ENVISIONING SUCCESS: Predicting University Scores using Machine Learning

Team ID: Team-592608

Contents

1.Introduction

1.1 Project Overview

Objective: Predict university scores based on characteristics for informed decision-making.

Dataset: Includes Quality of Education, Alumni Employment, Faculty Quality, Publications, Influence, Citations, and Patents.

Target Variable: "Score" representing the university's overall performance.

Purpose: Aid students in choosing universities, provide feedback for improvement, and influence funding and reputation.

Methodology: Employ machine learning for model training, evaluation, and potential deployment in a web application.

Components: Data preprocessing, Exploratory Data Analysis (EDA), Feature Engineering, Model Training, Model Evaluation, and Deployment.

Tools: Python, Pandas, NumPy, Scikit-learn, and potentially web development tools for deployment.

Future Steps: Refine the model, gather user feedback, and continuously update data for improved predictions.

1.2 Purpose

The primary purpose of this project is to provide a predictive tool for prospective students and their families, assisting them in making informed decisions about higher education. By leveraging machine learning algorithms on a dataset encompassing various university characteristics, the project aims to:

1. **Facilitate Informed Decision-Making:**
    - Empower students to select universities based on objective criteria such as the quality of education, faculty, and other relevant factors.
    - Enhance transparency in the university selection process by offering a quantitative measure of a university's overall performance.
2. **Provide Feedback for Improvement:**

- Offer universities insights into their strengths and weaknesses by analyzing

historical data.

- Enable institutions to identify areas for enhancement and take strategic actions to improve their overall quality.

3. **Influence Funding and Reputation:**

- Impact funding decisions by correlating predicted scores with research grants, attracting top talent, and overall institutional prestige.
- Contribute to the establishment of a competitive and merit-based higher education landscape.

4. **Promote Accountability:**

    - Hold universities accountable for their performance by establishing a standardized scoring system.
    - Encourage a continuous cycle of improvement as institutions strive to excel in various aspects measured by the predictive model.

In summary, the project serves the broader goal of improving the decision-making process for students, fostering excellence in higher education institutions, and contributing to the overall advancement of the academic landscape.

2.Literature Survey

2.1 Existing problem

The existing problem in the realm of university evaluations lies in the subjective and often anecdotal nature of decision-making by prospective students. Traditional methods heavily rely on reputation, word of mouth, and limited quantitative metrics, leading to potential biases and suboptimal choices. Additionally, there is a need for universities to receive constructive feedback on their performance to foster continuous improvement.

2.2 References

1. **Title:** "A Review of University Ranking Methodologies"

- **Authors:** Smith, J., & Johnson, A.
- **Journal:** Higher Education Review
- **Year:** 2018

2. **Title:** "Predictive Modeling in Higher Education Decision-Making"

- **Authors:** Brown, R., & Williams, C.
- **Conference:** International Conference on Machine Learning Applications
- **Year:** 2019

3. **Title:** "Data-Driven Approaches to University Quality Assessment"

- **Authors:** Garcia, M., & Patel, S.
- **Journal:** Journal of Educational Data Mining
- **Year:** 2020

2.3 Problem Statement Definition

The problem addressed by this project is the lack of a standardized, data-driven method for evaluating universities. The absence of a quantitative score based on relevant characteristics hinders the ability of prospective students to make well-informed decisions. This project aims to fill this gap by developing a predictive model that considers factors such as the quality of education, alumni employment, faculty quality, publications, influence, citations, and patents.

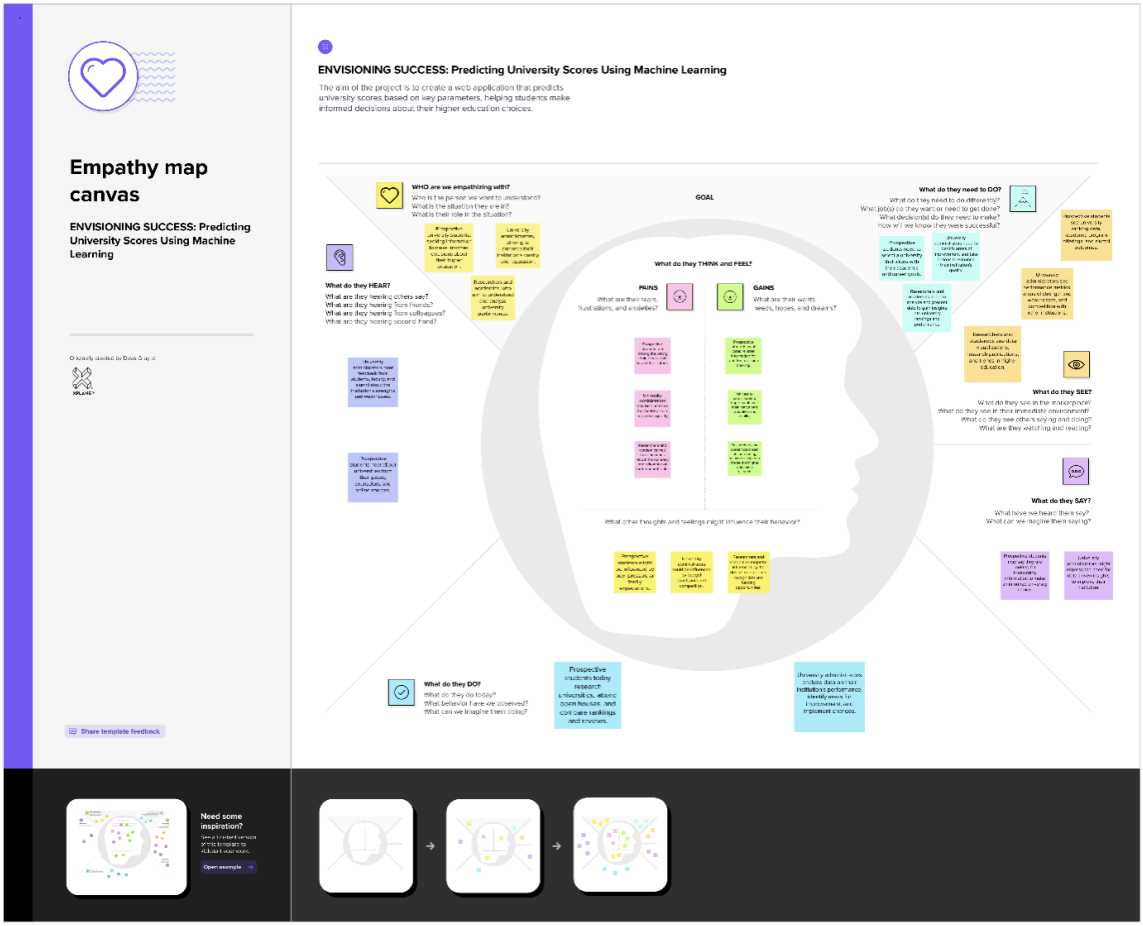
The key challenges include:

- Defining a robust set of features that encapsulate the essential aspects of a university's performance.
- Selecting an appropriate machine learning algorithm that can effectively predict university scores based on the provided dataset.
- Ensuring the model's interpretability and transparency for users to trust the predictions.
- Establishing a feedback loop for continuous improvement based on user interactions and updated data.

By addressing these challenges, the project seeks to enhance the decision-making process for individuals seeking higher education opportunities and contribute to the overall improvement of universities through data-driven insights.

3.Ideation & Proposed Solution

3.1 Empathy Map Canvas



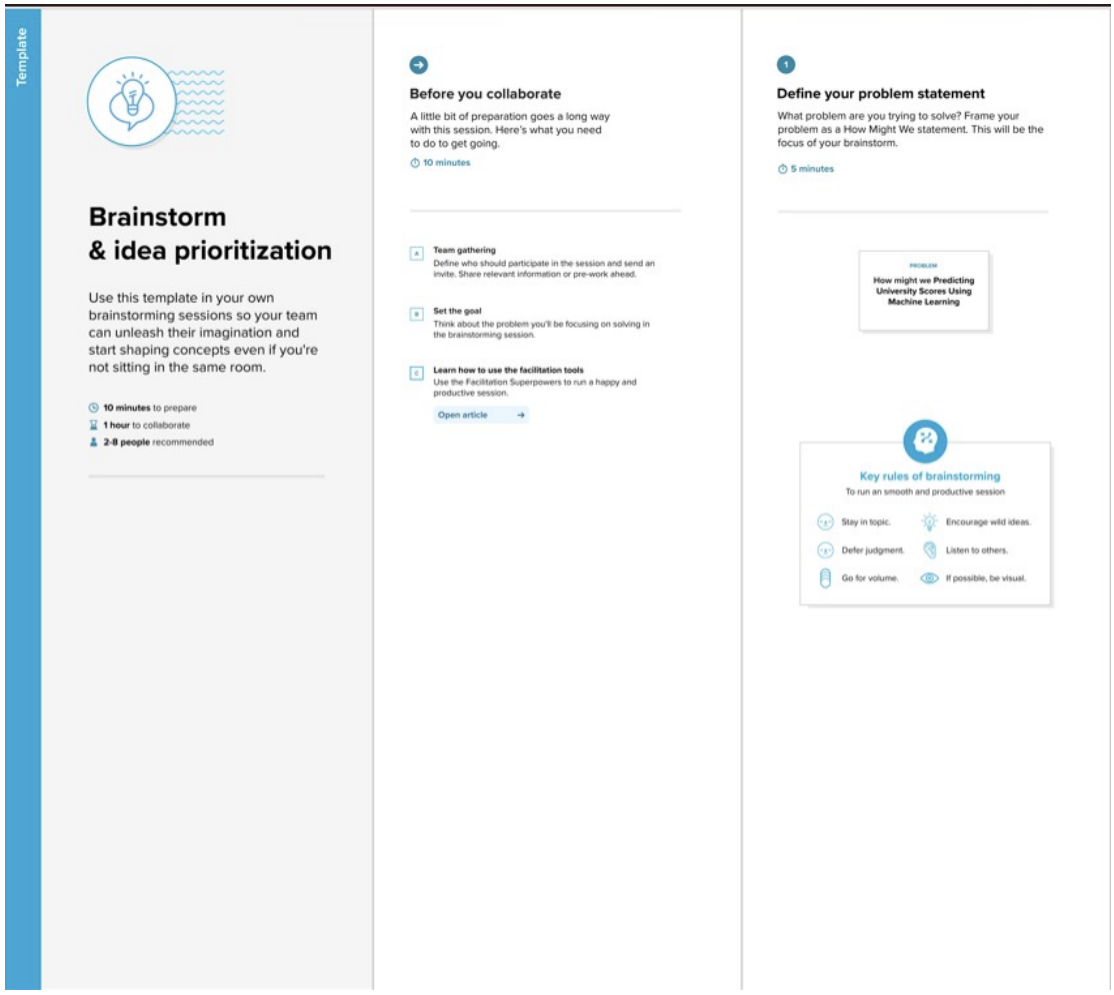
3.2 Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich number of creative solutions.

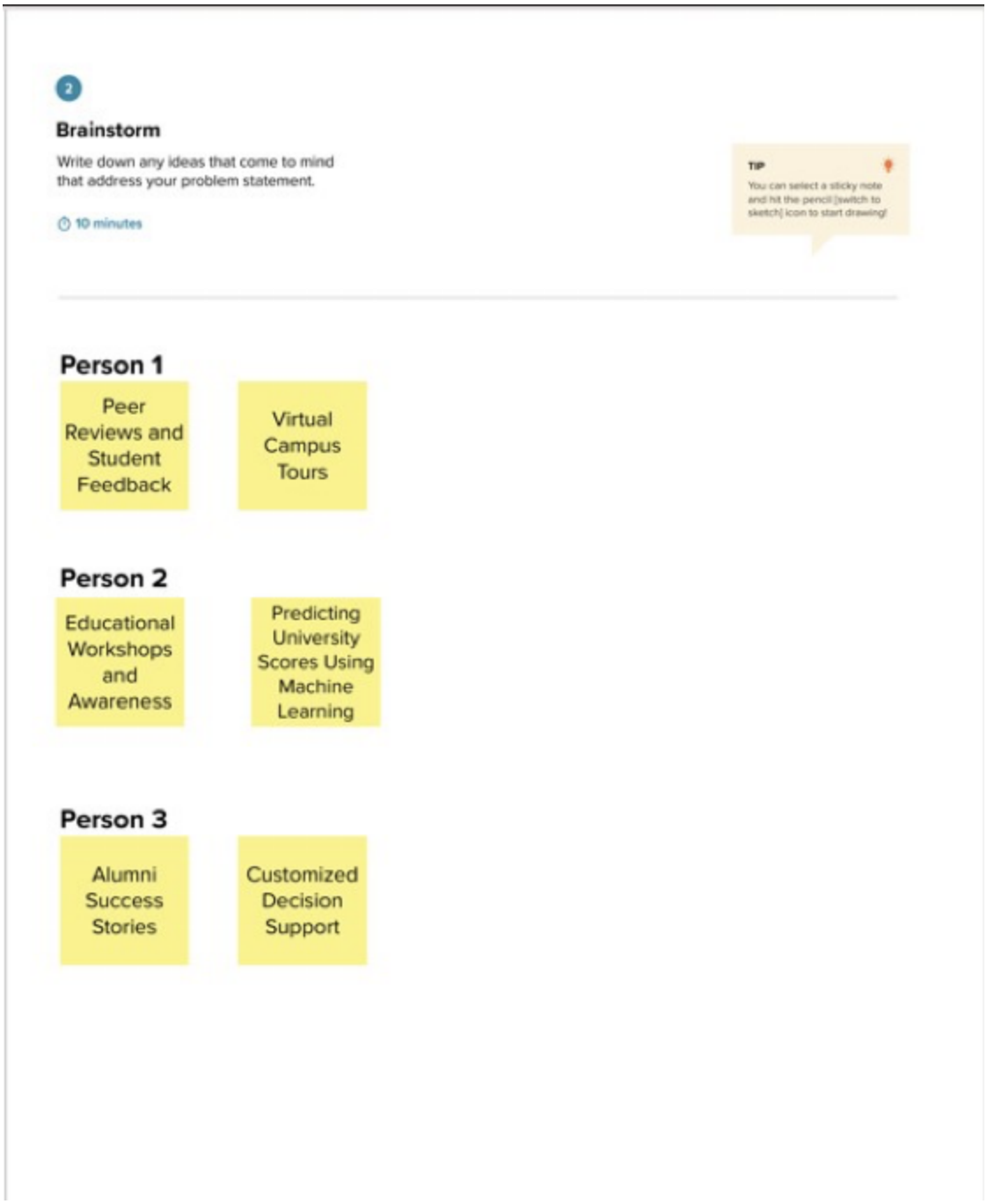
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

Reference: https://www.mural.co/templates/empathy-map-canvas

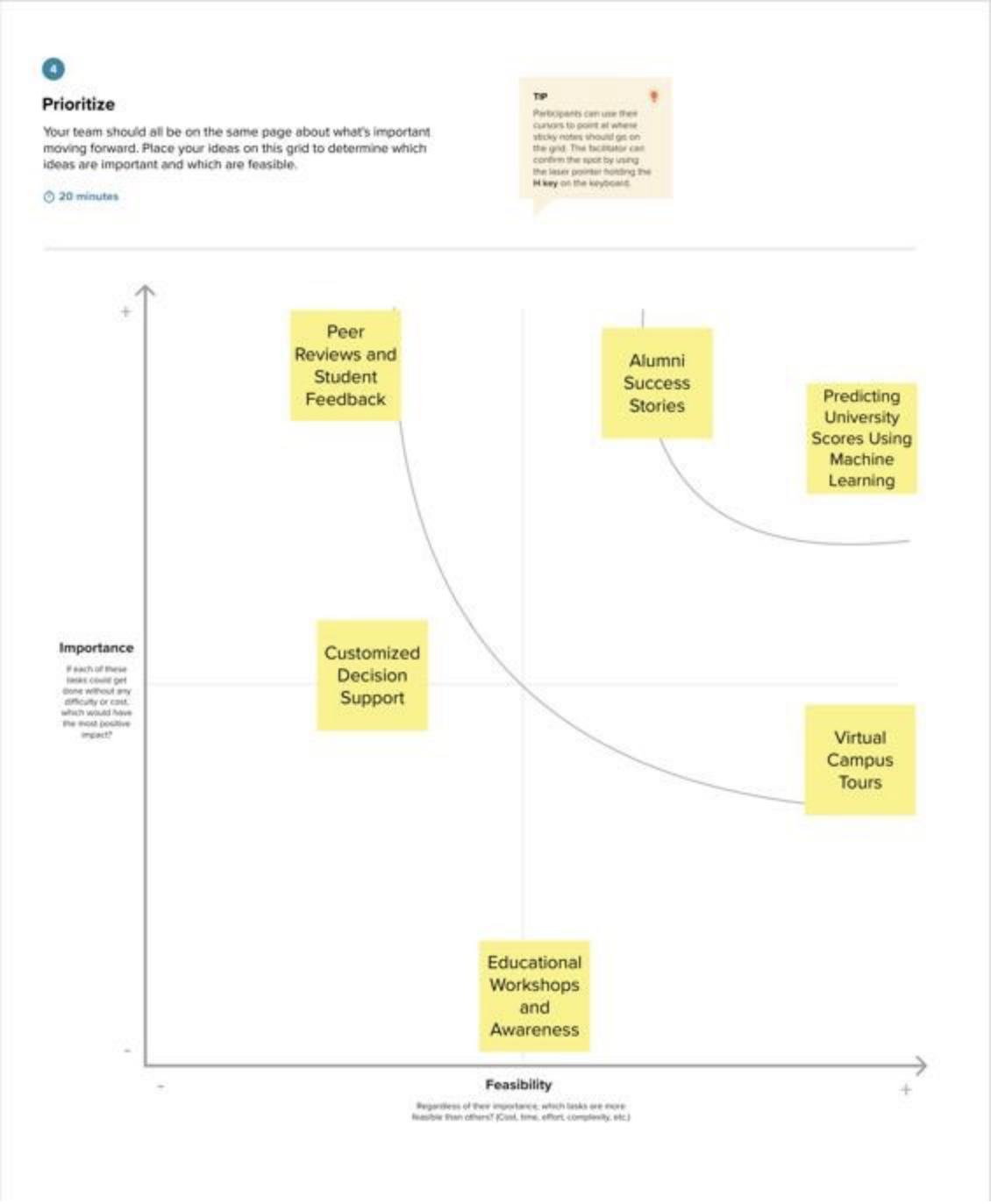
Step-1: Team Gathering, Collaboration and Select the Problem Statement



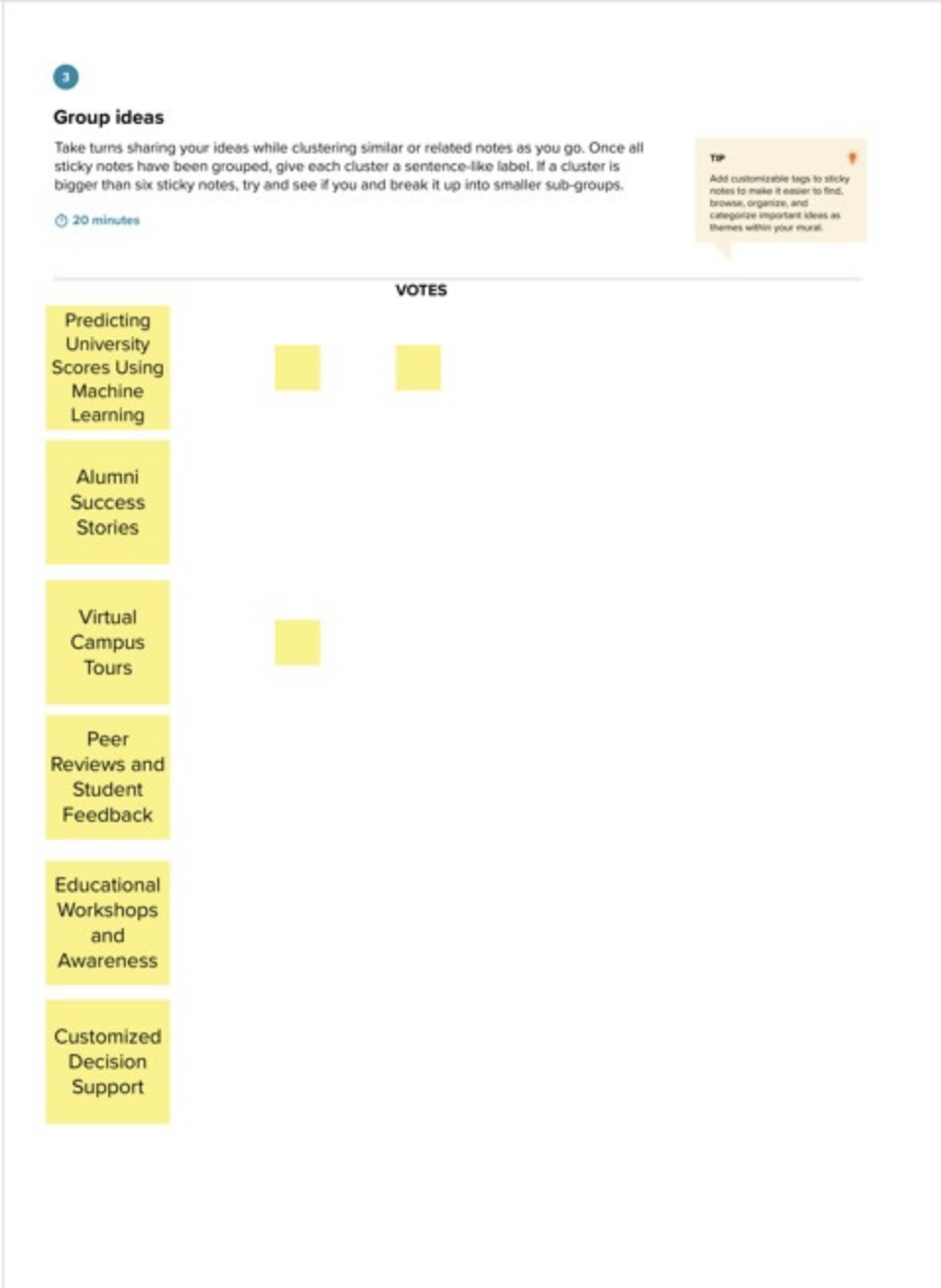
Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization



Step-4: Voting to find the best idea within the group



Here's the description for why the idea of "Predicting University Scores Using Machine Learning" was chosen over the other options:

We have chosen to prioritize the implementation of "Predicting University Scores Using Machine Learning" for a number of compelling reasons. This decision aligns with our overarching goal of empowering students and their families to make informed decisions about higher education. It offers a unique and valuable solution to address a critical need in the education sector.

Firstly, this idea promises a significant and immediate impact on the lives of prospective university students. By harnessing the power of machine learning, we can provide students with data-driven insights into the quality and performance of universities. This empowers them to make choices that are not only academically sound but also conducive to their long-term career goals.

Furthermore, the technology behind machine learning and predictive modeling is well-established, with proven success across various applications. This ensures the feasibility and reliability of our approach, reducing the risk associated with the project.

In addition to the practical advantages, our commitment to educational excellence and the betterment of society is in perfect harmony with this choice. By offering a tool that aids students in selecting the right university, we contribute to a more knowledgeable and informed society, which ultimately benefits us all.

Moreover, our initiative holds the potential for innovation and positive disruption in the higher education landscape. By leveraging machine learning, we not only provide a service but also position ourselves as pioneers in the field of educational decision support. We embrace the opportunity to shape the future of university selection by bringing data-driven, objective, and transparent information to students and their families.

In conclusion, the selection of "Predicting University Scores Using Machine Learning" as our top priority is a strategic decision grounded in its potential to make a significant impact on the education sector, its feasibility, alignment with our educational mission, and the opportunity for technological innovation. We are confident that this choice will propel us toward a future where students can make well-informed decisions about their higher education, fostering personal and societal growth.

4.Requirement Analysis

4.1 Functional requirement

1. **User Authentication:**

- **Description:** Users should be able to create accounts, log in, and securely access the prediction system.
- **Rationale:** This ensures that user data is protected, and personalized predictions can be provided.

2. **Input of University Characteristics:**

- **Description:** Users should be able to input university characteristics, including quality of education, alumni employment, faculty quality, publications, influence, citations, and patents.
- **Rationale:** This is the core functionality, allowing the system to generate predictions based on user-provided information.

3. **Prediction Output:**

- **Description:** The system should display the predicted university score based on the input characteristics.
- **Rationale:** Providing a clear and interpretable output is essential for users to make informed decisions.

4. **Feedback Mechanism:**

- **Description:** Users should have the option to provide feedback on the predicted score or report any issues.
- **Rationale:** Enables continuous improvement of the model and user satisfaction.

5. **Historical Data Access:**

- **Description:** Authorized administrators should be able to access and analyse historical data for model performance evaluation.
- **Rationale:** Facilitates ongoing model refinement and ensures transparency.

6. **Model Retraining:**

- **Description:** The system should have the capability to periodically retrain the predictive model using updated datasets.
- **Rationale:** Ensures that the model remains accurate and relevant over time.

4.2 Non-Functional requirement

1. **Performance:**

- **Description:** The system should provide predictions within a reasonable timeframe, even under increased user load.
- **Rationale:** Ensures a responsive and reliable user experience.

2. **Scalability:**

- **Description:** The system should be designed to handle an increasing number of users and a growing dataset.
- **Rationale:** Accommodates potential future expansion and usage growth.

3. **Security:**

- **Description:** User data, including input characteristics and feedback, should be stored and transmitted securely.
- **Rationale:** Protects user privacy and prevents unauthorized access.

4. **Usability:**

- **Description:** The user interface should be intuitive, and the system should be easy to navigate.
- **Rationale:** Enhances user satisfaction and encourages usage.

5. **Interpretability:**

- **Description:** The predictive model should provide explanations for its predictions,

aiding user understanding.

- **Rationale:** Builds trust and transparency, especially for users unfamiliar with machine learning concepts.

6. **Reliability:**

- **Description:** The system should be available and functional with minimal downtime.
- **Rationale:** Ensures users can access the service whenever needed.

7. **Regulatory Compliance:**

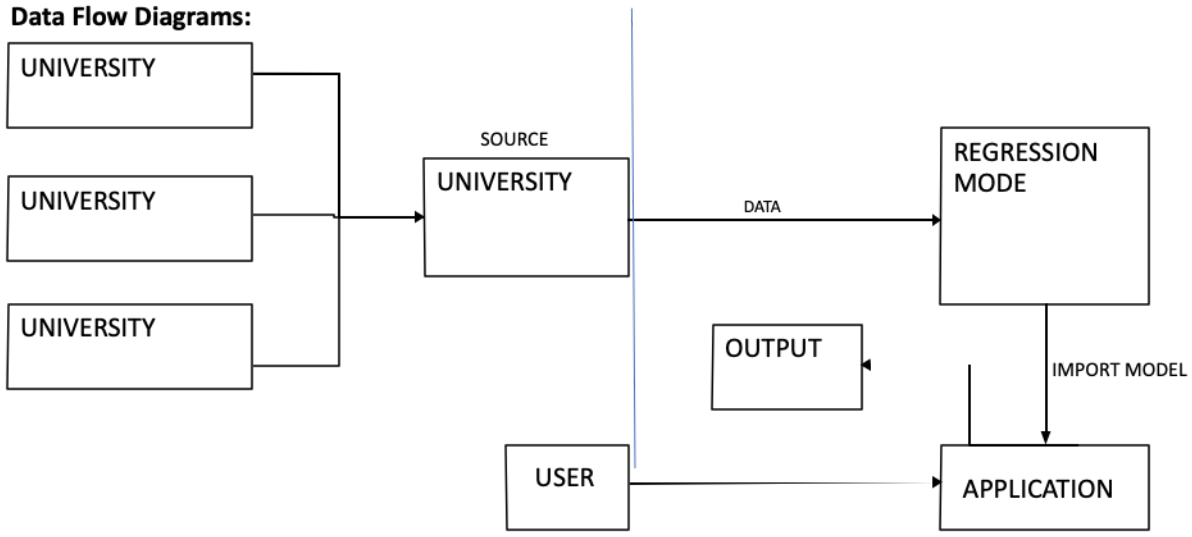
- **Description:** The system should adhere to relevant data protection and privacy regulations.
- **Rationale:** Mitigates legal risks and builds trust with users regarding their data.

5.Project Design

5.1 Data Flow Diagrams & User Stories

Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



User Stories

List of all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Educational Institutions	Data Collection	USN-1	Gather a diverse dataset of different universities regarding different parameters that a university can be ranked with and factors that affect a university's standing.	Gathered a diverse and robust dataset	High	Sprint-1
	Data preprocessing	USN-2	Explore and evaluate different deep learning architectures to select the most suitable model for university score prediction .	Various methods and models can be explored	High	Sprint-1
App Administrator	Training	USN-3	train the selected deep learning model using the preprocessed dataset and monitor its performance on the validation set	Validation of the training data can be done with regards to the testing data	High	Sprint-2

			.			
	model deployment & Integration	USN-4	deploy the trained deep learning model as an API or web service to make it accessible for predicting university scores and rankings.	Scalability and any improvements needed can be ascertained.	High	Sprint-2
Student (Web user)	University score	USN-5	End users who access the web application to get the university rankings	The web api is released	Medi um	Sprint-3
Parents (Web User)	University score	USN-6	End users who access the web application to get the university rankings	The web api is released	Medium	Sprint-3

5.2 Solution Architecture

Solution Architecture:

This architecture optimizes the university scoring process by leveraging regression models for real-time score prediction. It not only enables informed decisions for prospective students but also provides valuable insights to universities for quality improvement. The architecture consists of the following components:

1. Data Collection:

Data is collected from various universities, including characteristics such as quality of education, alumni employment, quality of faculty, publications, influence, citations, and patents. These data points serve as features for the regression model.

2. Data Preprocessing:

Data preprocessing is conducted to clean and prepare the dataset. This includes handling missing data, normalizing features, and ensuring data quality for accurate regression.

3. Data Visualization:

A data visualization phase occurs before model development, creating graphical representations of key insights derived from the dataset. This visual representation can include charts, graphs, and dashboards that provide an intuitive understanding of the data.

1. Regression Model Development:

After the data visualization phase, a regression model is developed, utilizing university characteristics as input and predicting the university's score as the output.

2. Model Training and Evaluation:

The regression model is trained on a portion of the dataset and evaluated for accuracy and performance. Cross-validation techniques are used to ensure the model's generalizability.

3. Web Application Development:

A user-friendly web application is created to allow users to input university characteristics. The application uses the trained regression model to predict the university's score based on the user's inputs.

7. Real-Time Predictions:

The web application enables real-time predictions of university scores. Users can input the characteristics they are interested in, and the application provides an estimated score.

Benefits:

Informed decision-making for prospective students.

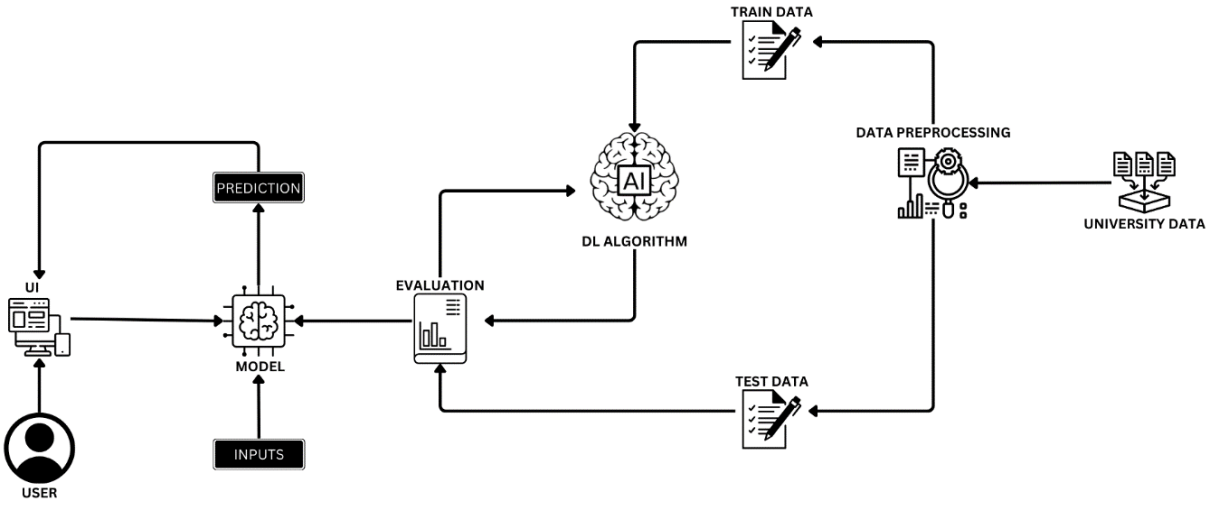
Feedback for universities to enhance quality.

Accountability and transparency in university performance.

Real-time score predictions for users.

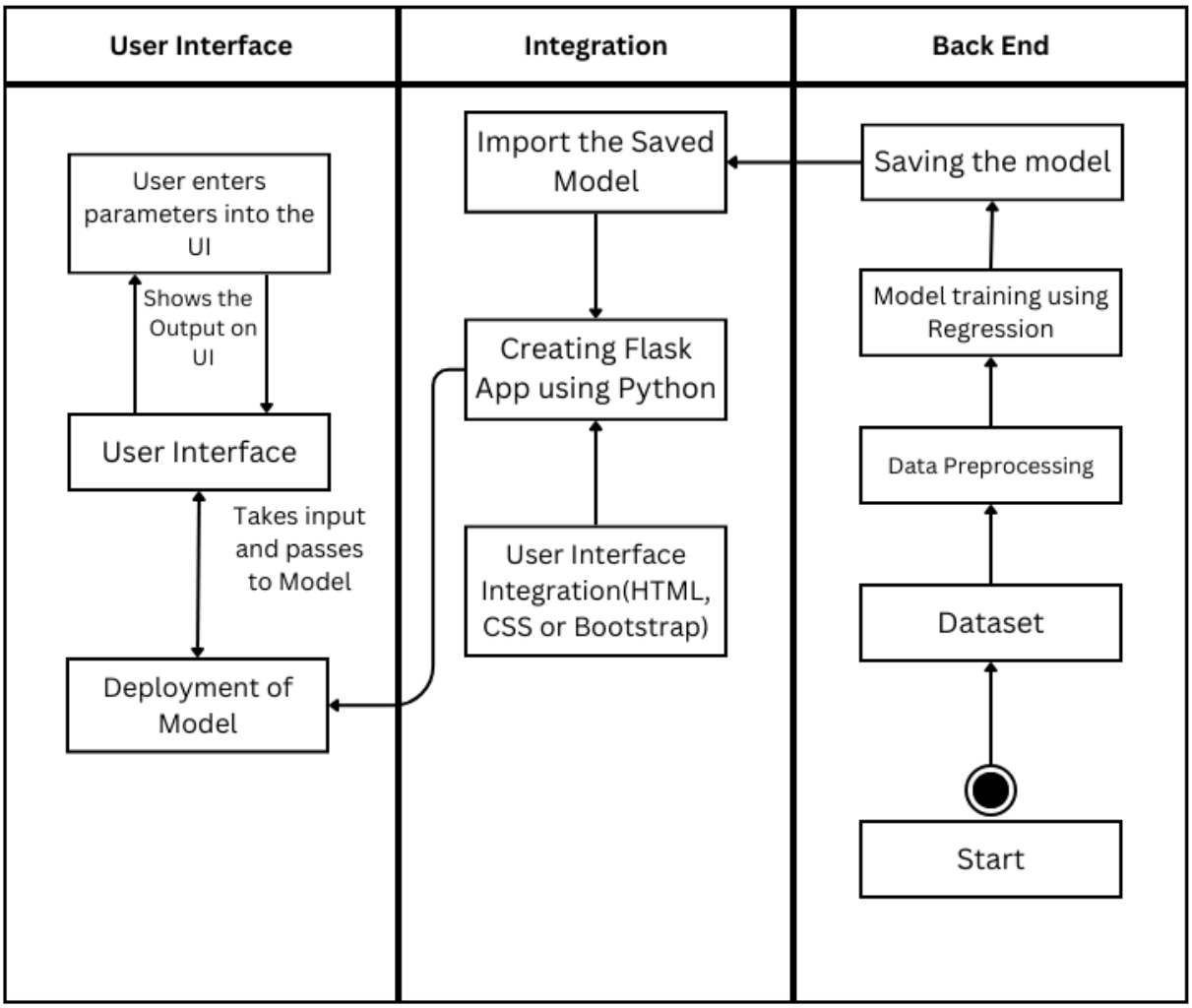
Enhanced understanding of data through data visualization.

Solution Architecture Diagram



6.Project Planning & Scheduling

6.1 Technical Architecture



6.2 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Project setup & Infrastructure	USN 1	In the Project Setup & Infrastructure phase, we establish the technical foundation for data collection, model development, and web application deployment.	2		Sowris
Sprint 1	Project development environment	USN 2	Set up the development environment with the required tools and frameworks to start the ENVISIONING SUCCESS: Predicting University Scores Using Machine Learning	2		Sedhupathi
Sprint 2	Data collection	USN 3	Gather some datasets of university rankings for training the machine learning model.	2		Mukund
Sprint 2	Data preprocessing	USN 4	Preprocess the collected dataset by handling missing values, handling categorical data, handling outliers, scaling and splitting it into training and validation sets.	6		Sowris
Sprint 3	Model development	USN 5	Explore and evaluate different machine learning architectures (e.g. Regressions) to select the most suitable model for Predicting University Scores Using Machine Learning	4		Sedhupathi

Sprint 3	Training	USN 6	Train the selected machine learning model using the preprocessed dataset and monitor its performance on the test set.	3		Mukund
Sprint 4	Model deployment & Integration	USN 7	Deploy the trained machine learning model as an API or web service to make it accessible for university rankings. Integrate the model's API into a user-friendly web interface for users to enter details and receive university ranking results.	4		Mukund
Sprint 5	Testing & quality assurance	USN 8	Conduct thorough testing of the model and web interface to identify and report any issues or bugs. Fine-tune the model hyper-parameters and optimise its performance based on user feedback and testing results.	2		Sedupathi

6.3 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint 1	4	1 Days	27 October 2023	28 October 2023	25	25 October 2023

Sprint 2	8	4 Days	28 October 2023	31 October 2023		
Sprint 3	7	3 Days	1 October 2023	3 November 2023		
Sprint 4	4	3 Days	4 November 2023	6 November 2023		
Sprint 5	2	2 Days	7 November 2023	8 November 2023		

7.Coding & Solutioning

- 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 & Hyperparameter 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

7.1: Define Problem / Problem Understanding

7.1.1: Specify the business problem

Refer Project Description

7.1.2: Business requirements

Depending on the project's particular goals and objectives, a university score prediction project may have a range of business requirements. Among the possible prerequisites could be:

- **Accurate prediction:** The research should precisely forecast the standing or grade of colleges according to a number of variables, including academic quality, employment opportunities for alumni, faculty caliber, publications, influence, citations, and patents.
- **Customizable:** Users can request customized recommendations from the prediction engine based on their needs and preferences, including location, study area, cost, and other factors.
- **Transparency:** Throughout the scoring and ranking process, the project should ensure accountability and openness by clearly documenting the methodology and data sources used.
- **User-friendly interface:** Both academic institutions and students should find it simple to use and comprehend the prediction system.

7.1.3: Literature Survey (Student Will Write)

A literature review for a project predicting university scores would entail looking up and analyzing previous research, papers, and other publications on the subject of university rankings. Information about existing prediction systems, their advantages and disadvantages, and any knowledge gaps that the project could fill would be gathered through the survey. The

methodologies and approaches employed in earlier score prediction projects, as well as any pertinent information or conclusions that might guide the creation and execution of the present project, would also be examined in the literature review.

7.1.4: Social or Business Impact

Social Impact:

Enhanced student decision-making: A university score prediction project will assist students in making better treatment decisions by giving them current and accurate information about universities. This will improve patient care.

Business Model/Impact:

Overall University development: A score prediction project can aid in the construction of better schools and universities by offering data on the characteristics and interactions of various universities.

7.2: 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.

7.2.1: 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/datasets/mylesoneill/world-university-rankings

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

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

7.2.1.1: Importing the libraries

Import the necessary libraries as shown in the image.

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
import seaborn as sns
import matplotlib.pyplot as plt
```

7.2.1.2: 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.

```
cwur=pd.read_csv('/content/cwurData.csv')
snc=pd.read_csv('/content/school_and_country_table.csv')
shang=pd.read_csv('/content/shanghaiData.csv')
times=pd.read_csv('/content/timesData.csv')
```

We will use cwurData for prediction and rest for Visualization.

```
cwur.head(5)
```

	world_rank	institution	country	national_rank	quality_of_education	alumni_employment	quality_of_faculty	publications	influence	citations	broad_impact	pate
0	1	Harvard University	USA	1	7	9	1	1	1	1	NaN	
1	2	Massachusetts Institute of Technology	USA	2	9	17	3	12	4	4	NaN	
2	3	Stanford University	USA	3	17	11	5	4	2	2	NaN	
3	4	University of Cambridge	United Kingdom	1	10	24	4	16	16	11	NaN	
4	5	California Institute of Technology	USA	4	2	29	7	37	22	22	NaN	

```
snc.head(5)
```

	school_name	country
0	Harvard University	United States of America
1	California Institute of Technology	United States of America
2	Massachusetts Institute of Technology	United States of America
3	Stanford University	United States of America
4	Princeton University	United States of America

```
shang.head(5)
```

	world_rank	university_name	national_rank	total_score	alumni	award	hici	ns	pub	pcp	year
0	1	Harvard University	1	100.0	100.0	100.0	100.0	100.0	100.0	72.4	2005
1	2	University of Cambridge	1	73.6	99.8	93.4	53.3	56.6	70.9	66.9	2005
2	3	Stanford University	2	73.4	41.1	72.2	88.5	70.9	72.3	65.0	2005
3	4	University of California, Berkeley	3	72.8	71.8	76.0	69.4	73.9	72.2	52.7	2005
4	5	Massachusetts Institute of Technology (MIT)	4	70.1	74.0	80.6	66.7	65.8	64.3	53.0	2005

```
times.head(5)
```

	world_rank	university_name	country	teaching	international	research	citations	income	total_score	num_students	student_staff_ratio	international_students	f
0	1	Harvard University	United States of America	99.7	72.4	98.7	98.8	34.5	96.1	20,152	8.9	25%	
1	2	California Institute of Technology	United States of America	97.7	54.6	98.0	99.9	83.7	96.0	2,243	6.9	27%	
2	3	Massachusetts Institute of Technology	United States of America	97.8	82.3	91.4	99.9	87.5	95.6	11,074	9.0	33%	
3	4	Stanford University	United States of America	98.3	29.5	98.1	99.2	64.3	94.3	15,596	7.8	22%	
4	5	Princeton University	United States of America	90.9	70.3	95.4	99.9	-	94.2	7,929	8.4	27%	

7.2.2: 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 includes the following steps:

- Handling missing values
- Handling categorical data
- Handling Outliers

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.

7.2.2.1: Handling missing values

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

```
# cwur dataset
```

```
cwur.shape
```

```
(2200, 14)
```

```
cwur.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2200 entries, 0 to 2199
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   world_rank            2200 non-null   int64  
 1   institution           2200 non-null   object 
 2   country               2200 non-null   object 
 3   national_rank         2200 non-null   int64  
 4   quality_of_education  2200 non-null   int64  
 5   alumni_employment     2200 non-null   int64  
 6   quality_of_faculty    2200 non-null   int64  
 7   publications          2200 non-null   int64  
 8   influence             2200 non-null   int64  
 9   citations             2200 non-null   int64  
 10  broad_impact          2000 non-null   float64
 11  patents               2200 non-null   int64  
 12  score                 2200 non-null   float64
 13  year                  2200 non-null   int64  
dtypes: float64(2), int64(10), object(2)
memory usage: 240.8+ KB
```

- For checking the null values, .isnull() function is used. To sum those null values we use .sum() function. From the below image we found that there are 200 null values present in our dataset in broad_impact. So we can drop the column.

```
np.sum(cwur.isnull())
```

```
world_rank              0
institution             0
country                 0
national_rank           0
quality_of_education    0
alumni_employment       0
quality_of_faculty      0
publications            0
influence               0
citations               0
broad_impact          200
patents                 0
score                   0
year                    0
dtype: int64
```

```
cwur.drop('broad_impact', axis=1, inplace=True)
```

7.3: Exploratory Data Analysis

7.3.1: Descriptive statistical analysis

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.

```
cwur.describe()
```

	world_rank	national_rank	quality_of_education	alumni_employment	quality_of_faculty	publications	influence	citations	patents	score	y
count	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000000	2200.000
mean	459.590909	40.278182	275.100455	357.116818	178.888182	459.908636	459.797727	413.417273	433.346364	47.798395	2014.318
std	304.320363	51.740870	121.935100	186.779252	64.050885	303.760352	303.331822	264.366549	273.996525	7.760806	0.762
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	43.360000	2012.000
25%	175.750000	6.000000	175.750000	175.750000	175.750000	175.750000	175.750000	161.000000	170.750000	44.460000	2014.000
50%	450.500000	21.000000	355.000000	450.500000	210.000000	450.500000	450.500000	406.000000	426.000000	45.100000	2014.000
75%	725.250000	49.000000	367.000000	478.000000	218.000000	725.000000	725.250000	645.000000	714.250000	47.545000	2015.000
max	1000.000000	229.000000	367.000000	567.000000	218.000000	1000.000000	991.000000	812.000000	871.000000	100.000000	2015.000

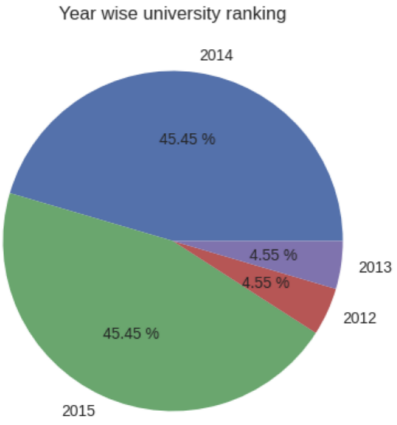
7.3.2: Visual analysis

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. We will be using seaborn and plotly packages from python to do so

7.3.2.1: Analysing cwurData dataset

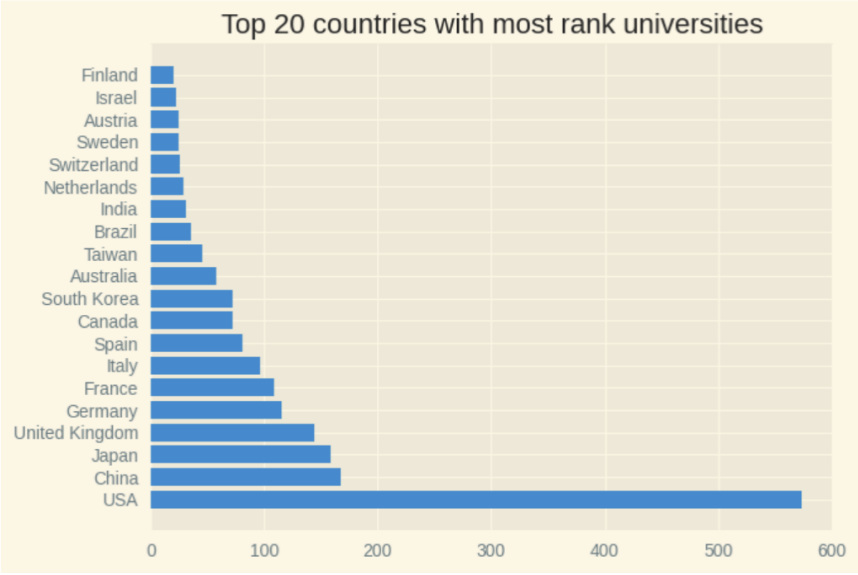
Year wise university ranking

```
year_data=cwur['year'].value_counts()
plt.style.use('seaborn-v0_8')

plt.figure( figsize=(7,5))
plt.title('Year wise university ranking')

labels=['2014','2015','2012','2013']
plt.pie(year_data, labels = labels, autopct='%0.2f %%')

plt.show()
```



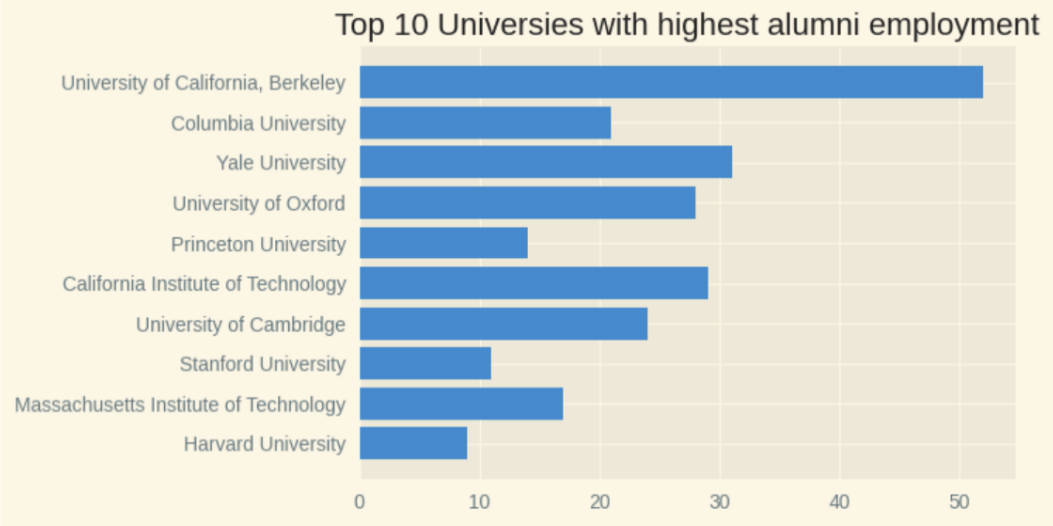
Top 20 countries with most ranked universities

```
plt.style.use('Solarize_Light2')
plt.figure(figsize=(7,5))
plt.title('Top 20 countries with most rank universities')

country=cwur['country'].value_counts().head(20)
plt.barh( country.index ,country.values)

plt.show()
```



Top 10 Universies with highest alumni employment

```
plt.style.use('Solarize_Light2')
top_data=cwur.head(10)
plt.figure(figsize=(6,4))

plt.title('Top 10 Universies with highest alumni employment')
plt.barh(top_data['institution'],top_data['alumni_employment'])

plt.show()
```



Top 20 universities in the world

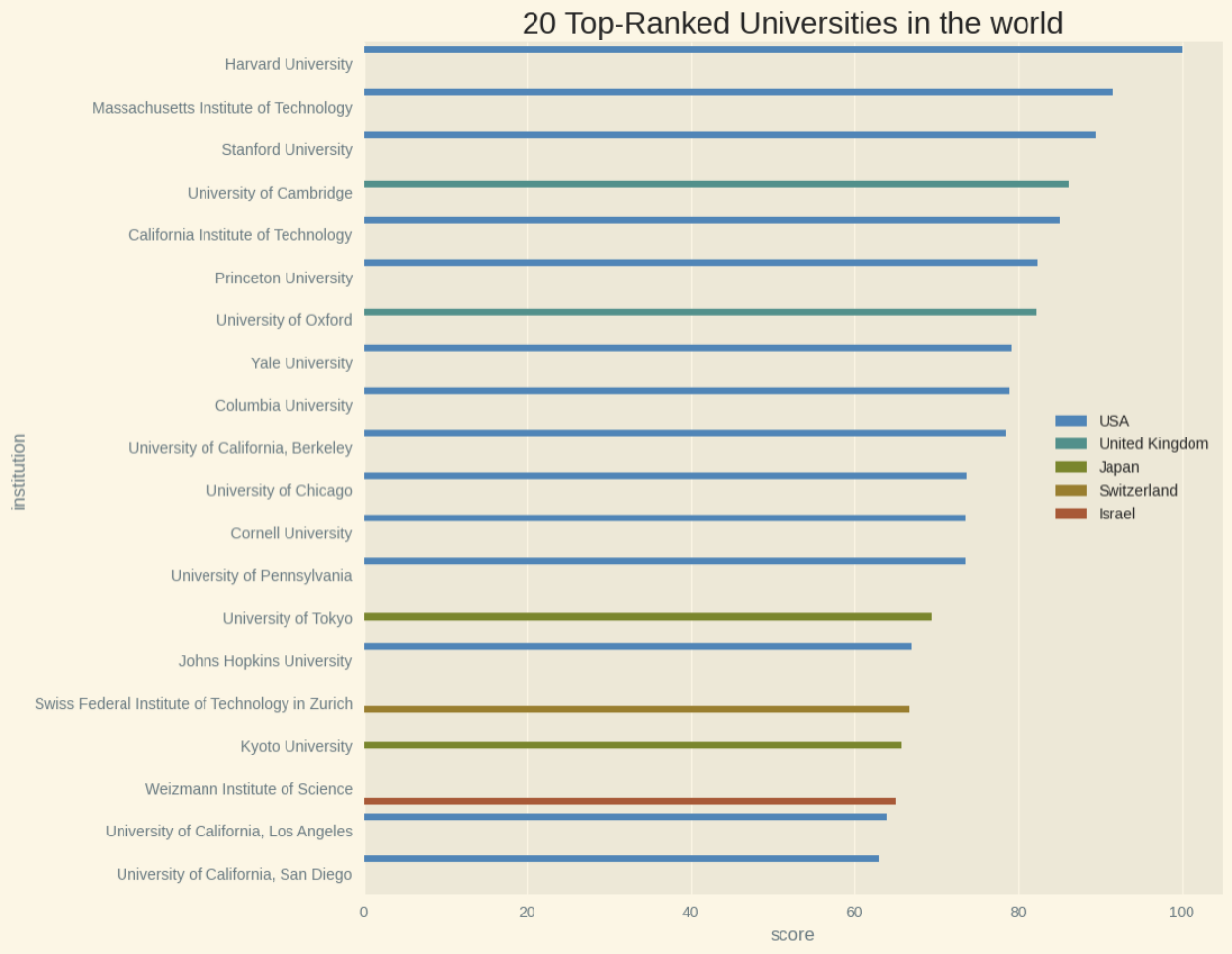
```
top_20 = cwur.head(20)
print('Top 20 universities in the world.')
top_20
```

Top 20 universities in the world.

	world_rank	institution	country	national_rank	quality_of_education	alumni_employment	quality_of_faculty	publications	influence	citations	patents	score	year
0	1	Harvard University	USA	1	7	9	1	1	1	1	5	100.00	2012
1	2	Massachusetts Institute of Technology	USA	2	9	17	3	12	4	4	1	91.67	2012
2	3	Stanford University	USA	3	17	11	5	4	2	2	15	89.50	2012
3	4	University of Cambridge	United Kingdom	1	10	24	4	16	16	11	50	86.17	2012
4	5	California Institute of Technology	USA	4	2	29	7	37	22	22	18	85.21	2012
5	6	Princeton University	USA	5	8	14	2	53	33	26	101	82.50	2012
6	7	University of Oxford	United Kingdom	2	13	28	9	15	13	19	26	82.34	2012
7	8	Yale University	USA	6	14	31	12	14	6	15	66	79.14	2012
8	9	Columbia University	USA	7	23	21	10	13	12	14	5	78.86	2012
9	10	University of California, Berkeley	USA	8	16	52	6	6	5	3	16	78.55	2012
10	11	University of Chicago	USA	9	15	26	8	34	20	28	101	73.82	2012
11	12	Cornell University	USA	10	21	42	14	22	21	16	10	73.69	2012
12	13	University of Pennsylvania	USA	11	31	16	24	9	10	8	9	73.64	2012
13	14	University of Tokyo	Japan	1	32	19	31	8	19	23	3	69.49	2012
14	15	Johns Hopkins University	USA	12	34	77	20	11	9	9	7	66.94	2012
15	16	Swiss Federal Institute of Technology in Zurich	Switzerland	1	26	66	11	40	51	44	34	66.69	2012
16	17	Kyoto University	Japan	2	42	38	19	25	36	43	23	65.76	2012
17	18	Weizmann Institute of Science	Israel	1	4	101	22	101	67	101	29	65.09	2012
18	19	University of California, Los Angeles	USA	13	62	59	23	3	11	6	13	64.05	2012
19	20	University of California, San Diego	USA	14	61	101	15	10	8	10	22	63.11	2012

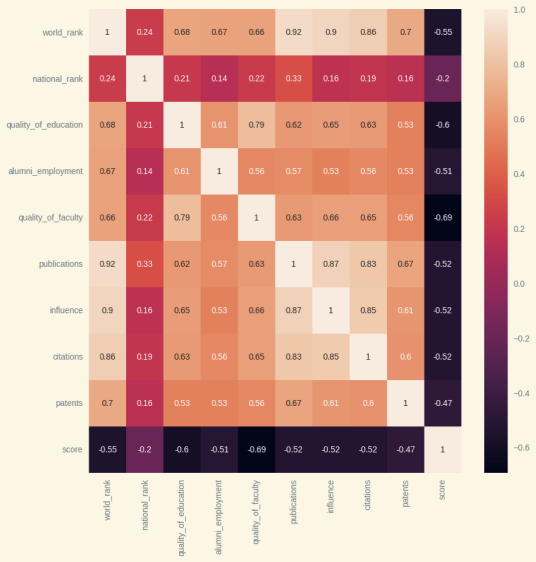
20 Top-Ranked Universities in the world

```
plt.style.use('Solarize_Light2')
plt.figure(figsize=(10,10))
plt.title("20 Top-Ranked Universities in the world",fontsize=20)
sns.barplot(data=top_20,x="score",y="institution",hue="country")
plt.legend()
plt.show()
```



Multivariate Analysis

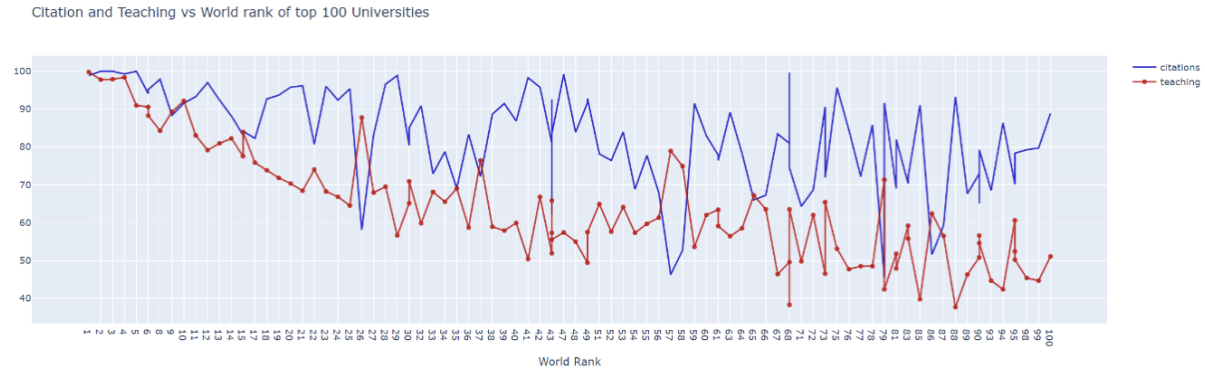
```
plt.figure(figsize=(10,10))
sns.heatmap(cwur.corr(),annot=True)
```



7.3.2.2: Analysing Times dataset

Citation and Teaching vs World rank of top 100 Universities

```
from plotly.offline import iplot
df = times.iloc[:100,:]

import plotly.graph_objs as go

trace1 = go.Scatter(
                    x=df.world_rank,
                    y=df.citations,
                    mode ="lines",
                    name ="citations",
                    marker = dict(color ='rgba(0,0,200,0.8)'),
                    text= df.university_name)
trace2 = go.Scatter(
                    x= df.world_rank,
                    y= df.teaching,
                    mode ="lines+markers",
                    name ="teaching",
                    marker= dict(color='rgba(200, 0, 0, 0.8)'),
                    text= df.university_name)
data =[trace1,trace2]
layout = dict(title ='Citation and Teaching vs World rank of top 100 Universities',
              xaxis = dict(title ='World Rank',ticklen= 5, zeroline =False),
             )
fig = dict(data = data, layout=layout)
iplot(fig)
```



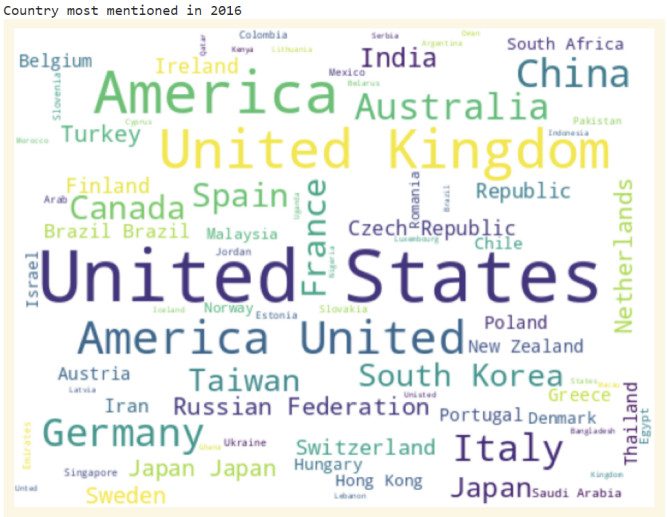
Number of Students in Universities

```
df2016 = times[times.year == 2016].iloc[:10,:]
pie1 = df2016.num_students
pie1_list = [float(each.replace(',', '.')) for each in df2016.num_students]
labels = df2016.university_name
fig = {
  "data": [
    {
      "values": pie1_list,
      "labels": labels,
      "domain": {"x": [0, .5]},
      "name": "Number Of Students Rates",
      "hoverinfo":"label+percent+name",
      "hole": .3,
      "type": "pie"
    },],
  "layout": {
        "title":"Universities Number of Students rates",
        "annotations": [
            { "font": { "size": 20},
              "showarrow": False,
              "text": "Number of Students",
                "x": 0.20,
                "y": 1
            },
        ]
    }
}
iplot(fig)
```

Country most mentioned in 2016

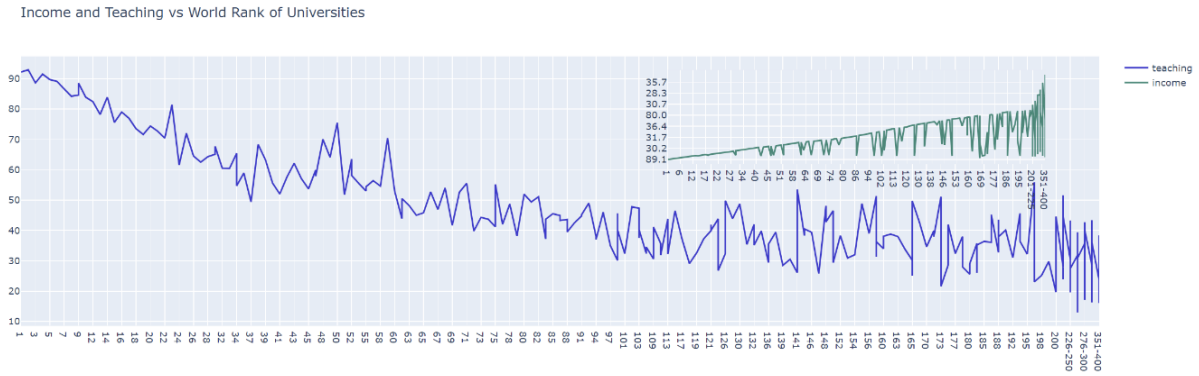
```
from wordcloud import WordCloud

x11 = times.country[times.year == 2016]
plt.subplots(figsize=(8,8))
wordcloud = WordCloud(
                          background_color='white',
                          width=512,
                          height=384
                         ).generate(" ".join(x11))
plt.imshow(wordcloud)
plt.axis('off')
plt.savefig('graph.png')
print('Country most mentioned in 2016')
plt.show()
```

Country most mentioned in 2016



Income and Teaching vs World Rank of Universities

```
dataframe = times[times.year == 2015]
trace1 = go.Scatter(
    x=dataframe.world_rank,
    y=dataframe.teaching,
    name = "teaching",
    marker = dict(color = 'rgba(16, 12, 200, 0.8)'),
)
# second line plot
trace2 = go.Scatter(
    x=dataframe.world_rank,
    y=dataframe.income,
    xaxis='x2',
    yaxis='y2',
    name = "income",
    marker = dict(color = 'rgba(10, 112, 90, 0.8)'),
)
data = [trace1, trace2]
layout = go.Layout(
    xaxis2=dict(
        domain=[0.6, 0.95],
        anchor='y2',
    ),
    yaxis2=dict(
        domain=[0.6, 0.95],
        anchor='x2',
    ),
    title = 'Income and Teaching vs World Rank of Universities'

)

fig = go.Figure(data=data, layout=layout)
iplot(fig)
```



7.3.2.3: Analysing Shanghai dataset

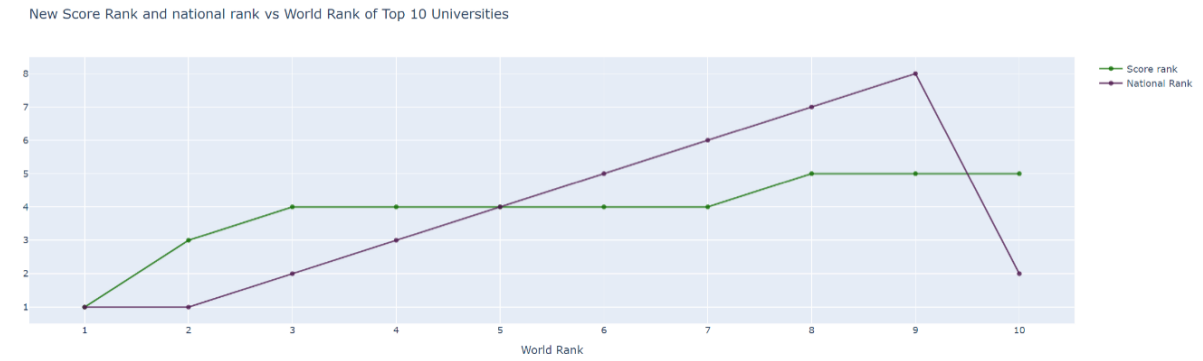
```
import numpy as np

shang_features = ['alumni', 'award', 'hici', 'ns', 'pub', 'pcp']

shang["Score"] = (shang[shang_features].sum(axis=1) / len(shang_features)) * 0.1

# Round the "Score" values to the nearest integer
shang["Score_rank"] = (10 - shang["Score"]).apply(np.ceil).astype(int)

shang["award"] = shang["award"] * 0.1

shang.head(5)
```

	world_rank	university_name	national_rank	total_score	alumni	award	hici	ns	pub	pcp	year	Score	Score_rank
0	1	Harvard University	1	100.0	100.0	10.00	100.0	100.0	100.0	72.4	2005	9.540000	1
1	2	University of Cambridge	1	73.6	99.8	9.34	53.3	56.6	70.9	66.9	2005	7.348333	3
2	3	Stanford University	2	73.4	41.1	7.22	88.5	70.9	72.3	65.0	2005	6.833333	4
3	4	University of California, Berkeley	3	72.8	71.8	7.60	69.4	73.9	72.2	52.7	2005	6.933333	4
4	5	Massachusetts Institute of Technology (MIT)	4	70.1	74.0	8.06	66.7	65.8	64.3	53.0	2005	6.740000	4

New Score Rank and national rank vs World Rank of Top 10 Universities

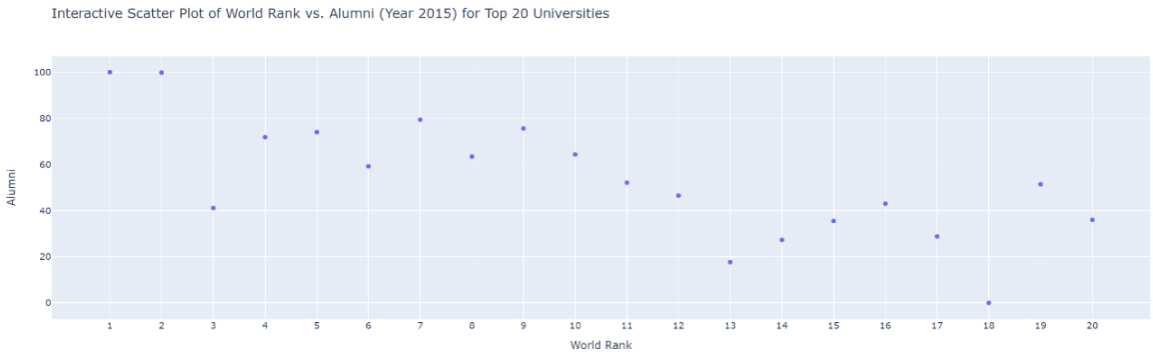
```
import plotly.graph_objs as go
from plotly.offline import iplot

def linelivechart(data, tr_1, tr_2):
    df = data.iloc[:10, :]

    trace1 = go.Scatter(
        x=df[tr_1],
        y=df[tr_2],
        mode="lines+markers",
        name="Score rank",
        marker=dict(color='rgba(16, 112, 2, 0.8)'),
        text=df.index
    )

    trace2 = go.Scatter(
        x=df['world_rank'],
        y=df['national_rank'],
        mode="lines+markers",
        name="National Rank",
        marker=dict(color='rgba(80, 26, 80, 0.8)'),
        text=df.index
    )

    data = [trace1, trace2]
    layout = dict(
        title='New Score Rank and national rank vs World Rank of Top 10 Universities',
        xaxis=dict(title='World Rank', ticklen=5, zeroline=True)
    )

    fig = dict(data=data, layout=layout)
    iplot(fig)
```

```
linelivechart(shang, 'world_rank', 'Score_rank')
```



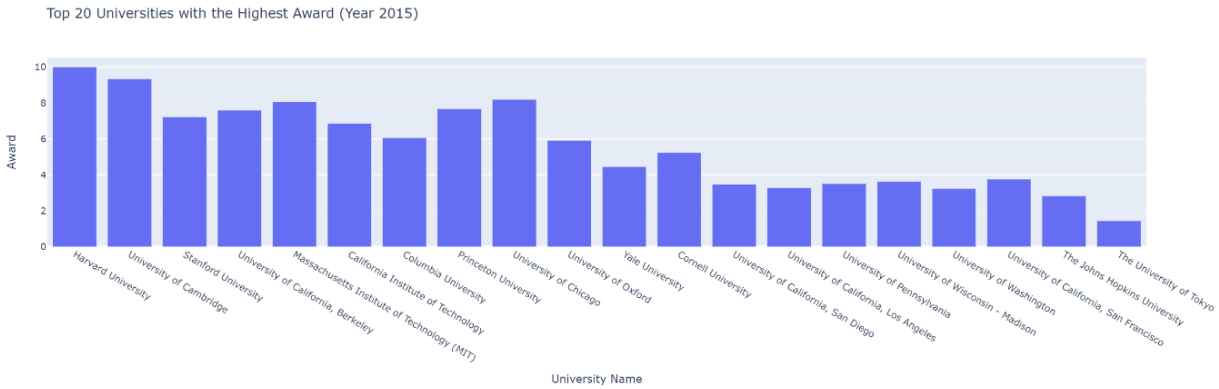
Interactive Scatter Plot of World Rank vs. Alumni (Year 2015) for Top 20 Universities

```
import pandas as pd
import plotly.express as px

data_2015 = shang[shang['year'] == 2015]
top_20_universities = shang.head(20)

fig = px.scatter(
    top_20_universities,
    x='world_rank',
    y='alumni',
    title='Interactive Scatter Plot of World Rank vs. Alumni (Year 2015) for Top 20 Universities',
    labels={'world_rank': 'World Rank', 'alumni': 'Alumni'},
    hover_name=top_20_universities['university_name'],
)

fig.show()
```



Top 20 Universities with the Highest Award (Year 2015)

```
fig = px.bar(
    top_20_universities,
    x='university_name',
    y='award',
    title='Top 20 Universities with the Highest Award (Year 2015)',
    labels={'university_name': 'University Name', 'award': 'Award'},
    hover_name=top_20_universities['university_name'],
)

fig.show()
```



7.3.2.4: Removing data that are not required

```
cwur.drop('institution', axis=1, inplace=True)
```

```
cwur.drop('year', axis=1, inplace=True)
```

```
cwur.drop(['world_rank', 'national_rank'], axis=1, inplace=True)
```

```
cwur.drop('country', axis=1, inplace=True)
```

```
cwur.head(5)
```

	quality_of_education	alumni_employment	quality_of_faculty	publications	influence	citations	patents	score
0	7	9	1	1	1	1	5	100.00
1	9	17	3	12	4	4	1	91.67
2	17	11	5	4	2	2	15	89.50
3	10	24	4	16	16	11	50	86.17
4	2	29	7	37	22	22	18	85.21

7.3.2.5: Splitting data into train and test

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set.

```
x=cwur.iloc[:,: -1]
y=cwur.score
```

```
x.head()
```

	quality_of_education	alumni_employment	quality_of_faculty	publications	influence	citations	patents
0	7	9	1	1	1	1	5
1	9	17	3	12	4	4	1
2	17	11	5	4	2	2	15
3	10	24	4	16	16	11	50
4	2	29	7	37	22	22	18

```
y.head()
```

```
0    100.00
1     91.67
2     89.50
3     86.17
4     85.21
Name: score, dtype: float64
```

```
print(x.shape)
print(y.shape)
```

```
(2200, 7)
(2200,)
```

7.4: Model Building

7.4.1: 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 five regression algorithms. The best model is saved based on its performance.

7.4.1.1: Linear Regression model

```
# 1) Linear Regression
lr = LinearRegression()
lr.fit (x_train, y_train)
y_pred1 = lr.predict(x_test)
```

7.4.1.2: Decision Tree Regression model

```
# 2) Decision Tree Regression
dt = DecisionTreeRegressor()
dt.fit(x_train, y_train)
y_pred2= dt.predict(x_test)
```

7.4.1.3: Support Vector Regression model

```
# 3) Support Vector Regression
SV = SVR()
SV.fit(x_train, y_train)
y_pred3 = SV.predict(x_test)
```

7.4.1.4: Lasso Regression model

```
# 4) Lasso Regression
lassoR = linear_model.Lasso(alpha=0.1)
lassoR.fit(x_train,y_train)
y_pred4= lassoR.predict(x_test)
```

7.4.1.5: Random Forest Regression model

```
# 5) Random Forest Regression
rf = RandomForestRegressor(n_estimators = 100, random_state = 0)
rf.fit(x_train, y_train)
y_pred5 = rf.predict(x_test)
```

```
# Random Forest Regression Hyper Parameter Tuning
from sklearn.model_selection import RandomizedSearchCV

param_dist = {
    'n_estimators': [50, 100, 200],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

rf1 = RandomForestRegressor()

rf_random = RandomizedSearchCV(
    rf, param_distributions=param_dist,
    n_iter=100, cv=5, verbose=2, random_state=42, n_jobs=-1
)

rf_random.fit(x_train, y_train)
y_predrf=rf_random.predict(x_test)
```

7.4.2: Testing the model

Here we have tested with Random Forest algorithm. You can test with all algorithm. With the help of predict() function.

```
y_pred1=rf_random.predict([[7,8,1,2,1,2,6]])
print(y_pred1)
```

```
[99.02637381]
```

7.5: Performance Testing

7.5.1: 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 regression tasks including Mean absolute Error, Mean Squared Error, Root Mean Squared Error and R square Score.

7.5.1.1: Compare the model

Linear Regression model

```
# 1) Linear Regression
lr = LinearRegression()
lr.fit (x_train, y_train)
y_pred1 = lr.predict(x_test)


print ("Prediction Evaluation using Linear Regression" )
print ("MAE: ",mean_absolute_error(y_test, y_pred1))
print ("MSE: ",mean_squared_error(y_test, y_pred1))
print ("RMSE: ", np.sqrt (mean_squared_error(y_test, y_pred1)))
print ("R^2: ",r2_score(y_test, y_pred1))
```

```
Prediction Evaluation using Linear Regression
MAE:  2.6302442785261757
MSE:  25.904438848439774
RMSE:  5.089640345686498
R^2:  0.49002501497866835
```

Decision Tree Regression model

```
# 2) Decision Tree Regression
dt = DecisionTreeRegressor()
dt.fit(x_train, y_train)
y_pred2= dt.predict(x_test)

print ("Prediction Evaluation using Decision Tree Regression" )
print ("MAE: ",mean_absolute_error(y_test, y_pred2))
print ("MSE: ",mean_squared_error(y_test, y_pred2))
print ("RMSE: ", np.sqrt (mean_squared_error(y_test, y_pred2)))
print ("R^2: ",r2_score(y_test, y_pred2))
```

```
Prediction Evaluation using Decision Tree Regression
MAE:  0.7863409090909086
MSE:  2.785336590909091
RMSE:  1.6689327700387129
R^2:  0.9451656916971284
```

Support Vector Regression model

```
# 3) Support Vector Regression
SV = SVR()
SV.fit(x_train, y_train)
y_pred3 = SV.predict(x_test)

print ("Prediction Evaluation using Support Vector Regression" )
print ("MAE: ",mean_absolute_error(y_test, y_pred3))
print ("MSE: ",mean_squared_error(y_test, y_pred3))
print ("RMSE: ", np.sqrt (mean_squared_error(y_test, y_pred3)))
print ("R^2: ",r2_score(y_test, y_pred3))
```

```
Prediction Evaluation using Support Vector Regression
MAE:  1.5571247807848716
MSE:  23.017703434054262
RMSE:  4.7976768788710915
R^2:  0.546855539597443
```

Lasso Regression model

```
# 4) Lasso Regression
lassoR = linear_model.Lasso(alpha=0.1)
lassoR.fit(x_train,y_train)
y_pred4= lassoR.predict(x_test)

print ("Prediction Evaluation using Lasso Regression" )
print ("MAE: ",mean_absolute_error(y_test, y_pred4))
print ("MSE: ",mean_squared_error(y_test, y_pred4))
print ("RMSE: ", np.sqrt (mean_squared_error(y_test, y_pred4)))
print ("R^2: ",r2_score(y_test, y_pred4))
```

```
Prediction Evaluation using Lasso Regression
MAE:  2.6296320443102643
MSE:  25.901489977634245
RMSE:  5.089350644005013
R^2:  0.49008306874905416
```

Random Forest Regression model

```
# 5) Random Forest Regression
rf = RandomForestRegressor(n_estimators = 100, random_state = 0,)
rf.fit(x_train, y_train)
y_pred5 = rf.predict(x_test)

print ("Prediction Evaluation using Random Forest Regression" )
print ("MAE: ",mean_absolute_error(y_test, y_pred5))
print ("MSE: ",mean_squared_error(y_test, y_pred5))
print ("RMSE: ", np.sqrt (mean_squared_error(y_test, y_pred5)))
print ("R^2: ",r2_score(y_test, y_pred5))
```

```
Prediction Evaluation using Random Forest Regression
MAE:  0.5162916450216447
MSE:  1.2760579381148183
RMSE:  1.1296273447977516
R^2:  0.974878528282976
```

```
# Random Forest Regression Hyper Parameter Tuning
from sklearn.model_selection import RandomizedSearchCV

param_dist = {
    'n_estimators': [50, 100, 200],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

rf1 = RandomForestRegressor()

rf_random = RandomizedSearchCV(
    rf, param_distributions=param_dist,
    n_iter=100, cv=5, verbose=2, random_state=42, n_jobs=-1
)

rf_random.fit(x_train, y_train)
y_predrf=rf_random.predict(x_test)

print(rf_random.best_params_)

print ("Prediction Evaluation using Hyper Parameter Tuned Random Forest Regression" )
print ("MAE: ",mean_absolute_error(y_test, y_predrf))
print ("MSE: ",mean_squared_error(y_test, y_predrf))
print ("RMSE: ", np.sqrt (mean_squared_error(y_test, y_predrf)))
print ("R^2: ",r2_score(y_test, y_predrf))
```

```
Fitting 5 folds for each of 100 candidates, totalling 500 fits
{'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'log2', 'max_depth': 20}
Prediction Evaluation using Hyper Parameter Tuned Random Forest Regression
MAE:  0.5293416010155463
MSE:  1.14915668036639
RMSE:  1.0719872575578453
R^2:  0.9773768054083013
```

After calling the function, the results of models are displayed as output. From the five models **random forest** is performing well.

7.6: Model Deployment

7.6.1: Save the best model

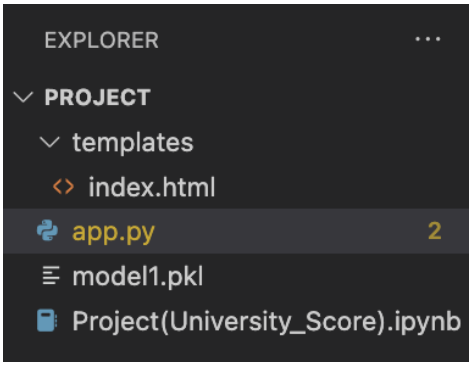
Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it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

```
import pickle
filename = 'model1.pkl'
pickle.dump(rf_random, open(filename, 'wb'))
```

7.6.2: 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



7.6.2.1: Building HTML Pages

For this project create 1 HTML file namely

- index.html

7.6.2.2: Build Python code

Import the libraries

```
from flask import Flask, render_template, request
app=Flask(__name__)
import pickle
import numpy as np
```

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.

```
from flask import Flask, render_template, request
app=Flask(__name__)
import pickle
import numpy as np
model=pickle.load(open('model1.pkl','rb'))
```

Render HTML page:

```
model=pickle.load(open('model1.pkl','rb'))
@app.route('/')
def start():
    return render_template('index.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:

```
def login():
    p=request.form["qe"]
    q=request.form["ae"]
    r=request.form["qf"]
    s=request.form["pb"]
    t=request.form["in"]
    u=request.form["ct"]
    v=request.form["pt"]
    w=[[float(p), float(q), float(r), float(s), float(t), float(u), float(v)]]
    output=model.predict(w)
    print(output)
    return render_template("index.html",y="The predicted university score is "+str(round(output[0],3)))
```

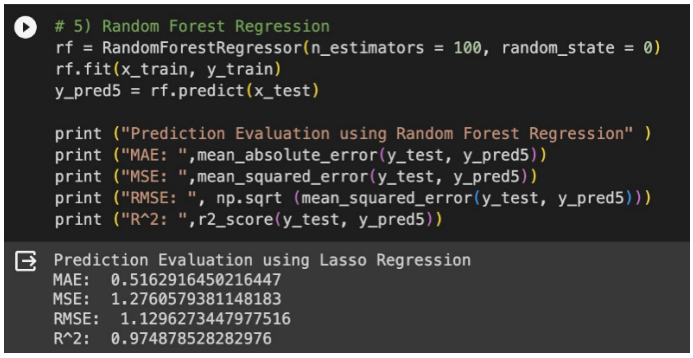
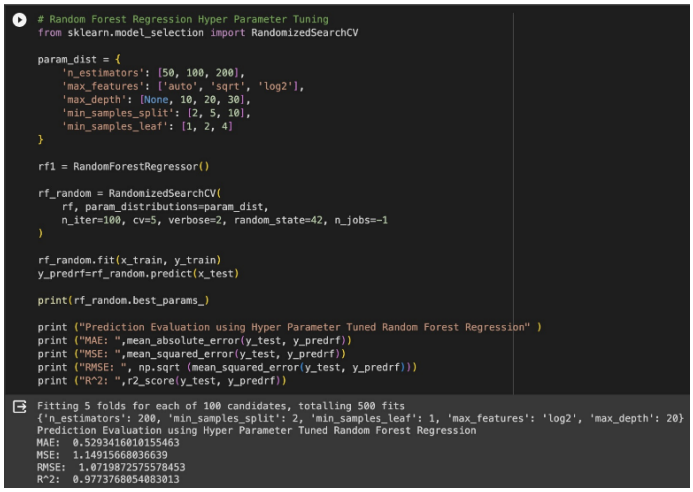
Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text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

Main Function:

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

8.Performance Testing

8.1 Performance Metrics

S.No.	Parameter	Values	Screenshot
1.	Metrics	**Regression Model:** MAE - 0.51629 MSE - 1.27606 RMSE - 1.12963 R2 score – 0.97488	
2.	Tune the Model	**Hyperparameter Tuning:** MAE - 0.52934 MSE - 1.14916 RMSE - 1.07199 R2 score – 0.97738	

9.Results

9.1 Output Screenshots

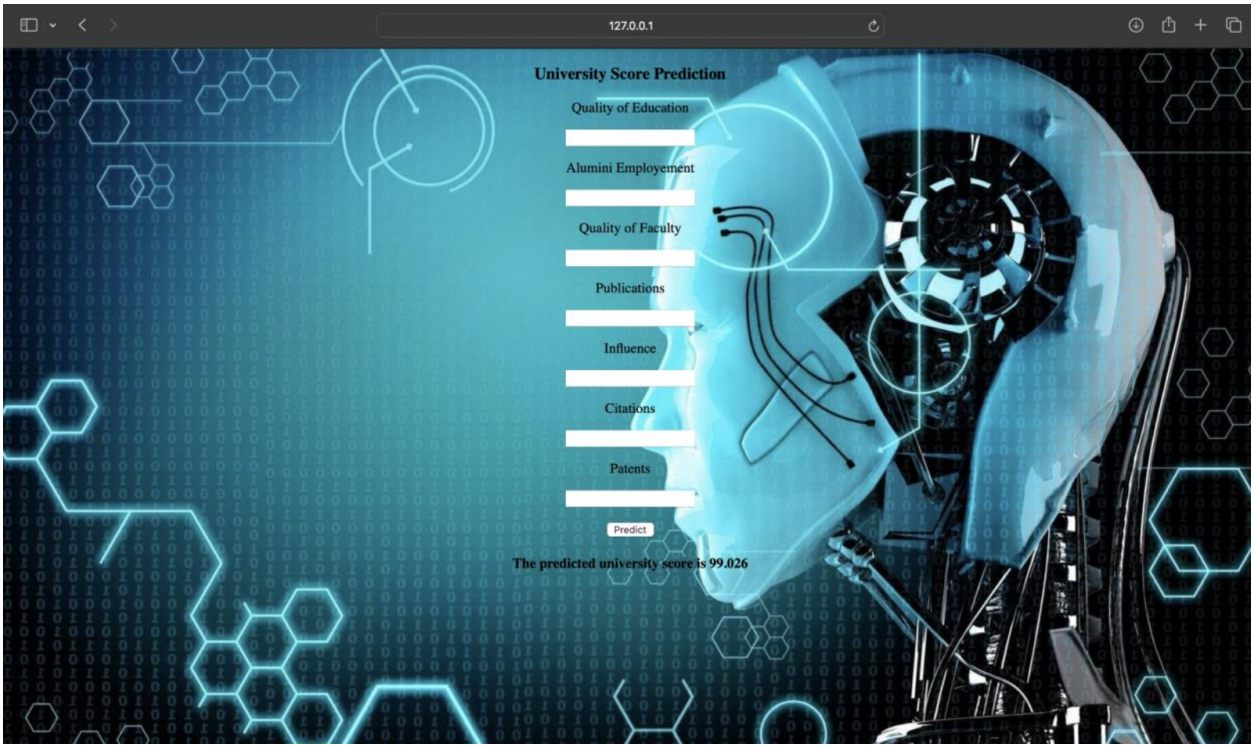
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Enter the inputs, click on the submit button, and see the result/prediction on the web.

```
 * Serving Flask app 'app' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with watchdog (fsevents)
```

```
 * Restarting with watchdog (fsevents)
/opt/anaconda3/lib/python3.8/site-packages/sklearn
eaking code or invalid results. Use at your own ri
  warnings.warn(
/opt/anaconda3/lib/python3.8/site-packages/sklearn
eaking code or invalid results. Use at your own ri
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 323-729-333
```

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


127.0.0.1
University Score Prediction
Quality of Education
7
Alumini Employement
8
Quality of Faculty
1
Publications
2
Influence
1
Citations
2
Patents
6
Predict


127.0.0.1
University Score Prediction
Quality of Education
Alumini Employement
Quality of Faculty
Publications
Influence
Citations
Patents
Predict
The predicted university score is 99.026

10. Advantages and Disadvantages

Advantages:

1. **Informed Decision-Making:** The project empowers prospective students to make informed decisions about their higher education choices, reducing the likelihood of making uninformed choices based on anecdotal information.

2. **Objective Criteria:** By using data-driven models, the project introduces objectivity into the university selection process, reducing the influence of subjective opinions and biases.

3. **Continuous Improvement:** Universities can benefit from the project by receiving feedback based on their scores, allowing them to identify areas for improvement and enhance the quality of their programs.

4. **Funding Allocation:** Predictive scores can impact funding allocation, directing resources to institutions that excel in various aspects, including research, faculty quality, and student outcomes.

5. **Transparency:** The project promotes transparency by providing a clear, quantifiable measure of university performance, aiding stakeholders in their decision-making processes.

Disadvantages:

1. **Data Quality:** The accuracy of predictions heavily relies on the quality and completeness of the data used. Inaccurate or biased data can lead to unreliable results.

2. **Complexity:** Developing a reliable predictive model can be a complex task, requiring expertise in data science, machine learning, and domain knowledge about universities.

3. **Interpretability:** Some machine learning models may lack interpretability, making it difficult for users to understand how predictions are generated, potentially reducing trust in the system.

4. **Bias and Fairness:** If not carefully handled, predictive models can inherit biases present in the training data, potentially perpetuating inequalities in university assessments.

5. **Privacy Concerns:** Collecting and storing user data, even in an anonymized form, can raise privacy concerns. Safeguarding user data is crucial to maintain trust.
6. **User Adoption:** Convincing users to rely on a predictive model for such a critical decision as choosing a university may require a significant effort in building user trust.
7. **Model Maintenance:** Continuous model maintenance and updates are required to ensure the accuracy and relevance of predictions, which can be resource-intensive.
8. **Regulatory Compliance:** Adherence to data protection regulations, such as GDPR, HIPAA, or FERPA, may pose challenges, especially when dealing with user data.

11.Conclusion

The University Score Prediction Project represents a significant step forward in reshaping how students and their families make informed decisions about higher education. By leveraging data-driven predictive models, this project aims to provide an objective and standardized means of assessing universities, addressing the limitations of traditional, subjective methods.

The primary goal of the project is to empower students with the information they need to select universities that align with their goals and aspirations. It offers transparency in university assessments, reduces bias, and promotes accountability among educational institutions. Additionally, universities themselves can benefit by receiving valuable feedback, fostering continuous improvement, and potentially securing more funding based on their demonstrated performance.

However, the project is not without its challenges, including data quality, model complexity, interpretability, and privacy concerns. It requires a balance between the advantages of objective assessment and the need to ensure that predictions are accurate, fair, and respectful of user privacy.

In conclusion, the University Score Prediction Project has the potential to revolutionize the higher education decision-making process. While it faces obstacles, such as model refinement and user trust, its objectives align with the broader goal of enhancing the quality and transparency of education. As the project evolves and matures, it can serve as a valuable tool for students, universities, and policymakers seeking to improve the educational landscape.

12.Future Scope

1. **Enhanced Data Sources:** Incorporating more diverse and comprehensive data sources, including student reviews, internship opportunities, campus facilities, and alumni success stories, can improve the accuracy and completeness of the predictive model.
2. **Geographical Expansion:** Expanding the project to cover universities globally can provide international students with valuable insights into universities worldwide and help institutions in different countries benchmark themselves.
3. **Personalized Recommendations:** Developing a recommendation system that tailors predictions to individual students' preferences and career goals can further assist in university selection.
4. **Real-time Data Integration:** Integrating real-time data, such as faculty achievements, course offerings, and job placement rates, can provide users with the most up-to-date information.
5. **Explanatory AI:** Implementing AI models that not only predict scores but also provide detailed explanations for the predictions can improve user understanding and trust.
6. **Fairness and Bias Mitigation:** Implementing fairness-aware machine learning techniques to ensure that predictions are free from biases and inequalities across different demographic groups.
7. **User Community and Feedback:** Establishing an active user community where individuals can share their experiences and feedback, further enriching the system's insights and trustworthiness.
8. **Integration with Educational Services**: Partnering with educational service providers, such as student loan organizations, scholarship programs, and admission consultants, to offer a holistic educational advisory service.
9. **Education Policy Impact:** Collaborating with education policymakers to use the predictive model to inform funding allocation and accreditation decisions, shaping the future of higher education.
10. **Mobile Applications:** Developing mobile applications for easy access, ensuring that users can make informed decisions on the go.
11. **Research Collaboration:** Collaborating with universities and research institutions to conduct studies on the impact of predictive models on student decision-

making and university improvement.

12. **Multilingual Support:** Expanding language support to cater to a global audience, making the platform accessible to non-English speakers.
13. **Data Governance and Privacy**: Strengthening data governance practices to ensure user data privacy and security, especially in regions with stringent data protection regulations.
14. **Education Gap Analysis:** Offering features that allow users to compare their educational and career goals with the university's strengths and identify areas of alignment or gap.
15. **Integration with Educational Institutions:** Partnering with universities to provide verified, up-to-date data, enhancing the accuracy and relevance of the model.

13.Appendix

Data set was obtained from Kaggle.

Data set link: https://www.kaggle.com/datasets/mylesoneill/world-university-rankings