**Project Title**

**Intelligent Admissions: The Future of University Decision Making with Machine Learning**

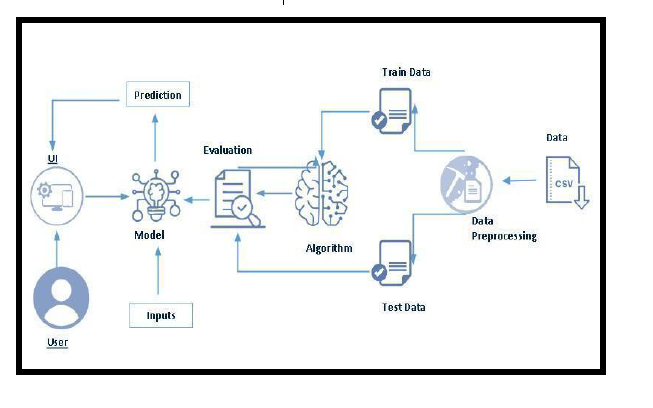
**Project Description**

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**



**Project Flow:**

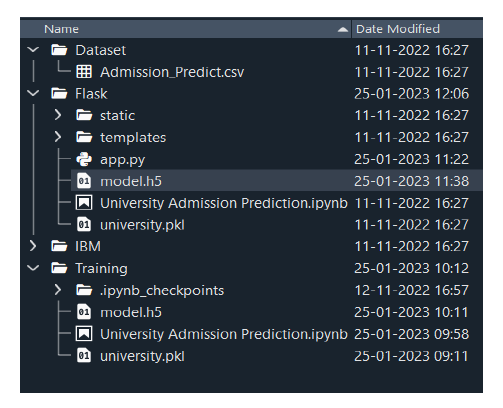
* User interacts with the UI to enter the input.
* Entered input is analyzed by the model which is integrated.
* Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities listed below,

1. Define Problem / Problem Understanding
   1. Specify the business problem
   2. Business requirements
   3. Literature Survey
   4. Social or Business Impact.
2. Data Collection & Preparation
   1. Collect the dataset
   2. Data Preparation
3. Exploratory Data Analysis
   1. Descriptive statistical
   2. Visual Analysis
4. Model Building
   1. Training the model in multiple algorithms
   2. Testing the model
5. Performance Testing & Hyperparameter Tuning
   1. Testing model with multiple evaluation metrics
   2. Comparing model accuracy before & after applying hyperparameter tuning
6. Model Deployment
   1. Save the best model
   2. Integrate with Web Framework
7. Project Demonstration & Documentation
   1. Record explanation Video for project end to end solution
   2. Project Documentation-Step by step project development procedure

**Project Structure**

Create the Project folder which contains files as shown below



* We are building a flask application which needs HTML pages stored in the templates folder and a python script app.py for scripting.
* model.h5 is our saved model. Further we will use this model for flask integration.
* Training folder contains a model training file.

**Milestone 1: Define Problem / Problem Understanding**

**Activity 1: Specify the business problem**

**Activity 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.

**Activity 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.

**Activity 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.

**Milestone 2: Data Collection & Preparation**

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

**Activity 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.

**Activity 1.1: 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

**Activity 1.2: 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.

#read\_csv is a pandas function to read csv files

data=pd.read\_csv('E:\\NMDS\Admission\_Predict.csv')

data.head()

Serial No. GRE Score TOEFL Score University Rating SOP LOR CGPA Research Chance of Admit

0 1 337 118 4 4.5 4.5 9.65 1 0.92

1 2 324 107 4 4.0 4.5 8.87 1 0.76

2 3 316 104 3 3.0 3.5 8.00 1 0.72

3 4 322 110 3 3.5 2.5 8.67 1 0.80

4 5 314 103 2 2.0 3.0 8.21 0 0.65

**Activity 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.

**Activity 2.1: 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.

*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

*data.shape*

(400, 9)

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.

*data.isnull().any()*

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

*data.isnull().sum()*

Serial No. 0

GRE Score 0

TOEFL Score 0

University Rating 0

SOP 0

LOR 0

CGPA 0

Research 0

Chance of Admit 0

dtype: int64

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

*#let us rename the column chance of Admit because it has trainling space*

*data=data.rename(columns={'chance of Admit':'chance of Admit'})*

**Milestone 3: Exploratory Data Analysis**

**Activity 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()*

erial 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 0.97000

**Activity 2: Visual analysis**

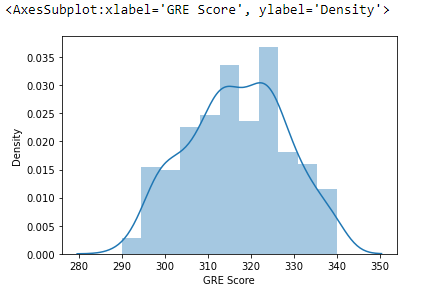
Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions.

**Activity 2.1: 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.

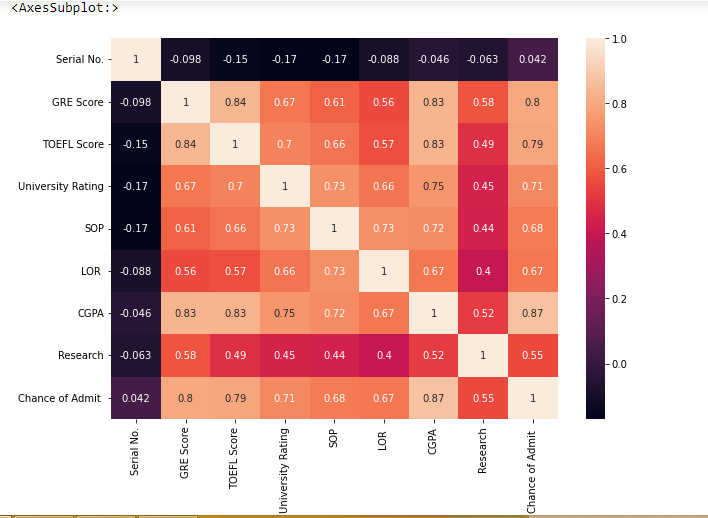
*sns.distplot(data['GRE Score'])*



**Activity 2.2: Bivariate analysis**

*plt.figure(figsize=(10,7))*

*sns.heatmap(data.corr(),annot=True***)**

****

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

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

*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*

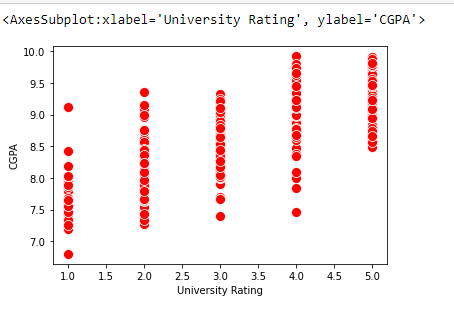
*2. Students in research score high in TOEFL and GRE compared to non research*

*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','LOR','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])*

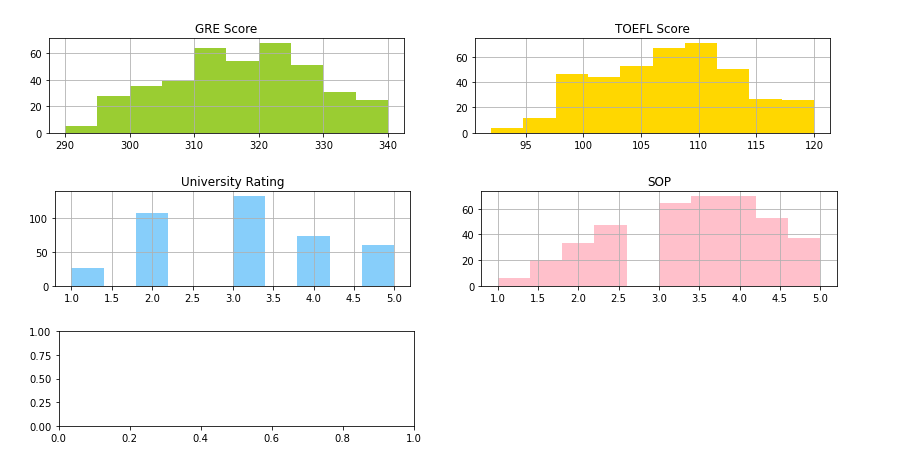
*plt.subplot2grid((4,2),(i,1))*

*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.*

*from sklearn.preprocessing import MinMaxScaler*

*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*

*x=data.iloc[:,0:7].values*

*x*

array([[ 1. , 337. , 118. , ..., 4.5 , 4.5 , 9.65],

[ 2. , 324. , 107. , ..., 4. , 4.5 , 8.87],

[ 3. , 316. , 104. , ..., 3. , 3.5 , 8. ],

...,

[398. , 330. , 116. , ..., 5. , 4.5 , 9.45],

[399. , 312. , 103. , ..., 3.5 , 4. , 8.78],

[400. , 333. , 117. , ..., 5. , 4. , 9.66]])

*y=data.iloc[:,7:].values*

*y*

array([[1. , 0.92],

[1. , 0.76],

[1. , 0.72],

[1. , 0.8 ],

[0. , 0.65],

[1. , 0.9 ],

[1. , 0.75],

[0. , 0.68],

[0. , 0.5 ],

[0. , 0.45],

[1. , 0.52],

[1. , 0.84],

[1. , 0.78],

[1. , 0.62],

[1. , 0.61],

[0. , 0.54],

[0. , 0.66],

[1. , 0.65],

[0. , 0.63],

[0. , 0.62],

[1. , 0.64],

[0. , 0.7 ],

[1. , 0.94],

[1. , 0.95],

[1. , 0.97],

[1. , 0.94],

[0. , 0.76],

[1. , 0.44],

[0. , 0.46],

[0. , 0.54],

[1. , 0.65],

[1. , 0.74],

[1. , 0.91],

[1. , 0.9 ],

[1. , 0.94],

[1. , 0.88],

[0. , 0.64],

[0. , 0.58],

[0. , 0.52],

[0. , 0.48],

[1. , 0.46],

[1. , 0.49],

[1. , 0.53],

[0. , 0.87],

[0. , 0.56],

[0. , 0.45],

[0. , 0.47],

[1. , 0.71],

[1. , 0.94],

[1. , 0.94],

[0. , 0.57],

[0. , 0.61],

[0. , 0.57],

[1. , 0.64],

[1. , 0.85],

[1. , 0.78],

[1. , 0.84],

[1. , 0.92],

[1. , 0.96],

[0. , 0.77],

[0. , 0.71],

[0. , 0.79],

[1. , 0.89],

[1. , 0.82],

[0. , 0.76],

[1. , 0.71],

[1. , 0.8 ],

[0. , 0.78],

[1. , 0.84],

[1. , 0.9 ],

[1. , 0.92],

[1. , 0.97],

[1. , 0.8 ],

[1. , 0.81],

[0. , 0.75],

[1. , 0.83],

[1. , 0.96],

[1. , 0.79],

[1. , 0.93],

[1. , 0.94],

[1. , 0.86],

[0. , 0.79],

[0. , 0.8 ],

[0. , 0.77],

[0. , 0.7 ],

[0. , 0.65],

[0. , 0.61],

[0. , 0.52],

[0. , 0.57],

[0. , 0.53],

[0. , 0.67],

[0. , 0.68],

[1. , 0.81],

[0. , 0.78],

[0. , 0.65],

[0. , 0.64],

[1. , 0.64],

[0. , 0.65],

[1. , 0.68],

[1. , 0.89],

[1. , 0.86],

[1. , 0.89],

[1. , 0.87],

[1. , 0.85],

[1. , 0.9 ],

[0. , 0.82],

[0. , 0.72],

[0. , 0.73],

[0. , 0.71],

[0. , 0.71],

[0. , 0.68],

[0. , 0.75],

[0. , 0.72],

[1. , 0.89],

[1. , 0.84],

[1. , 0.93],

[1. , 0.93],

[1. , 0.88],

[1. , 0.9 ],

[1. , 0.87],

[1. , 0.86],

[1. , 0.94],

[0. , 0.77],

[1. , 0.78],

[0. , 0.73],

[0. , 0.73],

[0. , 0.7 ],

[0. , 0.72],

[1. , 0.73],

[1. , 0.72],

[1. , 0.97],

[1. , 0.97],

[0. , 0.69],

[0. , 0.57],

[0. , 0.63],

[1. , 0.66],

[0. , 0.64],

[1. , 0.68],

[1. , 0.79],

[1. , 0.82],

[1. , 0.95],

[1. , 0.96],

[1. , 0.94],

[1. , 0.93],

[1. , 0.91],

[1. , 0.85],

[1. , 0.84],

[0. , 0.74],

[0. , 0.76],

[0. , 0.75],

[0. , 0.76],

[0. , 0.71],

[0. , 0.67],

[0. , 0.61],

[0. , 0.63],

[0. , 0.64],

[0. , 0.71],

[1. , 0.82],

[0. , 0.73],

[1. , 0.74],

[0. , 0.69],

[0. , 0.64],

[1. , 0.91],

[1. , 0.88],

[1. , 0.85],

[1. , 0.86],

[0. , 0.7 ],

[0. , 0.59],

[0. , 0.6 ],

[0. , 0.65],

[1. , 0.7 ],

[1. , 0.76],

[0. , 0.63],

[1. , 0.81],

[0. , 0.72],

[0. , 0.71],

[1. , 0.8 ],

[1. , 0.77],

[1. , 0.74],

[0. , 0.7 ],

[1. , 0.71],

[1. , 0.93],

[0. , 0.85],

[0. , 0.79],

[0. , 0.76],

[1. , 0.78],

[1. , 0.77],

[1. , 0.9 ],

[1. , 0.87],

[0. , 0.71],

[1. , 0.7 ],

[1. , 0.7 ],

[1. , 0.75],

[0. , 0.71],

[0. , 0.72],

[1. , 0.73],

[0. , 0.83],

[0. , 0.77],

[1. , 0.72],

[0. , 0.54],

[0. , 0.49],

[1. , 0.52],

[0. , 0.58],

[1. , 0.78],

[1. , 0.89],

[0. , 0.7 ],

[0. , 0.66],

[0. , 0.67],

[1. , 0.68],

[1. , 0.8 ],

[1. , 0.81],

[1. , 0.8 ],

[1. , 0.94],

[1. , 0.93],

[1. , 0.92],

[1. , 0.89],

[0. , 0.82],

[0. , 0.79],

[0. , 0.58],

[0. , 0.56],

[0. , 0.56],

[1. , 0.64],

[1. , 0.61],

[0. , 0.68],

[0. , 0.76],

[0. , 0.86],

[1. , 0.9 ],

[0. , 0.71],

[0. , 0.62],

[0. , 0.66],

[1. , 0.65],

[1. , 0.73],

[0. , 0.62],

[1. , 0.74],

[1. , 0.79],

[1. , 0.8 ],

[0. , 0.69],

[0. , 0.7 ],

[1. , 0.76],

[1. , 0.84],

[1. , 0.78],

[0. , 0.67],

[0. , 0.66],

[0. , 0.65],

[0. , 0.54],

[0. , 0.58],

[1. , 0.79],

[1. , 0.8 ],

[1. , 0.75],

[1. , 0.73],

[0. , 0.72],

[0. , 0.62],

[0. , 0.67],

[1. , 0.81],

[0. , 0.63],

[0. , 0.69],

[1. , 0.8 ],

[0. , 0.43],

[1. , 0.8 ],

[1. , 0.73],

[1. , 0.75],

[1. , 0.71],

[1. , 0.73],

[1. , 0.83],

[0. , 0.72],

[1. , 0.94],

[1. , 0.81],

[1. , 0.81],

[1. , 0.75],

[1. , 0.79],

[0. , 0.58],

[0. , 0.59],

[0. , 0.47],

[0. , 0.49],

[0. , 0.47],

[0. , 0.42],

[0. , 0.57],

[0. , 0.62],

[1. , 0.74],

[1. , 0.73],

[1. , 0.64],

[0. , 0.63],

[0. , 0.59],

[0. , 0.73],

[1. , 0.79],

[1. , 0.68],

[0. , 0.7 ],

[0. , 0.81],

[1. , 0.85],

[1. , 0.93],

[1. , 0.91],

[0. , 0.69],

[1. , 0.77],

[1. , 0.86],

[1. , 0.74],

[0. , 0.57],

[0. , 0.51],

[1. , 0.67],

[0. , 0.72],

[1. , 0.89],

[1. , 0.95],

[1. , 0.79],

[0. , 0.39],

[0. , 0.38],

[0. , 0.34],

[0. , 0.47],

[0. , 0.56],

[1. , 0.71],

[1. , 0.78],

[1. , 0.73],

[1. , 0.82],

[0. , 0.62],

[1. , 0.96],

[1. , 0.96],

[0. , 0.46],

[0. , 0.53],

[0. , 0.49],

[1. , 0.76],

[0. , 0.64],

[0. , 0.71],

[1. , 0.84],

[0. , 0.77],

[1. , 0.89],

[1. , 0.82],

[1. , 0.84],

[1. , 0.91],

[0. , 0.67],

[1. , 0.95]])

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.

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

*x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=0.30,random\_state=101)*

*#random\_state acts as the seed for the random number generator during the split*

***Let us convert it into classification problem***

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

*y\_train=(y\_train>0.5)*

*y\_train*

*y\_test=(y\_test>0.5)*

***Milestone 4: Model Building***

***Activity 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.*

***Activity 1.1: 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*

*from sklearn.linear\_model.logistic import LogisticRegression*

*cls =LogisticRegression(random\_state =0)*

*lr=cls.fit(x\_train,y\_train)*

*c:\Users\Tulasi\anaconda3\lib\site.packages\sklearn\utils\validation.py:760: DataConversionwarn*

*array was expected.please change the shape of y to(n\_samples,), for example using ravel().*

*y = column\_or\_1d(y,warn=true)*

*y\_pred =lr.predict(x\_test)*

*y\_pred*

**Activity 1.5: 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.

*#Libraries to train Neural network*

*import tensorflow as tf*

*from tensorflow import keras*

*from tensorflow.keras.layers import Dense,Activation,Dropout*

*from tensorfrom.keras.optimizers import Adam*

*#Initialize the model*

*model=Keras.Sequential()*

*#Add input layer*

*model.add(Dense(7,activation ='relu',input\_dim=7))*

*#Add hidden layer*

*model.add(Dense(7,activation ='relu'))*

*#Add output layer*

*model.add(Dense(1,activation ='linear'))*

*model.summary()*

*model: "sequential"*

*model.summary()*

*model: "sequential"*

*model.fit(x\_train, y\_train, batch\_size = 20, epochs = 100)*

*model.compile(loss = 'binary\_crossentropy', optimizer = 'adam',metics = ['accuracy'])*

*model.fit(x\_train, y\_train, batch\_size = 20, epochs = 100)*

*from sklearn.metrics import accuracy\_score*

**Activity 2: Testing the model**

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

*#make predictions on the training data*

*train\_predictions = model.predict(x\_train)*

*print(train\_predictions)*

*# Get the training accuracy*

*train\_acc = model.evaluate(x\_train, y\_train,verbos=0)[1]*

*print(train\_acc)*

*#Get the test accuracy*

*test\_acc = model.evaluate(x\_test, y\_test, verbose=0)[1]*

*print(test\_acc)*

*print(classification report(v test,pred))*

*pred=model.predict(x\_test)*

*pred = (pred>0.5)*

*pred*

**Milestone 5: Performance Testing & Hyperparameter Tuning**

**Activity 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.

**Activity 1.1: 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" %(roc\_auc\_score(y\_test,y\_pred)\*100))

print(confusion\_matrix(y\_test,y\_pred))

from sklearn.metrics import accuracy\_score

#make predictions on the training data

train\_predictions = model.predict(x\_train)

print(train\_predictions)

#Get the test accuracy

test\_acc = model.evaluate(x\_test, y\_test, verbose=0)[1]

print(test\_acc)

print(classification report(v test,pred))

pred=model.predict(x\_test)

pred = (pred>0.5)

pred

**Milestone 6: Model Deployment**

**Activity 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.

#save the model in h5 format

model.save('model5.h')

**Activity 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 uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

* Building HTML Pages
* Building server side script
* Run the web application

**Activity 2.1: Building Html Pages:**

For this project create two HTML files namely

* home.html
* predict.html

and save them in the templates folder.

**Activity 2.2: Build Python code:**

Import the libraries

import numpy as np

from flask import Flask,request,jsonify, render\_teplate

import pickle

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

#import necessary libraries

from.tensorflow.keras.models import load\_model

#model=pickle.load(open('university.pkl','rb'))

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 trained model

load.model('model.h5')

Render HTML page:

@pp.route('/')

def.home():

retun render\_template('Demo2.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.