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

**ABSTRACT**

* 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:**

Create the Project folder which contains files as shown below



**Project Flow:**

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

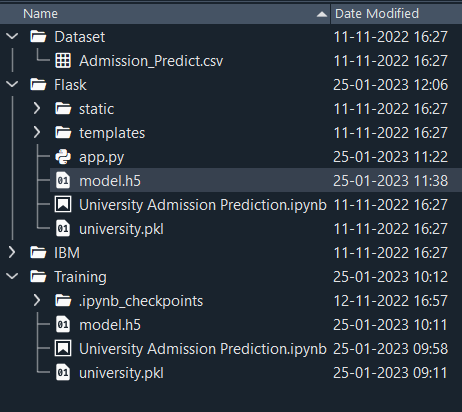
To accomplish this, we have to complete all the activities listed below,

* Define Problem / Problem Understanding
  + Specify the business problem
  + Business requirements
  + Literature Survey
  + Social or Business Impact.

* Data Collection & Preparation
  + Collect the dataset
  + Data Preparation
* Exploratory Data Analysis
  + Descriptive statistical
  + Visual Analysis
* Model Building
  + Training the model in multiple algorithms
  + Testing the model
* Performance Testing & Hyper parameter Tuning
  + Testing model with multiple evaluation metrics
  + Comparing model accuracy before & after applying hyperparameter tuning
* Model Deployment
  + Save the best model
  + Integrate with Web Framework
* Project Demonstration & Documentation
  + Record explanation Video for project end to end solution
  + 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.

**Mile stone 1**

### Specify The Business Problem

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.

**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.

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

### 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.

### Social Or Business Impact

**Social Impact:**

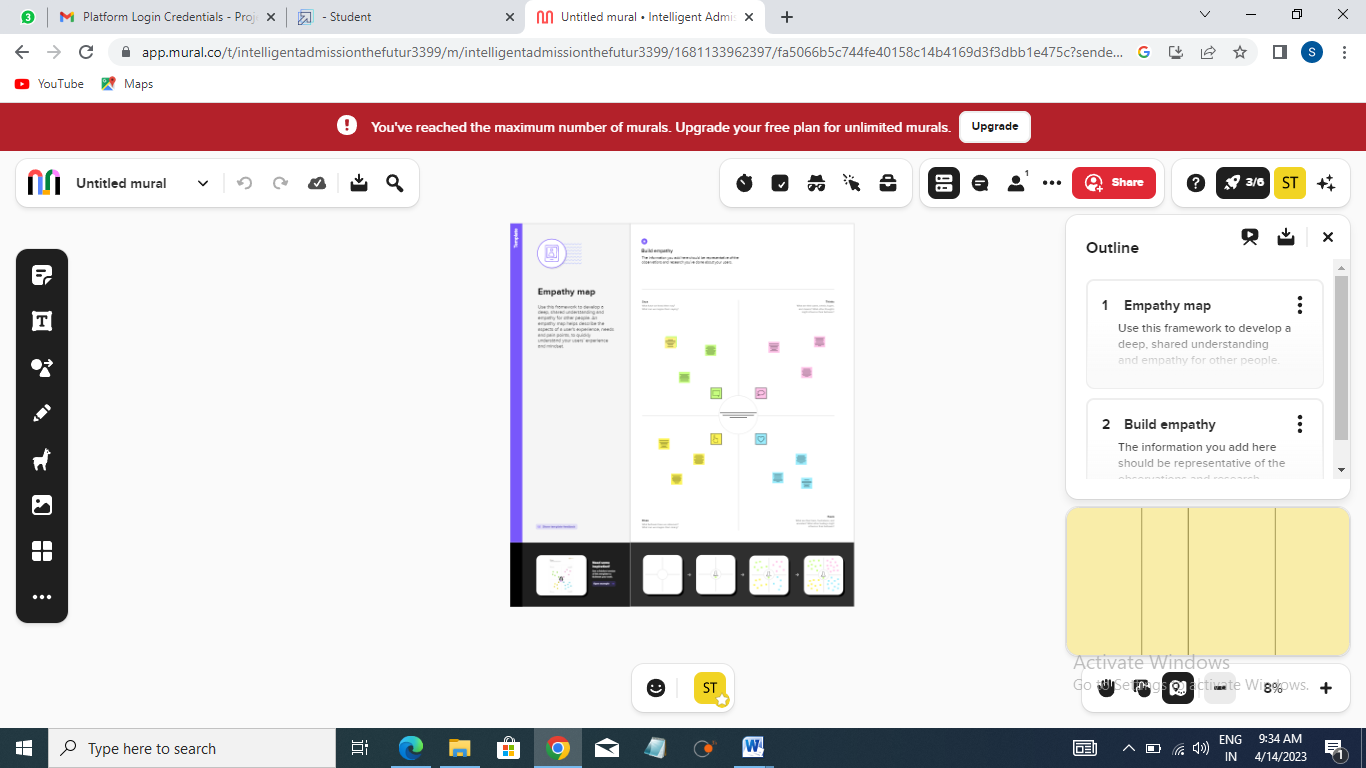
The ability to accurately predict the chances of university admission can help students make more informed decisions about which universities to apply to, increasing their chances of being admitted and ultimately gaining access to higher education.

**Business Model/Impact:-**

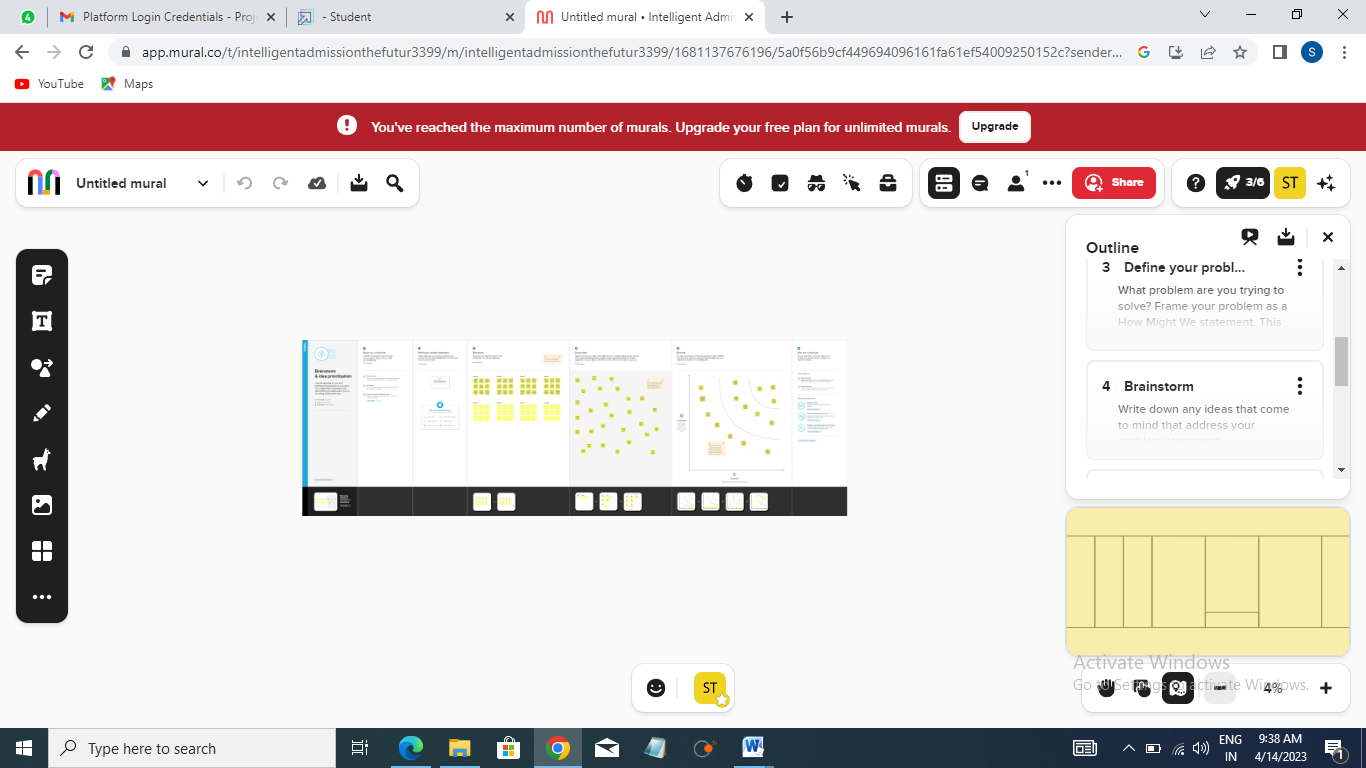
1. using machine learning models to predict university admission, the service can help universities more efficiently process and evaluate applications, potentially increasing the number of successful admissions.

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.

**Emphathy Map**



**Brain Strom**



Result

Define problem/problem understanding

**Mile stone 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.

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.

### 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.

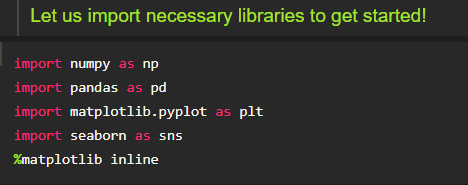
Link : <https://www.kaggle.com/rishal005/admission-predict>

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

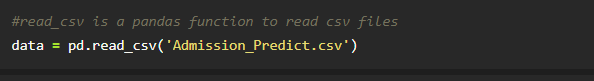
Import the necessary libraries as shown in the image. (optional) Here we have used visualisation style as fivethirtyeight.



### 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.



**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.

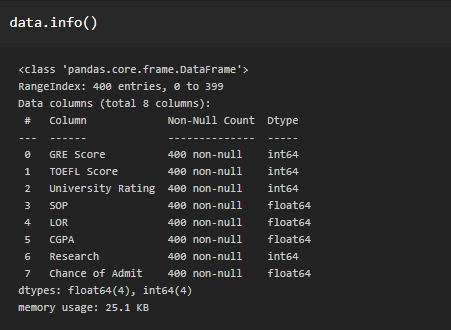
* Handling missing values
* Handling categorical data
* Handling Imbalance Data

Note:

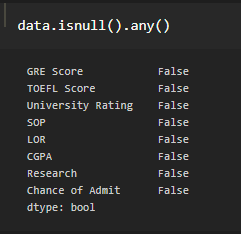
These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps

### Handling Missing Values

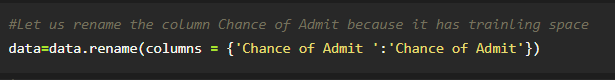
Let’s find the shape of our dataset first. To find the shape of our data, the df.shape() method is used. To find the data type, df.info() function is used.



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.



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



**Mile stone 3**

Exploratory Data Analysis

In this milestone, we will see exploratory data analysis

## Exploratory data analysis (EDA) is a (mainly) visual approach and philosophy that focuses on the initial ways by which one should explore a data set or experiment. Two main aspects of EDA are:

1. **Openness**, meaning a person exploring the data should be open to all possibilities prior to its exploration.
2. **Skepticism**, meaning one must ensure that the obvious story the data tells is not misleading.

There is no formal set of techniques that are used in EDA. Remember, EDA is an approach to how we analyze data, not a specific set of methods set in stone. It's a philosophy and art more so than a science.

Its purpose is to take a general view of some given data without making any assumptions about it. We are trying to get a feel for the data and what it might mean, as opposed to reject or accept some sort of premise around it, before we begin its exploration.

In other words, with EDA we let the data speak for itself instead of trying to force the data into some sort of predetermined model.

Nevertheless, some techniques are used to help us get a feel for the data. For instance, we can categorize data, quantify some of its basic aspects, or visualize it.

For instance, raw data can be plotted using histograms or other visualization techniques. Sometimes, the data is juxtaposed in a manner that helps us spot important patterns within or between data sets.

Overall, EDA can help us:

* Catch mistakes
* Gain new insights
* Detect outliers
* Test assumptions
* Identify important factors in the data
* Understand relationships

And perhaps, most importantly, EDA is used to help figure out our next steps with respect to the data. For instance, we might have new questions we need answered or new research we need to conduct.

### 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.

https://lh3.googleusercontent.com/LgjJ0J7dIqonJPoWp664CRM5b3WVux6yzKkq_uHnbDAeu3TvotaI9v6GYPN_HaMfCSbMqyfEfaA-lemAIkdbMJc9COy5c3JNhQyIw7pgWoR_z5WtKKN_5CK40wpw6AJIhwQ_D9de8yVivMKMYtfCdquUJ4Oh6zUdfDbZNmpWuG5T7byAykA1HH1w7tbNT_vVNmzz9dGb-g



### 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.

## ****The essence of visual analysis****

Visual analysis increases your understanding of how visual material **communicates** and **functions**, whether it generates **meaning**, elicits **emotion** or creates a **mood**. Visual analysis can be applied to any visual material including art, design and architecture.

Visual analysis identifies and explores characteristics of example material and relationships within the context in which they were produced and encountered. The purpose of visual analysis is to **make an argument** based on **visual evidence**. There are three parts to writing a visual analysis:

1. identify, describe and analyse the **visual** material
2. situate the visual material in its **context**
3. interpret and respond to the **content** of the visual material

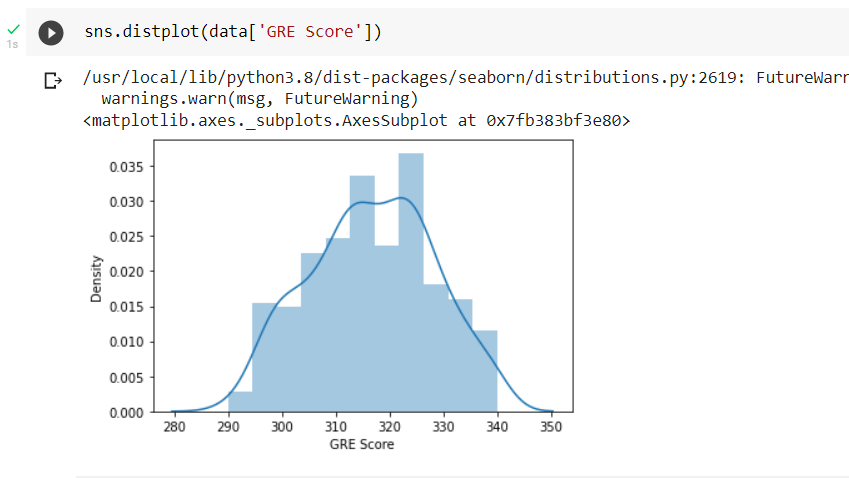
[**Identify, describe and analyse the visual material**](https://www.monash.edu/learnhq/excel-at-writing/annotated-assessment-samples/art-design-and-architecture/mada-visual-analysis#visual-material)

1. Begin by **stating** the type of material (for example, a building, a photograph, etc.), who made it, its title/name, and the year it was created. If relevant, also state its media, materials, components, dimensions, and location.
2. Then, **examine** and **describe** formal elements such as colour, line, shape, texture, and tone.
3. Next, **analyse** the composition, for example, relationships between elements like balance, geometry, pattern, proportion, repetition, rhythm, scale, and symmetrical and asymmetrical balance.
4. Begin by **stating** the type of material (for example, a building, a photograph, etc.), who made it, its title/name, and the year it was created. If relevant, also state its media, materials, components, dimensions, and location.
5. Then, **examine** and **describe** formal elements such as colour, line, shape, texture, and tone.
6. Next, **analyse** the composition, for example, relationships between elements like balance, geometry, pattern, proportion, repetition, rhythm, scale, and symmetrical and asymmetrical balance.
7. You can also **consider** ways the artist conveys feelings through form and space, for example, dynamism, harmony and tension, illusion, light and shade, modeling, perspective, and positive and negative space.

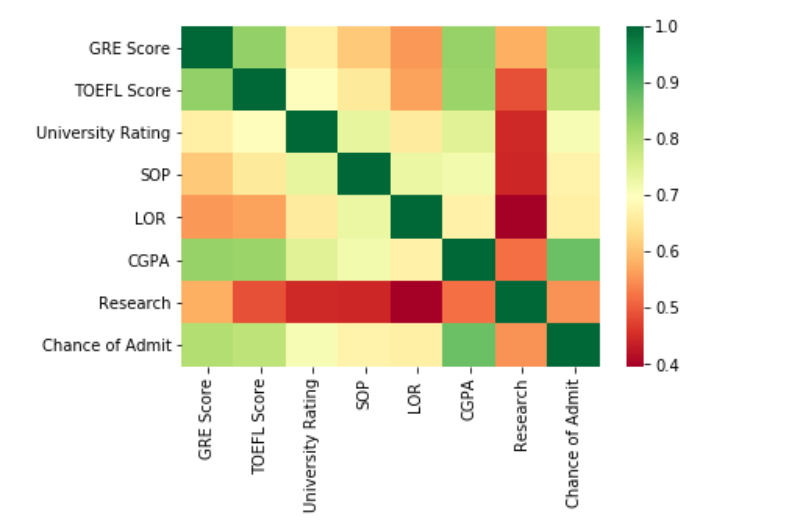
**Univariate Analysis**

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.



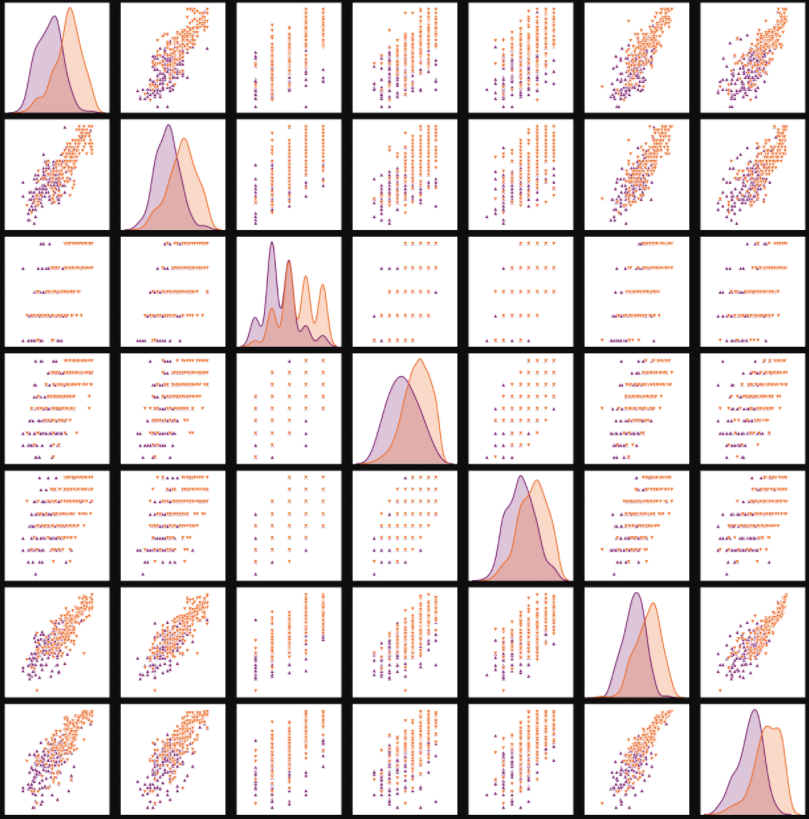
### Bivariate Analysis



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

https://lh4.googleusercontent.com/Bq8f-YTqlLS1fZl1ERD89gGfbEHbLetJpGTNbc2zPAi0VmkU8kCs4QZeLyIjIrctY_jwBhq2lZlapblxdL_RwPsYRX_PyCjjvc_HIooj-cI5cfcPE-_Z7WKrN8_20dRU3IxEVMWWCNHn6EW99QprxD3ARvxXVmWaejScjtT_nutzB5NVx4K_pNKs04dUvV5aFAgj3TLPmA



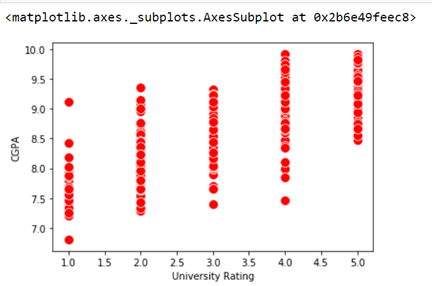
Pair plot usually gives pair wise relationships of the columns in the dataset

1.GRE score TOEFL score and CGPA all are linearly related to each other

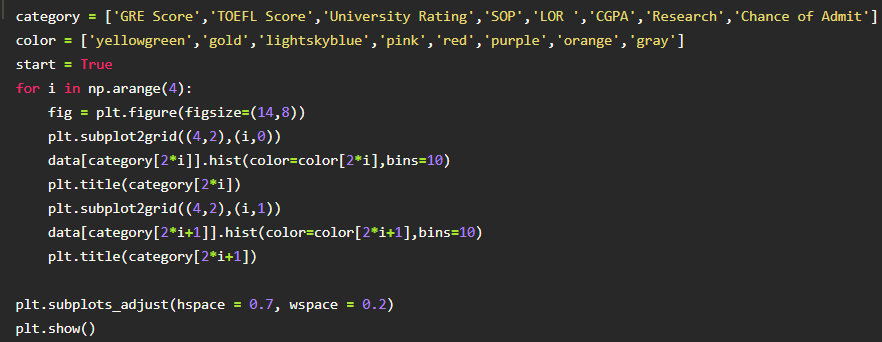
2. Students in research score high in TOEFL and GRE compared to non research candidates

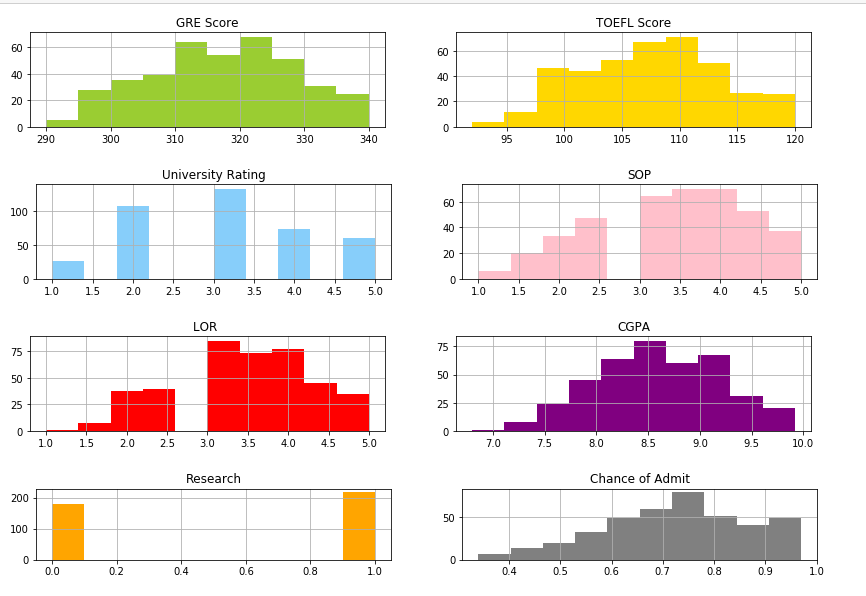
**Scatter Plot**: Matplot has a built-in function to create scatterplots called scatter(). A scatter plot is a type of plot that shows the data as a collection of points

https://lh5.googleusercontent.com/bf_sDnILd3us9zcZtifUnfzsVnCioa9Fi1ynNNj9E5wHKWSJPJJIiOplKuiS1K6FJuwTDaAq-3PVFm1z1TgJvwI2ca3nvNtxkkCC2VVNwaFlpD7rNHhkOirpQh9cHO9CkfdoFUkTMtUISay9W_BJCPlbr5KmiYetMAJ_oi4fdGNIBAOegm28mBopBydpevHenkKDjRodpw



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

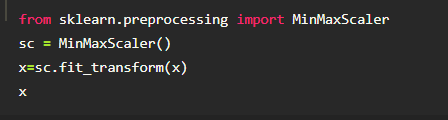




**Scaling the Data**

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.

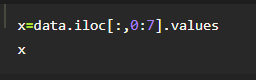


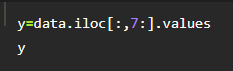
We will perform scaling only on the input values.

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**

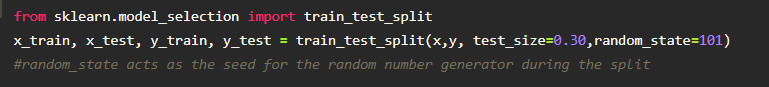
Now let’s split the Dataset into x and y





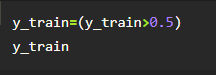
Changes: first split the dataset into x and y and then split the data set

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.



**Let us convert it into classification problem**

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



https://lh5.googleusercontent.com/eAnl6HBvgLfdxI8zXDHBz5NWDzUBomq_Px2vbuiivCpoVslxZcmzc7jUJkanU90vQi5ga-S1ZbJ1_coPdlc9kFCnAtaqSHkv4MAwXPnTThpSsCK_NffdmVv6d_iDHPO1xnul6IFRCcKbCxnZQpOM44MOCYGgwBYeyAHBVwwQZF1YpPDZUP5xIbZ9DyAT545N1I15rhQPMA

**Mile stone 4**

Model Building

In this Milestone, We will see Model Building

In regression analysis, model building is the process of developing a probabilistic model that best describes the relationship between the dependent and independent variables.

The major issues are finding the proper form (linear or curvilinear) of the relationship and selecting which independent variables to include.

In building models it is often desirable to use qualitative as well as quantitative variables.

As noted above, quantitative variables measure how much or how many  [qualitative variables](https://www.britannica.com/topic/qualitative-variable) represent types or categories.

For instance, suppose it is of interest to predict sales of an iced tea that is available in either bottles or cans.

Clearly, the independent variable “container type” could influence the dependent variable “sales.”

Container type is a qualitative variable, however, and must be assigned numerical [values](https://www.britannica.com/dictionary/values) if it is to be used in a regression study.

So-called dummy variables are used to represent qualitative variables in regression analysis.

For example, the dummy variable x could be used to represent container type by setting x = 0 if the iced tea is packaged in a bottle and x = 1 if the iced tea is in a can.

If the beverage could be placed in glass bottles, plastic bottles, or cans, it would require two dummy variables to properly represent the qualitative variable container type.

In general, k - 1 dummy variables are needed to model the effect of a qualitative variable that may assume k values.

### 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.

## ****Training Models using**** ****Model Parallelism****

So far we have trained each neural network on a single device. What if we want to train a single neural network across multiple devices? This requires chopping the model into separate chunks and running each chunk on a different device. Unfortunately, such model parallelism turns out to be pretty tricky, and it depends on the architecture of your neural network.

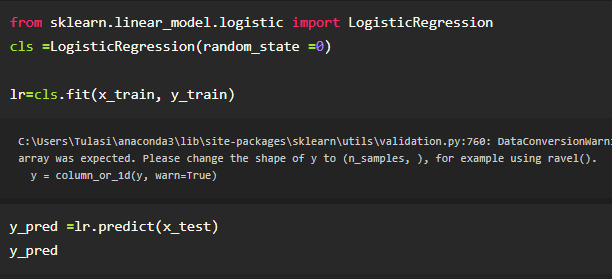
For fully connected networks, there is generally not much to be gained from this approach. Intuitively, it may seem that an easy way to split the model is to place each layer on a different device, but this does not work because each layer needs to wait for the output of the previous layer before it can do anything.

So perhaps you can slice it vertically for example, with the left half of each layer on one device, and the right part on another device? This is slightly better since both halves of each layer can indeed work in parallel, but the problem is that each half of the next layer requires the output of both halves, so there will be a lot of cross-device communication. This is likely to completely cancel out the benefit of the parallel computation since cross-device communication is slow.

### 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.



### ANN Model

Building and training an Artificial Neural Network (ANN) using the Keras library with TensorFlow as the backend.

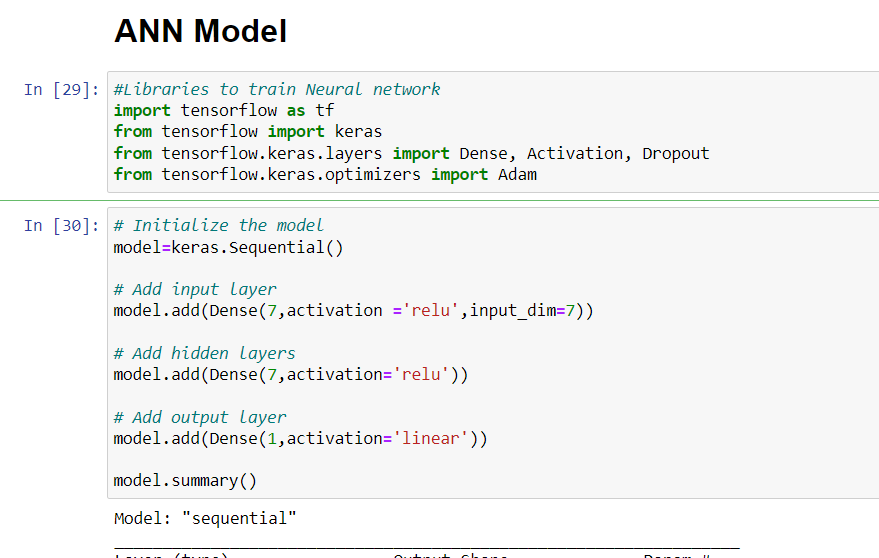
The ANN is initialised as an instance of the Sequential class, which is a linear stack of layers.

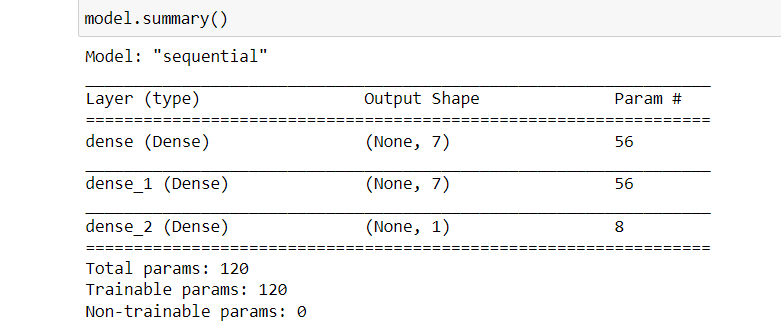
Then, the input layer and two hidden layers are added to the model using the Dense class, where the number of units and activation function are specified.

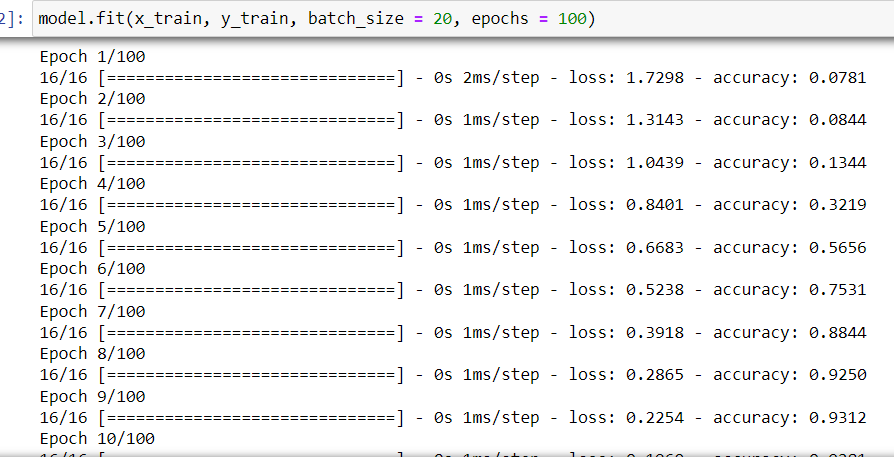
The output layer is also added using the Dense class with a sigmoid activation function.

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.

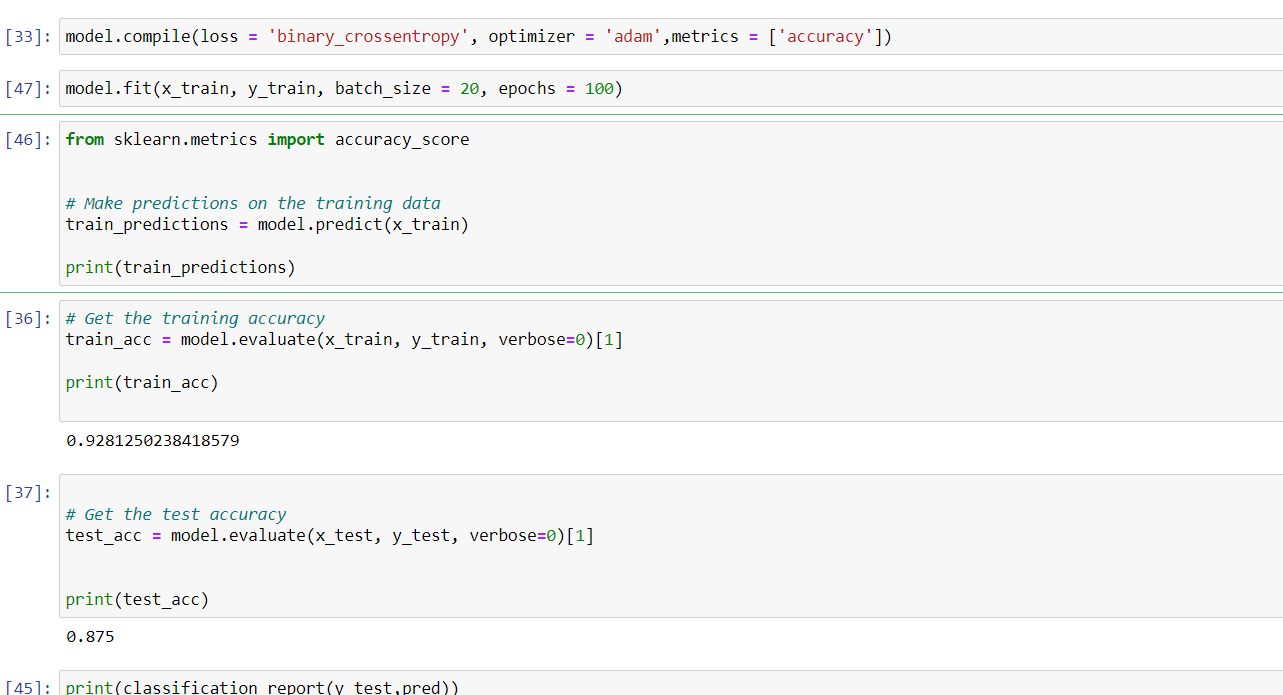


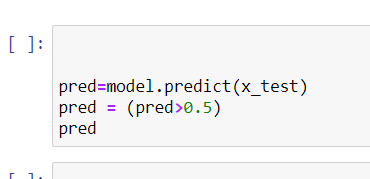


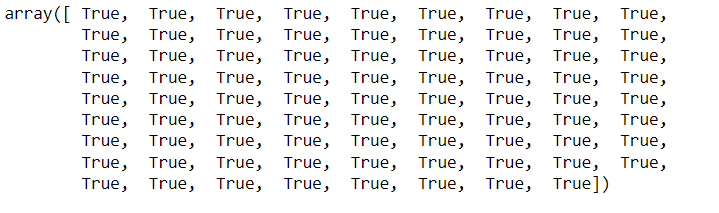


### Testing The Model

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

****

****

****

**Mile stone 5**

**Performance Testing & Hyper parameter Tuning**

In this milestone, We will see the performance testing and hyper parameter Tuning.

**Performance Testing**

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), hyper parameter optimizationor tuning is the problem of choosing a set of optimal [hyper parameters](https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning)) for a learning algorithm. A hyper parameter is a [parameter](https://en.wikipedia.org/wiki/Parameter) whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned.

The same kind of machine learning model can require different constraints, weights or learning rates to generalize different data patterns. These measures are called hyper parameters, and have to be tuned so that the model can optimally solve the machine learning problem. Hyper parameter optimization finds a tuple of hyper parameters that yields an optimal model which minimizes a predefined [loss function](https://en.wikipedia.org/wiki/Loss_function) on given independent data. The objective function takes a tuple of hyperparameters and returns the associated loss. [Cross-validation](https://en.wikipedia.org/wiki/Cross-validation_(statistics)) is often used to estimate this generalization performance.

There are 5 main types of performance testing.

1. Capacity Testing
2. [Load Testing](https://theqalead.com/tools/load-testing-tools/)
3. Volume Testing
4. Stress Testing
5. Soak Testing

### ****Capacity Testing:****

### Tests how many users the system can handle before performance dips below acceptable levels. By testing a software’s capacity it helps developers anticipate issues in terms of scalability and future user-base growth.

### ****Load Testing:****

### Confirms that the system can handle the required number of users and still operate at a high level of performance. This ensures that there is no day to day issues in performance.  ****Volume Testing:****

### Checks that the software can handle and process a large amount of data at once without breaking, slowing down, or losing any information.

### ****Stress Testing:****

### Intentionally tries to break the software by simulating a number of users that greatly exceeds expectations. The launch day of a new iPhone and the sudden spike in user traffic on the Apple website is a good example of a [stress test in the real world](https://www.businessinsider.com/apple-store-crashes-as-iphone-6-goes-on-sale-2014-9?r=US&IR=T).

### ****Soak Testing:****

### Simulates high traffic for an extended period of time. Checks the software’s ability to tolerate extended periods of high traffic.

### Hyper parameter Tuning

This page describes the concepts involved in hyper parameter tuning, which is the automated model enhancer provided by AI Platform Training. Hyper parameter tuning takes advantage of the processing infrastructure of Google Cloud to test different hyper parameter configurations when training your model. It can give you optimized values for hyper parameters, which maximizes your model's predictive accuracy.

Your training application handles three categories of data as it trains your model:

* Your input data (also called training data) is a collection of individual records (instances) containing the features important to your machine learning problem. This data is used during training to configure your model to accurately make predictions about new instances of similar data. However, the values in your input data never directly become part of your model.
* Your model's parameters are the variables that your chosen machine learning technique uses to adjust to your data. For example, a deep neural network (DNN) is composed of processing nodes (neurons), each with an operation performed on data as it travels through the network. When your DNN is trained, each node has a weight value that tells your model how much impact it has on the final prediction. Those weights are an example of your model's parameters. In many ways, your model's parameters are the model—they are what distinguishes your particular model from other models of the same type working on similar data.
* Your hyper parameters are the variables that govern the training process itself. For example, part of setting up a deep neural network is deciding how many hidden layers of nodes to use between the input layer and the output layer, and how many nodes each layer should use. These variables are not directly related to the training data. They are configuration variables. Note that parameters change during a training job, while hyper parameters are usually constant during a job.

### 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.

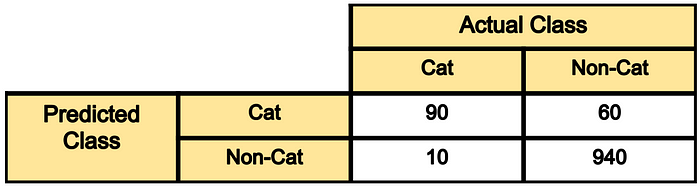
We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

## 1- Confusion Matrix (not a metric, but important to know!)

Let’s first make sure we know the basic terminologies used in classification problems before going through the detail of each metric. **You can skip this section if you are already familiar with the terminologies.**

One of the key concept in classification performance is **confusion matrix**(AKA error matrix), which is a tabular visualization of the model predictions versus the ground-truth labels. Each row of confusion matrix represents the instances in a predicted class and each column represents the instances in an actual class.

Let’s go through this with an example. Let’s assume we are building a binary classification to classify cat images from non-cat images. And let’s assume our test set has 1100 images (1000 non-cat images, and 100 cat images), with the below confusion matrix.



* **Out of 100 cat images** the model has predicted 90 of them correctly and has mis-classified 10 of them. If we refer to the “cat” class as positive and the non-cat class as negative class, then 90 samples predicted as cat are considered as as **true-positive**, and the 10 samples predicted as non-cat are **false negative**.
* **Out of 1000 non-cat images**, the model has classified 940 of them correctly, and misclassified 60 of them. The 940 correctly classified samples are referred as **true-negative**, and those 60 are referred as **false-positive**.

As we can see diagonal elements of this matrix denote the correct prediction for different classes, while the off-diagonal elements denote the samples which are misclassified.

Now that we have a better understanding of the confusion matrix, let’s get into the actual metrics.

**2- Classification Accuracy**

Classification accuracy is perhaps the simplest metrics one can imagine, and is defined as the**number of correct predictions divided by the total number of predictions,**multiplied by 100**.**So in the above example, out of 1100 samples 1030 are predicted correctly, resulting in a classification accuracy of:

**Classification accuracy**= (90+940)/(1000+100)= 1030/1100= 93.6%

## 3- Recall

Recall is another important metric, which is defined as the fraction of samples from a class which are correctly predicted by the model. More formally:

**Recall= True\_Positive/ (True\_Positive+ False\_Negative)**

Therefore, for our example above, the recall rate of cat and non-cat classes can be found as:

**Recall\_cat= 90/100= 90%**

**Recall\_NonCat= 940/1000= 94%**

## 5- F1 Score

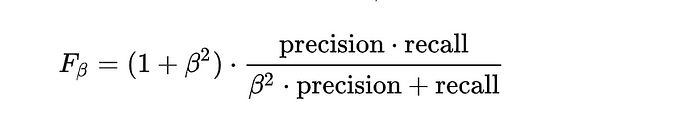
Depending on application, you may want to give higher priority to recall or precision. But there are many applications in which both recall and precision are important. Therefore, it is natural to think of a way to combine these two into a single metric. **One popular metric which combines precision and recall is called F1-score**, which is the harmonic mean of precision and recall defined as:

**F1-score= 2\*Precision\*Recall/(Precision+Recall)**

So for our classification example with the confusion matrix in Figure 1, the F1-score can be calculated as:

**F1\_cat= 2\*0.6\*0.9/(0.6+0.9)= 72%**

The generalized version of F-score is defined as below. As we can see F1-score is special case of F\_ℬ when ℬ= 1.

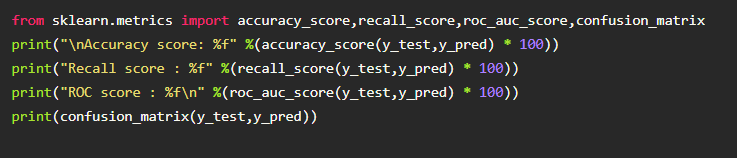


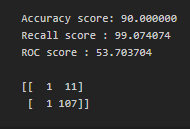
It is good to mention that there is always a trade-off between precision and recall of a model, if you want to make the precision too high, you would end up seeing a drop in the recall rate, and vice versa.

### Compare The Model

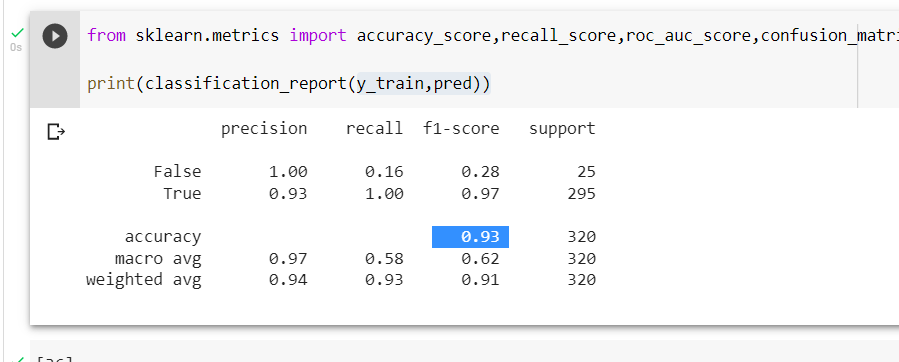
For comparing the above four models, the compareModel function is defined.

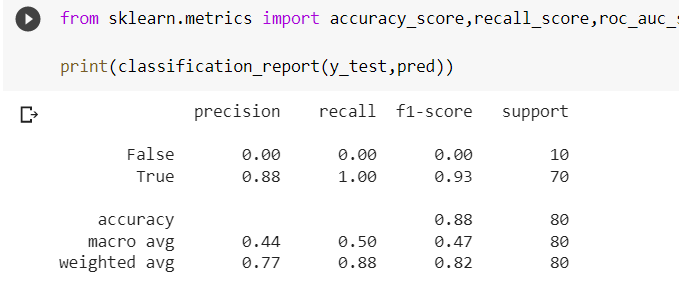
**Logistics Regression model**

****

****

**ANN Model** : Training Accuracy





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 93.% with training data , 88% accuracy for the testing data.

**Mile stone 6**

### Model Deployment

In this Milestone, We will see the model deployment.

Model deployment is the process of putting machine learning models into production.

This makes the model’s predictions available to users, developers or systems, so they can make business decisions based on data, interact with their application (like recognize a face in an image) and so on.

Model deployment is considered to be a challenging stage for data scientists.

This is because it is often not considered their core responsibility, and due to the technological and mindset differences between model development and training and the organizational tech stack, like versioning, testing and scaling which make deployment difficult.

These organizational and technological silos can be overcome with the right model deployment frameworks, tools and processes.

## Challenges of Machine Learning Model Deployment

There are a number of reasons model deployment is a resource-intensive and challenging process:

* **Silos:** Data scientists are focussed on training and optimizing models. In many organizations, data science is siloed from the rest of the machine learning lifecycle. The infamous ‘throw it over the wall’ approach is notorious for creating bottlenecks, duplicate work, and general chaos. The production system, which is managed and maintained by Develops and IT, is usually unfamiliar with the ML frameworks and files the models are based on, written and produced by the data scientists. In some cases, the models need to be re-coded, which can take weeks and is a very tedious process.
* **Preparing for the live environment:** In the lab, models are developed and trained based on existing data. However, in production, models will have to work with (sometimes real-time) real data from external sources. Usually the data needs to be processed before inferring through the model. Then, the output and predictions need to be properly consumed by the applications. These processes need to be prepared and orchestrated to ensure a smooth and successful deployment.
* **Monitoring models in production:** Deployment doesn’t end once the model is in production. Today, models are constantly changing due to changing business needs, data that changes or even use cases of the model that changes. To ensure their relevance and business value in performance, models in production need to be evaluated and flagged when a model isn’t performing, so they can be retrained and deployed again.
* **Infrastructure management:** A small amount of data doesn’t require 8 GPUs for inferring, but traffic at rush hour does require many computational resources to ensure high performance. Managing this requires the right processes and tools.

## Automating Machine Learning Model Deployment

Automating the deployment of models helps reduce friction and improve scalability and repeatability. By using [CI/CD](https://www.iguazio.com/glossary/ci-cd-for-machine-learning/) tools and integrating them into the MLOps pipeline, data scientists can continuously train their models and retrain if drift is detected.

When automating the model deployment pipeline, it is important to monitor that retraining is conducted correctly and that the outputs make sense. If the metrics show anomalies, the retrained model should probably not be deployed. So, automate with care: add alerts and triggers to your automation to ensure an accurate model is deployed.

### 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.



A note about saving models: models saved in .hdf5 format are great because the whole model is one place and can be loaded somewhere else, such as in deployment. However the files can get large, and saving your model at every epoch can get storage intensive fast. One option available in the ModelCheckpoint callback constructor is save\_weights\_only=True. This will save space, but will not save the entire model architecture. In order to recover it, you would rebuild the model and then assign the saved weights, rather than just loading it all in one step.

Another quirk I’ve found is that not every layer likes to be saved in .hdf5 format. In a previous blog post I showed you how to add a Text Vectorization layer to a NLP model to do preprocessing as part of the model itself. However, I found that that I couldn’t save my model in .hdf5 format if I used that layer. However, if I leave off the .hdf5 extension, then keras saves the model as a file directory of assets, and this works for the Text Vectorization layer.

After fitting, we can reload our model for evaluation at its best performing epoch with:

model = keras.models.load\_model(filepath)

**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

**Building Html Pages**

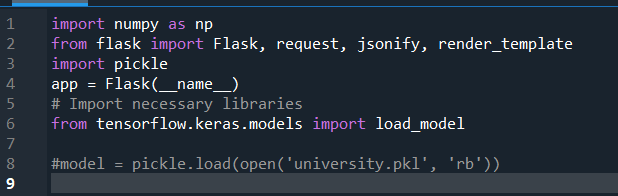
For this project create two HTML files namely

* home.html
* predict.html

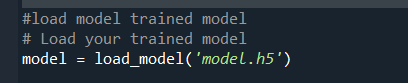
and save them in the templates folder.

### Build Python Code

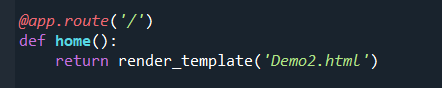
Import the libraries

****

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.



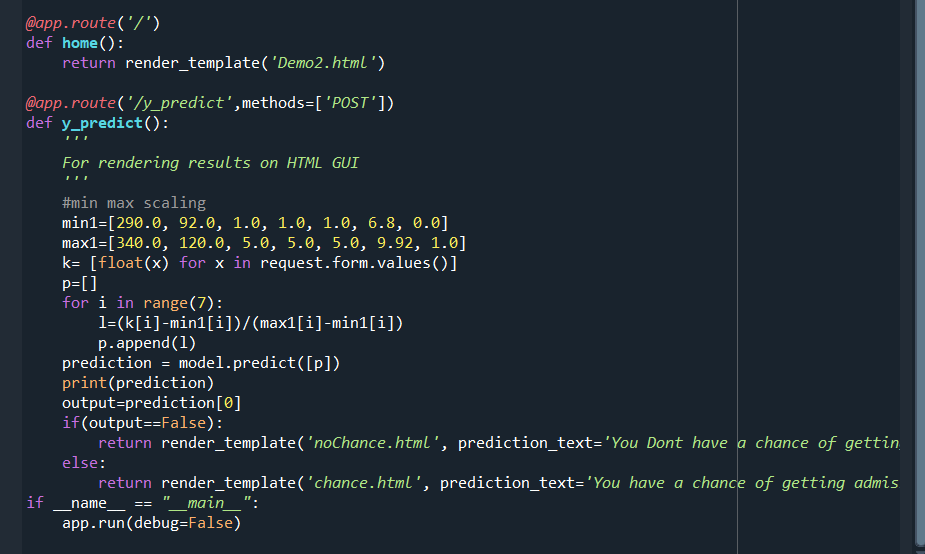
Render HTML page:



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

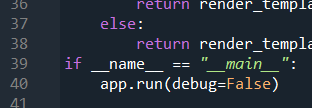
In the above example, ‘/’ URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:



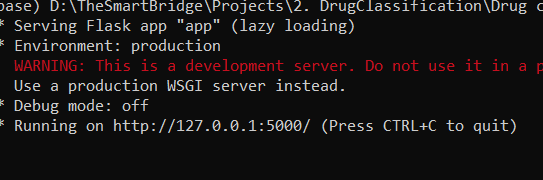
Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text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

Main Function:

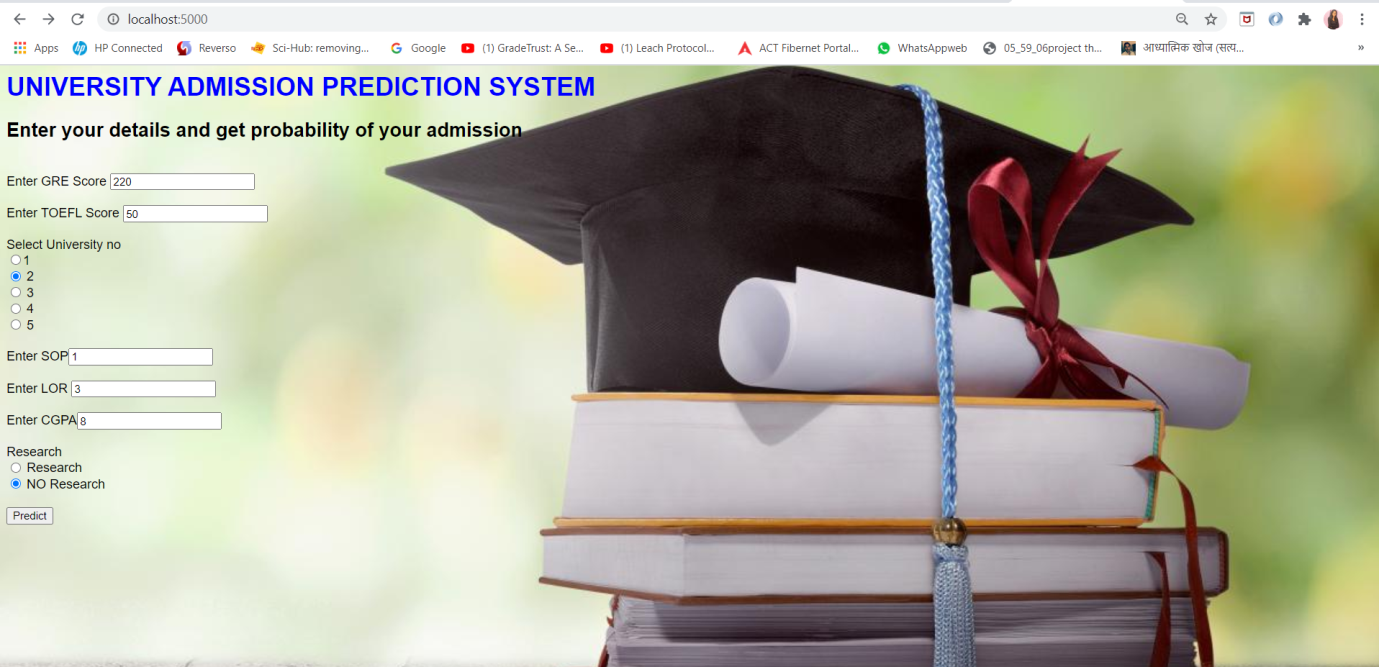


**Run The Web Application**

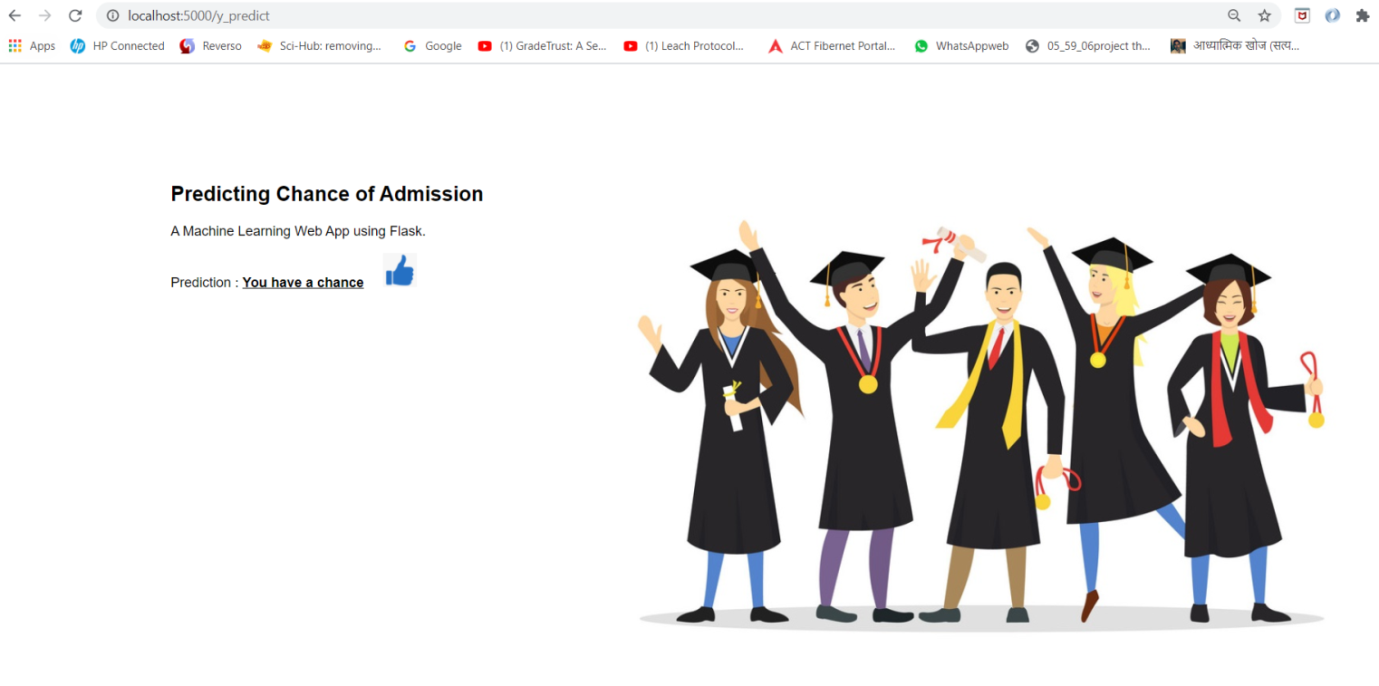
* Open anaconda prompt from the start menu
* Navigate to the folder where your python script is.
* Now type “python app.py” command
* Navigate to the localhost where you can view your web page.
* Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.



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

****

Now, when you click on click me to predict the button from the banner you will get redirected to the prediction page.

****

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

### Source code:

### # importing required lib

### import numpy as np

### import pandas as pd

### import seaborn as sns

### import matplotlib.pyplot as plt

### import warnings

### warnings.filterwarnings('ignore')

### #.Checking for available styles

### plt.style.available

### ['Solarize\_Light2',

### '\_classic\_test\_patch',

### '\_mpl-gallery',

### '\_mpl-gallery-nogrid',

### 'bmh',

### 'classic',

### 'dark\_background',

### 'fast',

### 'fivethirtyeight',

### 'ggplot',

### 'grayscale',

### 'seaborn-v0\_8',

### 'seaborn-v0\_8-bright',

### 'seaborn-v0\_8-colorblind',

### 'seaborn-v0\_8-dark',

### 'seaborn-v0\_8-dark-palette',

### 'seaborn-v0\_8-darkgrid',

### 'seaborn-v0\_8-deep',

### 'seaborn-v0\_8-muted',

### 'seaborn-v0\_8-notebook',

### 'seaborn-v0\_8-paper',

### 'seaborn-v0\_8-pastel',

### 'seaborn-v0\_8-poster',

### 'seaborn-v0\_8-talk',

### 'seaborn-v0\_8-ticks',

### 'seaborn-v0\_8-white',

### 'seaborn-v0\_8-whitegrid',

### 'tableau-colorblind10']

### # Applying styles to notebook

### plt.style.use('fivethirtyeight')

### # Reading csv data

### df=pd.read\_csv('/content/Admission\_Predict.csv')

### df.head()

### # checking data type

### df.info()

### """

### Type of Analysis

### 1)Univariate analysis

### 2)Bivariate analysis

### 3)Multivariate analysis

### 4)Descriptive analysis/statistics

### """

### #Univariate analysis - Extracting info from a single column

### #Checking data distribution

### plt.subplot(121)

### sns.distplot(df['SOP'])

### # Visualizing counts in each variable

### for i,j in enumerate('df\_cat'):

### plt.subplot(1,14,i+1)

### sns.countplot()

### 

### df['SOP'].max()

### # Creating new column

### df['SOP\_']=['1.0-2.0' if x<=2.0 else "2.0-4.0" if x>2.0 and x<=4.0 else '4.0+' for x in df['SOP']]

### df.head()

### #inding relation between abord and date

### sns.heatmap(pd.crosstab(df['SOP'],df['TOEFL Score']))

### pd.crosstab(df['SOP'],df['TOEFL Score'])

### y=df['SOP']

### # importing required lib

### import numpy as np

### import pandas as pd

### import seaborn as sns

### import matplotlib.pyplot as plt

### import warnings

### warnings.filterwarnings('ignore')

### # Reading csv data

### df = pd.read\_csv('/content/Admission\_Predict.csv')

### df.head()

### # Descriptive analysis-descriptive stat

### df.describe()

### #Libraries to train Neural network

### import tensorflow as tf

### from tensorflow import keras

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

### from tensorflow.keras.optimizers import Adam

### # Iitialize the model

### model=keras.Sequential()

### # Add input layer

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

### # Add hidden layers

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

### # Add output Layer

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

### model.summary()

### Model: "sequential"

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