

# Intelligent Admission: The Future Of University Decision Making With Machine Learning

## Introduction

### 1.1 OVERVIEW:

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.

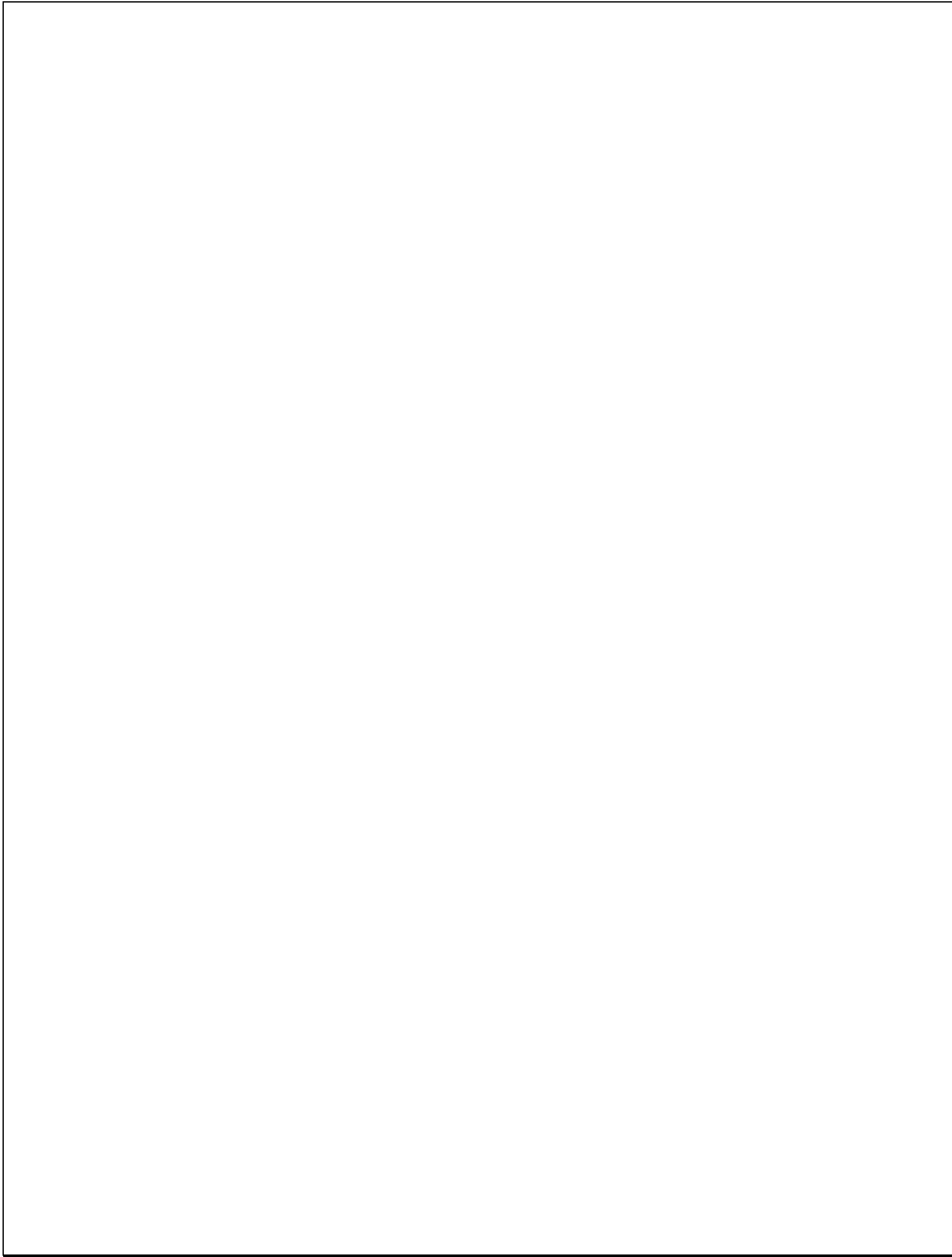
## 1.2 PURPOSE:

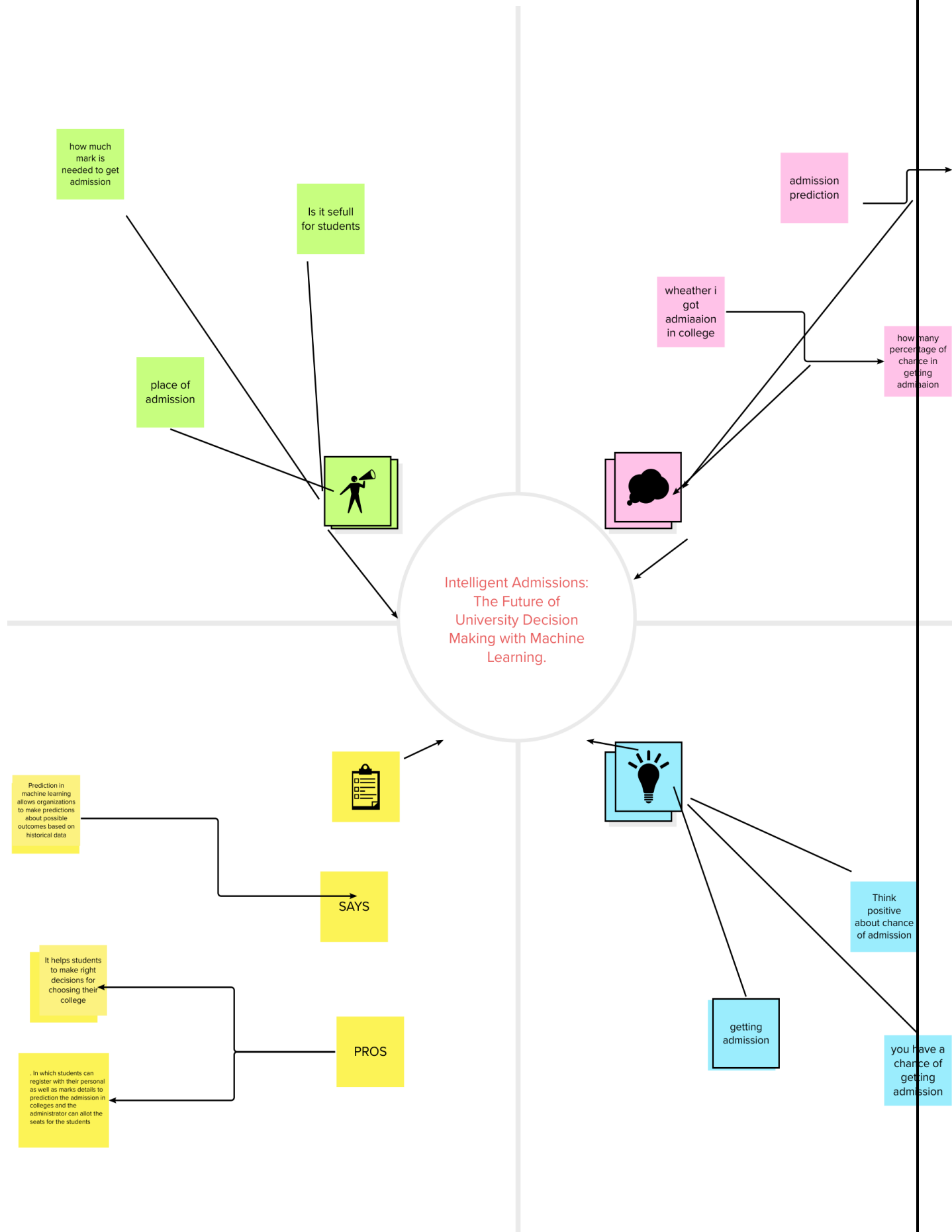
**It helps students to make right decisions for choosing their college.** In which students can register with their personal as well as marks details to prediction the admission in colleges and the administrator can allot the seats for the students

Prediction in machine learning **allows organizations to make predictions about possible outcomes based on historical data.** These assumptions allow the organization to make decisions resulting in tangible

business results. Predictive analytics can be used to anticipate when users will churn or leave an organization.


## 1.3 EMPATHY MAP





# BrainStorming And Idea Prioritization

Step 1



## Brainstorm & idea prioritization

Use this template to generate ideas and prioritize them. It's a great way to get your team's input and to make sure you're focusing on the most important ideas.

- 1. Brainstorm ideas
- 2. Prioritize ideas
- 3. Implement ideas

© 2020 The McGraw-Hill Companies

Step 2

Define your problem or challenge

What problem or challenge are you trying to solve? What are the goals of your project? What are the constraints?

1. Define the problem

2. Define the goals

3. Define the constraints

4. Define the success criteria

Step 3

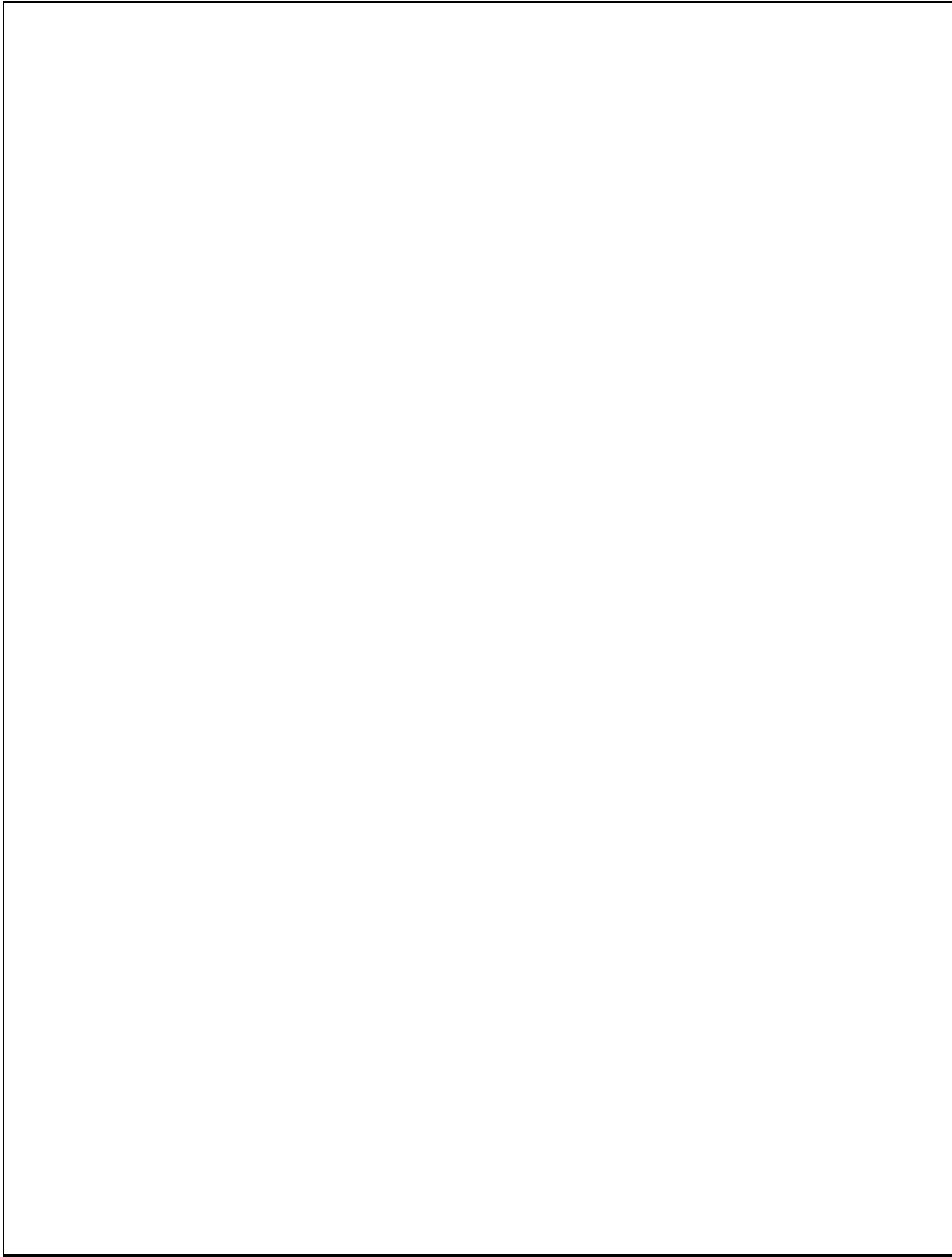
Brainstorm ideas

What ideas do you have to solve the problem? What are the goals of your project? What are the constraints?

1. Brainstorm ideas

2. Prioritize ideas

3. Implement ideas





2

## Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

Person 1

Use feedback and a new  
approach to solve the  
problem.

Use the same  
approach to solve the  
problem.


Person 2

Use feedback  
and a new  
approach to solve the  
problem.

Use the same  
approach to solve the  
problem.


Person 3

Use feedback and a new  
approach to solve the  
problem.

Use the same  
approach to solve the  
problem.


Person 5


Person 6


Person 7


TIP  
You  
and  
ske

Develop  
model that  
is count  
academic  
students  
such as  
and  
conclude

Pers

3

### Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups...

⌚ 20 minutes

Demographic factors:  
Consider variables  
such as age, gender,  
ethnicity, and  
socioeconomic status  
as possible predictors  
of university  
admission.

Academic achiev  
Look at factors s  
high school G  
standardized  
scores, and  
extracurricular ac  
as possible indica  
academic succo

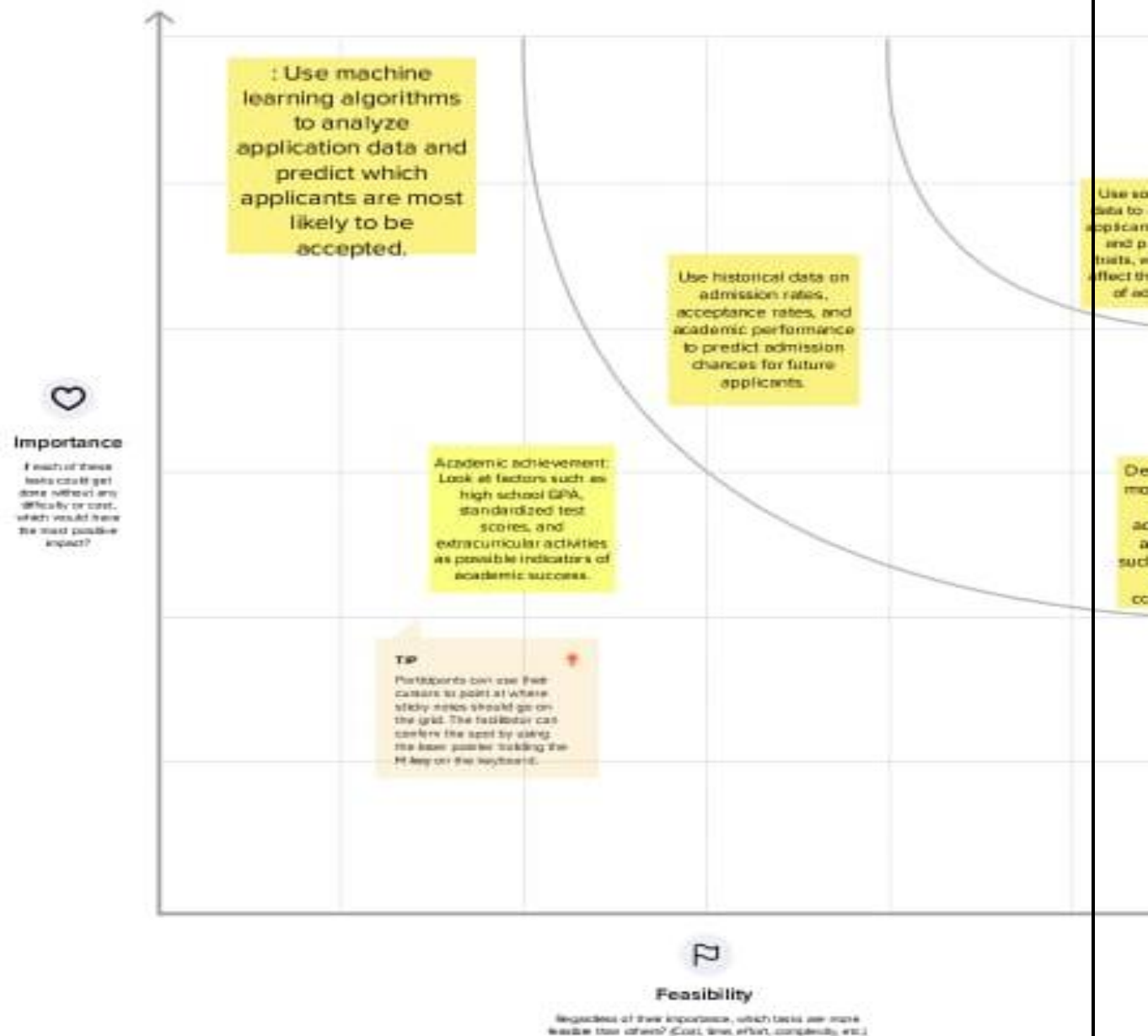
TIP  
Add c  
notes  
brown  
categ  
them

4

## Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

⌚ 20 minutes



## Importing the libraries

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

Let us import necessary libraries to get started!

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

## 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('Admission_Predict.csv')
```

## 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.

```
data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 8 columns):
#   Column              Non-Null Count  Dtype
---  -
0   GRE Score            400 non-null   int64
1   TOEFL Score          400 non-null   int64
2   University Rating    400 non-null   int64
3   SOP                  400 non-null   float64
4   LOR                  400 non-null   float64
5   CGPA                 400 non-null   float64
6   Research              400 non-null   int64
7   Chance of Admit      400 non-null   float64
dtypes: float64(4), int64(4)
memory usage: 25.1 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.

```
data.isnull().any()
```

```
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.

```
#Let us rename the column Chance of Admit because it has trainling space
data=data.rename(columns = {'Chance of Admit ':'Chance of Admit'})
```

---

## Exploratory Data Analysis

### 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()
```

	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
mean	316.807500	107.410000	3.087500	3.400000	3.452500	8.598925	0.547500	0.724350
std	11.473646	6.069514	1.143728	1.006869	0.898478	0.596317	0.498362	0.142609
min	290.000000	92.000000	1.000000	1.000000	1.000000	6.800000	0.000000	0.340000
25%	308.000000	103.000000	2.000000	2.500000	3.000000	8.170000	0.000000	0.640000
50%	317.000000	107.000000	3.000000	3.500000	3.500000	8.610000	1.000000	0.730000
75%	325.000000	112.000000	4.000000	4.000000	4.000000	9.062500	1.000000	0.830000
max	340.000000	120.000000	5.000000	5.000000	5.000000	9.920000	1.000000	0.970000

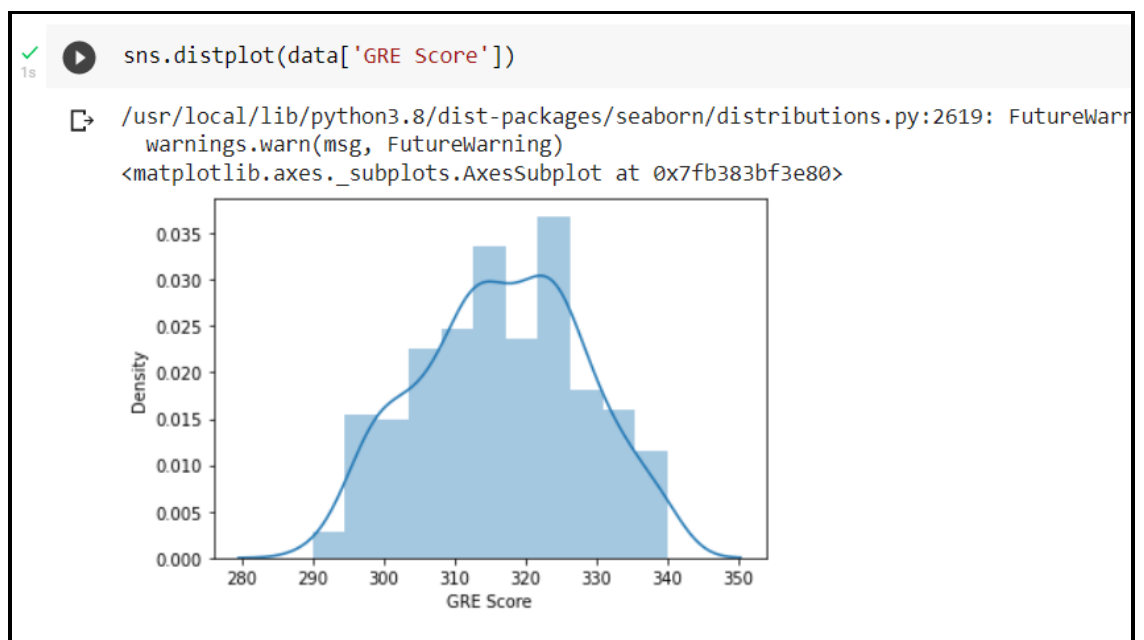
## Visual analysis

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.



### Bivariate analysis

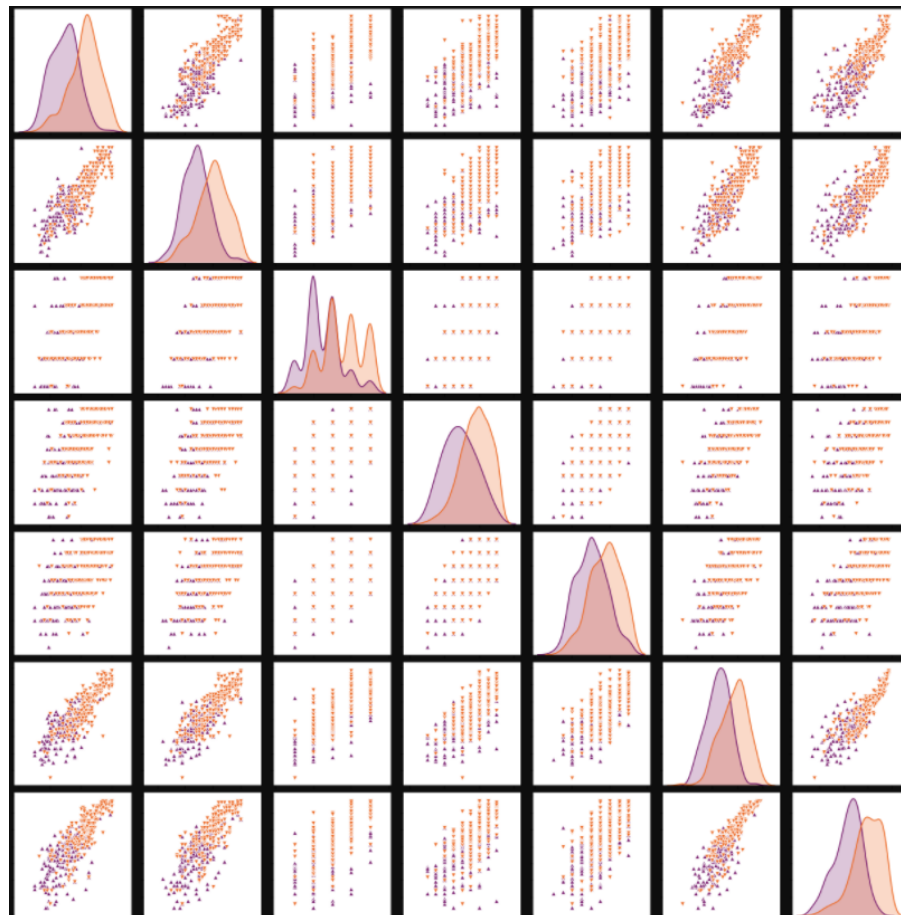


We see that the output variable "Chance of Admit" depends on CGPA, GRE, TOEFL. The columns SOP, LOR and Research have less impact on university admission

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

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





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

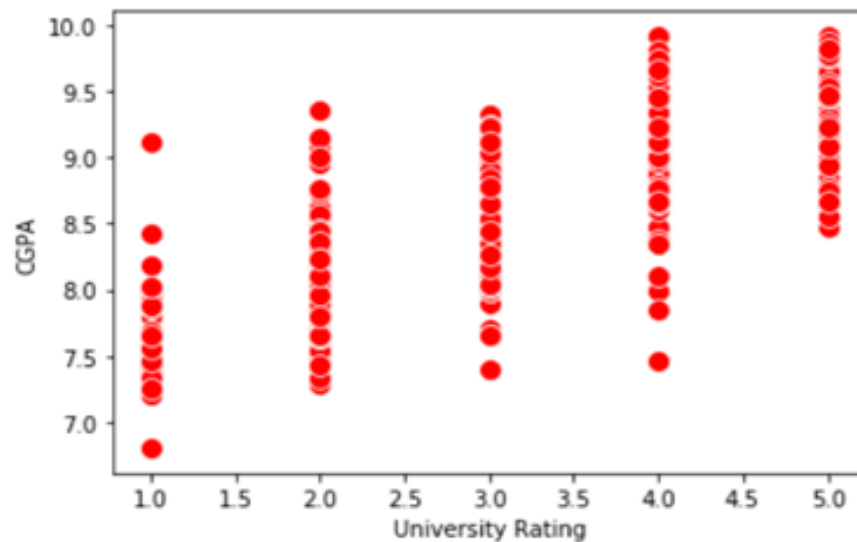
1. GRE score TOEFL score and CGPA all are linearly related to each other
2. Students in research score high in TOEFL and GRE compared to non research candidates

**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)
```

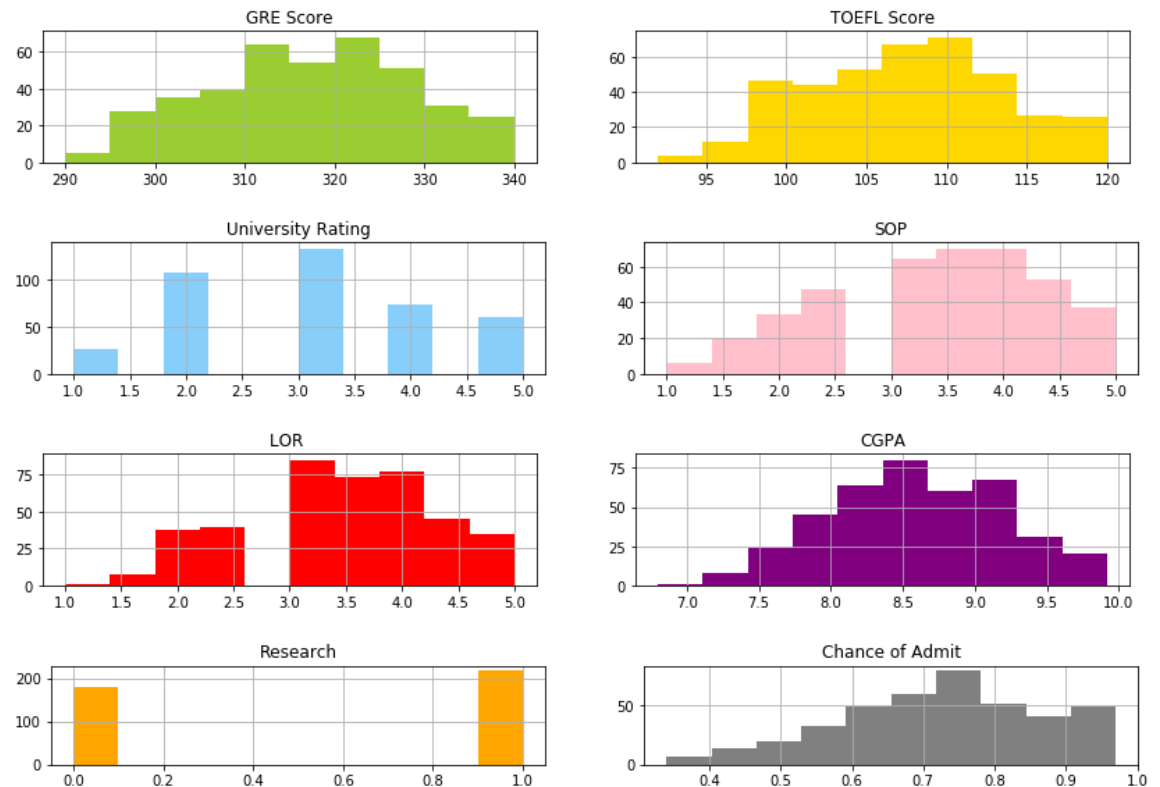
<matplotlib.axes.\_subplots.AxesSubplot at 0x2b6e49feec8>



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 of the important processes we have to perform on the dataset, because data measures in different ranges can lead to mislead in prediction.

Models such as KNN, Logistic regression need scaled data, as they follow distance-based methods and Gradient Descent concepts.

```
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
```

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

```
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)
```

## **Model Building**

### **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

```
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_id(y, warn=True)

y_pred =lr.predict(x_test)
y_pred
```

## 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.

## ANN Model

```
In [29]: #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
```

```
In [30]: # Initialize 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"

Layer (type)	Output Shape	Param #
--------------	--------------	---------

```
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 7)	56
dense_1 (Dense)	(None, 7)	56
dense_2 (Dense)	(None, 1)	8

Total params: 120  
Trainable params: 120  
Non-trainable params: 0

```
2]: model.fit(x_train, y_train, batch_size = 20, epochs = 100)
```

```
Epoch 1/100
16/16 [=====] - 0s 2ms/step - loss: 1.7298 - accuracy: 0.0781
Epoch 2/100
16/16 [=====] - 0s 1ms/step - loss: 1.3143 - accuracy: 0.0844
Epoch 3/100
16/16 [=====] - 0s 1ms/step - loss: 1.0439 - accuracy: 0.1344
Epoch 4/100
16/16 [=====] - 0s 1ms/step - loss: 0.8401 - accuracy: 0.3219
Epoch 5/100
16/16 [=====] - 0s 1ms/step - loss: 0.6683 - accuracy: 0.5656
Epoch 6/100
16/16 [=====] - 0s 1ms/step - loss: 0.5238 - accuracy: 0.7531
Epoch 7/100
16/16 [=====] - 0s 1ms/step - loss: 0.3918 - accuracy: 0.8844
Epoch 8/100
16/16 [=====] - 0s 1ms/step - loss: 0.2865 - accuracy: 0.9250
Epoch 9/100
16/16 [=====] - 0s 1ms/step - loss: 0.2254 - accuracy: 0.9312
Epoch 10/100
16/16 [=====] - 0s 1ms/step - loss: 0.1820 - accuracy: 0.9321
```

## Testing the model

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

```
[33]: model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

```
[47]: model.fit(x_train, y_train, batch_size = 20, epochs = 100)
```

```
[46]: from sklearn.metrics import accuracy_score
```

```
# Make predictions on the training data  
train_predictions = model.predict(x_train)  
  
print(train_predictions)
```

```
[36]: # Get the training accuracy  
train_acc = model.evaluate(x_train, y_train, verbose=0)[1]  
  
print(train_acc)
```

```
0.9281250238418579
```

```
[37]: # Get the test accuracy  
test_acc = model.evaluate(x_test, y_test, verbose=0)[1]  
  
print(test_acc)
```

```
0.875
```

```
[45]: print(classification_report(y_test, pred))
```

```
[ ]:  
  
pred=model.predict(x_test)  
pred = (pred>0.5)  
pred
```

```
array([[ True,  True,  True,  True,  True,  True,  True,  True,  True,  
        True,  True,  True,  True,  True,  True,  True,  True,  True,  
        True,  True,  True,  True,  True,  True,  True,  True,  True,  
        True,  True,  True,  True,  True,  True,  True,  True,  True,  
        True,  True,  True,  True,  True,  True,  True,  True,  True,  
        True,  True,  True,  True,  True,  True,  True,  True,  True,  
        True,  True,  True,  True,  True,  True,  True,  True,  True])
```

## Performance Testing & Hyperparameter Tuning

### 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.

## 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))
```

```
Accuracy score: 90.000000
Recall score : 99.074074
ROC score : 53.703704

[[ 1  11]
 [ 1 107]]
```

### ANN Model : Training Accuracy

```
from sklearn.metrics import accuracy_score, recall_score, roc_auc_score, confusion_matrix
print(classification_report(y_train, pred))
```

	precision	recall	f1-score	support
False	1.00	0.16	0.28	25
True	0.93	1.00	0.97	295
accuracy			0.93	320
macro avg	0.97	0.58	0.62	320
weighted avg	0.94	0.93	0.91	320



```

from sklearn.metrics import accuracy_score, recall_score, roc_auc_score
print(classification_report(y_test, pred))

```

	precision	recall	f1-score	support
False	0.00	0.00	0.00	10
True	0.88	1.00	0.93	70
accuracy			0.88	80
macro avg	0.44	0.50	0.47	80
weighted avg	0.77	0.88	0.82	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 93.% with training data , 88% accuracy for the testing data.

## Model Deployment

### 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] # Save the model in HDF5 format
      model.save('model.h5')

```

### 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

```
1 import numpy as np
2 from flask import Flask, request, jsonify, render_template
3 import pickle
4 app = Flask(__name__)
5 # Import necessary libraries
6 from tensorflow.keras.models import load_model
7
8 #model = pickle.load(open('university.pkl', 'rb'))
9
```

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 your trained model
model = load_model('model.h5')
```

Render HTML page:

```
@app.route('/')
def home():
    return 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.

Retrieves the value from UI:

```
@app.route('/')
def home():
    return render_template('Demo2.html')

@app.route('/y_predict', methods=['POST'])
def y_predict():
    '''
    For rendering results on HTML GUI
    '''
    #min max scaling
    min1=[290.0, 92.0, 1.0, 1.0, 1.0, 6.8, 0.0]
    max1=[340.0, 120.0, 5.0, 5.0, 5.0, 9.92, 1.0]
    k= [float(x) for x in request.form.values()]
    p=[]
    for i in range(7):
        l=(k[i]-min1[i])/(max1[i]-min1[i])
        p.append(l)
    prediction = model.predict([p])
    print(prediction)
    output=prediction[0]
    if(output==False):
        return render_template('noChance.html', prediction_text='You Dont have a chance of gettin
    else:
        return render_template('chance.html', prediction_text='You have a chance of getting admis
if __name__ == "__main__":
    app.run(debug=False)
```

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:

```
36         return render_template
37     else:
38         return render_template
39 if __name__ == "__main__":
40     app.run(debug=False)
41
```

## 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.

```
base) D:\TheSmartBridge\Projects\2. DrugClassification\Drug c
* Serving Flask app "app" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a p
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

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

**UNIVERSITY ADMISSION PREDICTION SYSTEM**

Enter your details and get probability of your admission

Enter GRE Score

Enter TOEFL Score

Select University no

☐ 1

☒ 2

☐ 3

☐ 4

☐ 5

Enter SOP

Enter LOR

Enter CGPA

Research

☐ Research

☒ NO Research

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

## Predicting Chance of Admission

A Machine Learning Web App using Flask.

Prediction : You have a chance 👍



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

---

# Advantages of Admission prediction

## Advantages

- It helps student for making decision for choosing a right college.
- 
- Here the chance of occurrence of error is less when compared with the existing system.
- 
- It is fast, efficient and reliable.
- 
- Avoids data redundancy and inconsistency.
- 
- Very user-friendly.
- 
- Easy accessibility of data.

## Disadvantages

- **Computer Literacy and Internet Access** – In India, though Internet penetration is rather high, Internet connectivity and speed issues are major impediments to bring any real advantage to university applicants. Most rural areas experience

high blackouts and electricity issues. This means, once again candidates in urban districts and areas are placed at a significant advantage.

- **Low Computer Literacy** – Another major concern is the low rate of computer literacy in India. Current estimates say that only about 6.5 percent Indians are computer savvy. A sudden shift to the online admission process is likely to cause confusion and despondency among a great many applicants.
- **Security Concerns** – In a country like India where security fails of online systems have become increasingly common over the years, online applications make it easier for systems to be breached and for applications or scores to be manipulated. The fear that hackers may target universities and educational institutions is a grave one. Unintentional system failures or server crashes may disrupt the entire admission process of universities and educational institutions. Another important concern is the confidentiality of student information and associated security risks involved in online application processing.

Future scope

Using Ground Truth to Calculate Model Performance.

...

1. Accounting for time series and seasonal variability ...

2. Definitions for ground truth vary across the organization. ...
3. Computer vision and NLP models require humans-in-the-loop labeling ...
4. Solving for Ground Truth

## 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.