	0.00	0.00	0.00	10
True	0.87	0.99	0.93	70
accuracy			0.86	80
macro avg	0.44	0.49	0.46	80
weighted avg	0.76	0.86	0.81	80

the results of models are displayed as output. From the both models ANN is performing well. From the below image, We can see the accuracy of the model. ANN is giving the accuracy of 86% with training data , 88% accuracy for the testing data.

## CHAPTER 7

### MODEL DEPLOYMENT

#### 7.1 Save the best model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.

```
✓ [39] model.save('model.h5')
```

#### 7.2 Integrate with Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the user where he has to enter the values for predictions. The entered values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building DATABASE
- Building server side script
- Run the web application

##### **Building Html Pages:**

For this project create six HTML files namely

- home.html
- signin.html
- login.html
- predict.html
- chance.html
- nochance.html

and save them in the templates folder.

##### **Build Python code:**

Import the libraries

```
import numpy as np
import pickle
import pymysql # Import MySQL library
from flask import Flask, request, render_template, redirect, url_for
from flask_mysqlldb import MySQL # For MySQL integration
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (`__name__`) as argument.

```
# Load model and scaler
model = pickle.load(open("university.pkl", "rb"))
scaler = pickle.load(open("scaler.pkl", "rb"))
```

## Render HTML page:

```
@app.route("/")
def home():
    return render_template("home.html")

@app.route("/register", methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        try:
            username = request.form["username"]
            email = request.form["email"]
            password = request.form["password"]

            # Save to MySQL database
            cursor = mysql.connection.cursor()
            cursor.execute("INSERT INTO user_id (username, email, password) VALUES (%s, %s, %s)",
                           (username, email, password))
            mysql.connection.commit()
            cursor.close()
            print("User registration successful!")

            return redirect(url_for("login"))
        except Exception as e:
            print("Error inserting data:", e)

    return render_template("register.html")

@app.route("/login", methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form["email"]
        password = request.form["password"]

        # Check user in database
        cursor = mysql.connection.cursor()
        cursor.execute("SELECT * FROM user_id WHERE email = %s AND password = %s",
                       (email, password))
        user = cursor.fetchone()
        cursor.close()

        if user:
            return redirect(url_for("predict"))
        else:
            return "Invalid login, try again!"

    return render_template("login.html")
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

### Retrieves the value from UI:

```
if request.method == 'POST':
    score = float(request.form["score"])
    prediction = model.predict(np.array([[score]]))

    if prediction[0] == 1:
        return redirect(url_for("chance"))
    else:
        return redirect(url_for("nochance"))

return render_template("prediction.html")

@app.route("/chance")
def chance():
    return render_template("chance.html")

@app.route("/nochance")
def nochance():
    return render_template("nochance.html")

if __name__ == "__main__":
    app.run(debug=True)
```

Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

### Main Function:

```
if __name__ == "__main__":
    app.run(debug=True)
```

### Run the web application

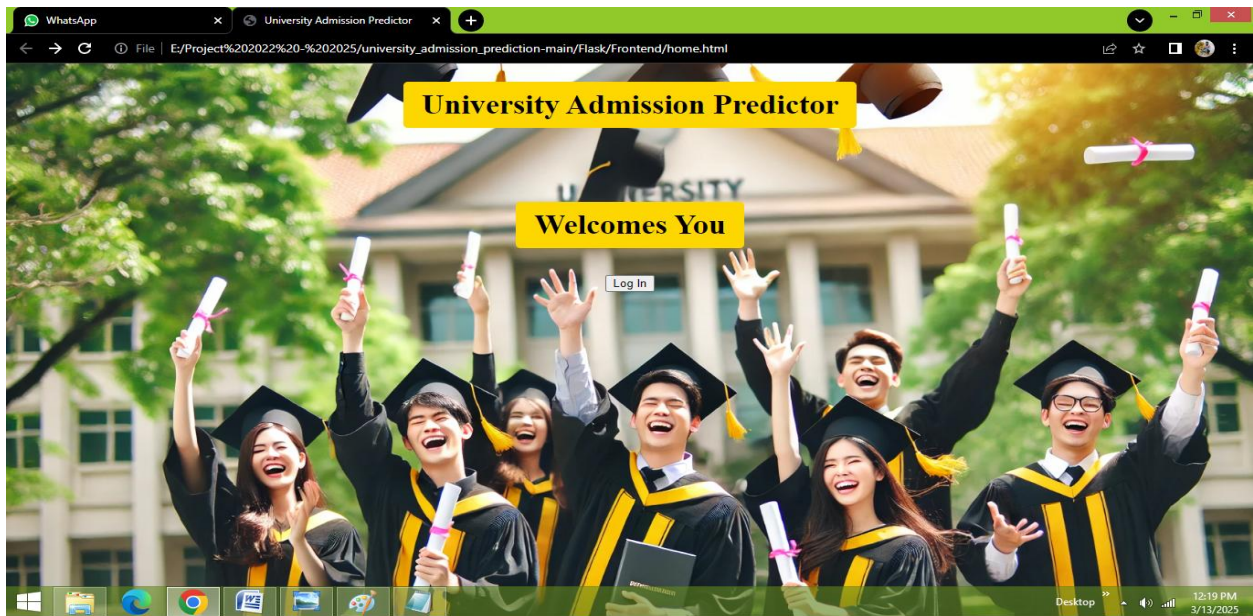
- Save the all files in flask\_project to run
- Open command prompt and write 'cd flask\_project' click enter
- Run the app.py file on prompt to run the application
- Navigate to the localhost where you can view your web page.

```
C:\Windows\system32\cmd.exe - app.py
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Nandhakumar S>cd flask_project

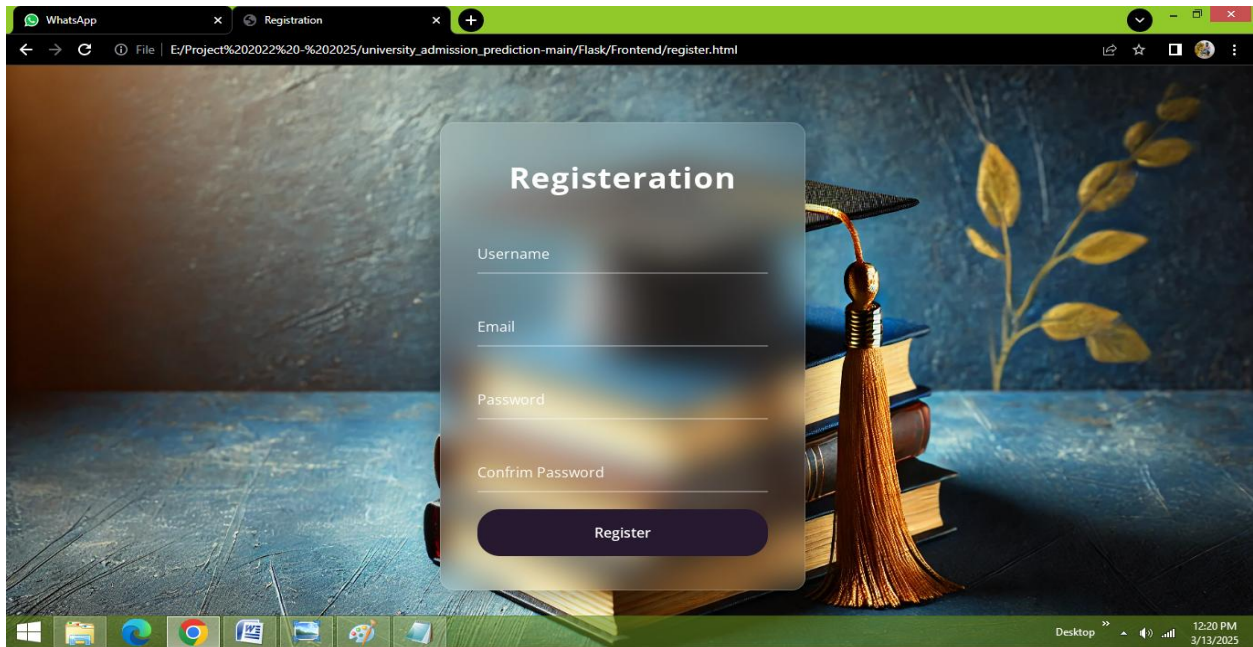
C:\Users\Nandhakumar S\flask_project>app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 849-707-694
```

Now,Go the web browser and write the localhost url (<http://127.0.0.1:5000>) to get the below result

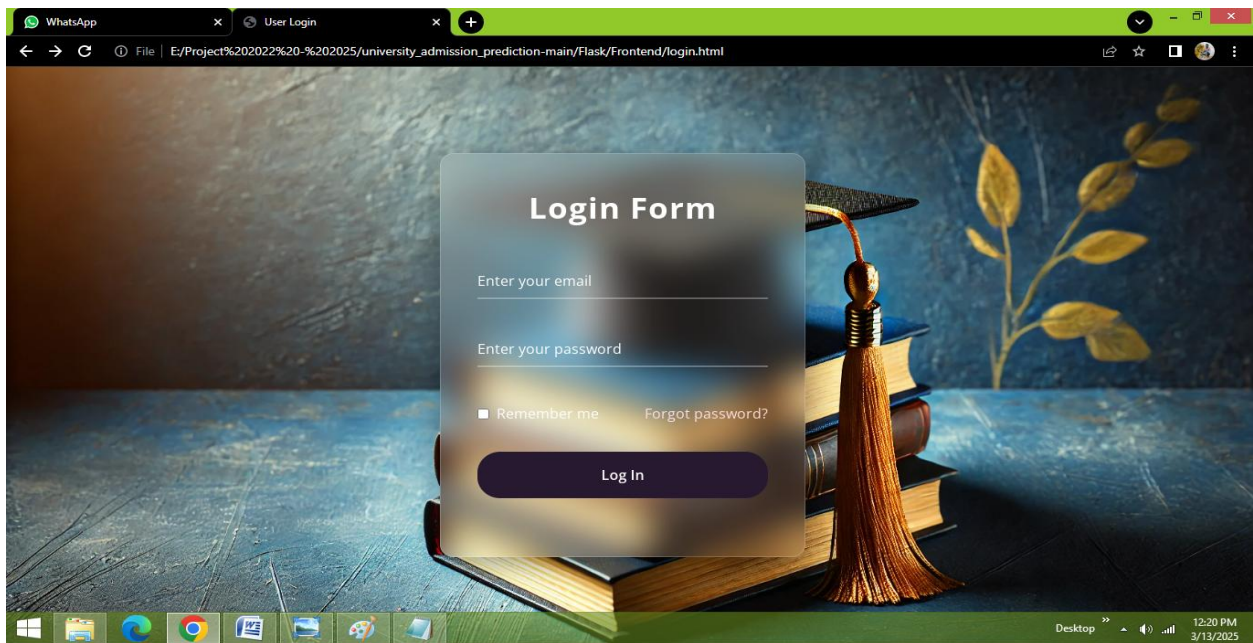


Now,click on the Sign in button to go the register page given below result



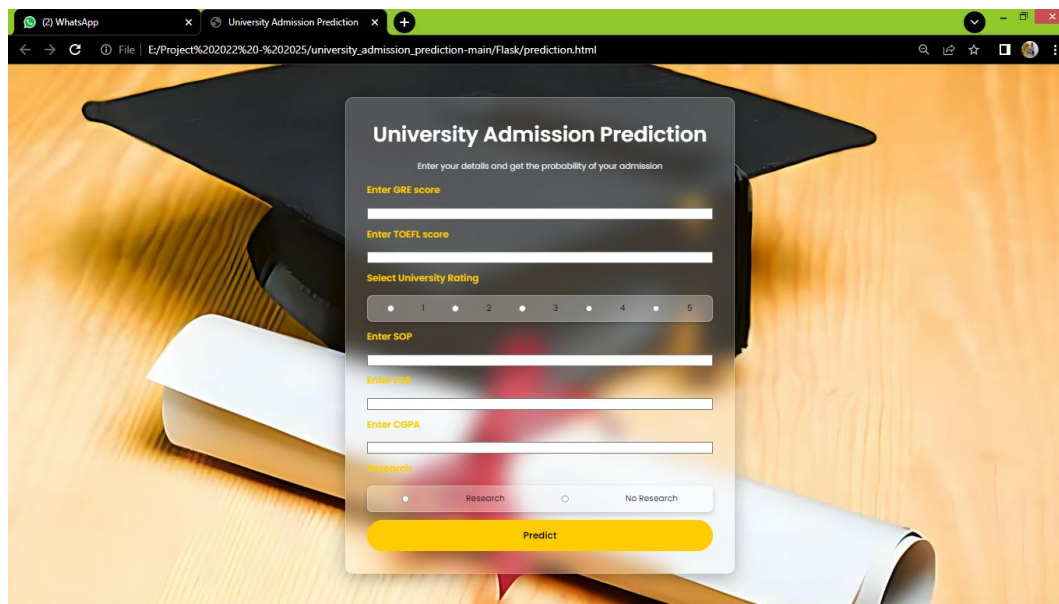


Now, input the values to register, after given the values click on the Register button its go to login page given below



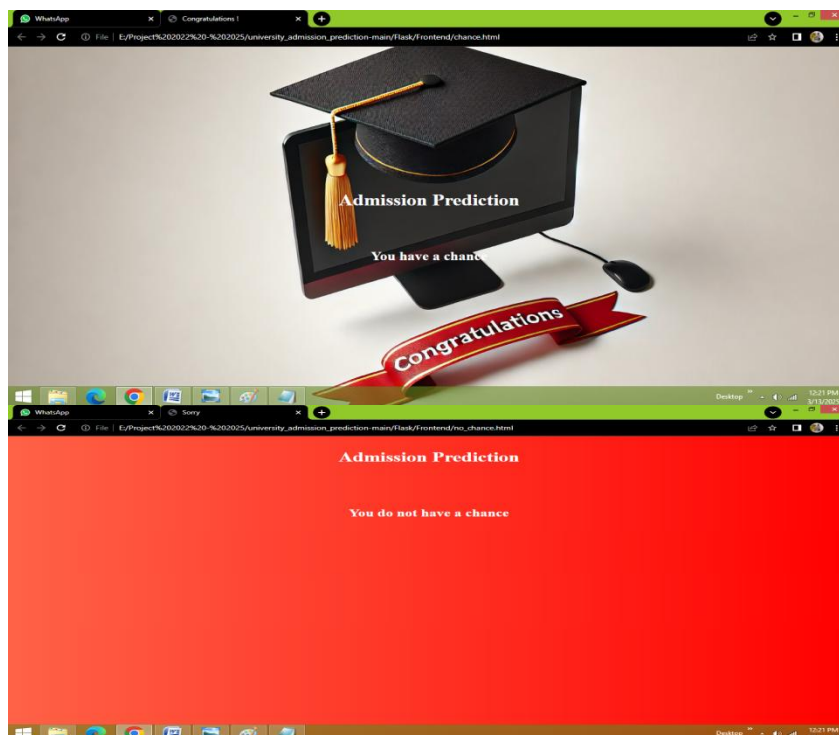


Now, click one the Login button to go Prediction page to predict the admission chance given bellow result



The screenshot shows a web browser window with a tab titled 'University Admission Prediction'. The URL is 'E:/Project%202022%20-%202025/university\_admission\_prediction-main/Flask/prediction.html'. The page features a background image of a graduation cap and books. A dark grey form titled 'University Admission Prediction' is centered on the page. The form includes the instruction 'Enter your details and get the probability of your admission' and the following fields: 'Enter GRE score', 'Enter TOEFL score', 'Select University Rating' (with a radio button selector for 1 to 5), 'Enter SOP', 'Enter ECE', 'Enter CGPA', and 'Research' (with radio buttons for 'Research' and 'No Research'). A yellow 'Predict' button is at the bottom of the form.

Input 1- Now, the user will give inputs to get the predicted result after clicking onto the predict button



## **CHAPTER 8**

### **CONCLUSION**

University admission is the process by which students are selected to attend a college or university. The process typically involves several steps, including submitting an application, taking entrance exams, and participating in interviews or other evaluations. Students are often worried about their chances of admission in University. the university admission process for students can be demanding, but by being well-informed, prepared, and organized, students can increase their chances of being admitted to the university of their choice. The aim of this project is to help students in short listing universities with their profiles. Machine learning algorithms are then used to train a model on this data, which can be used to predict the chances of future applicants being admitted. With this project, students can make more informed decisions about which universities to apply to, and universities can make more efficient use of their resources by focusing on the most promising applicants. The predicted output gives them a fair idea about their admission chances in a particular university. This analysis should also help students who are currently preparing or will be preparing to get a better idea.

### **REFERENCES**

1. The Hundred-Page Machine Learning Book by Andriy Burkov
2. Machine Learning For Absolute Beginners by Oliver Theobald
3. Machine Learning for Hackers by Drew Conway and John Myles White
4. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Geron Aurelien
5. Deep Learning by Ian Goodfellow, Yoshua Bengio and Aaron Courville