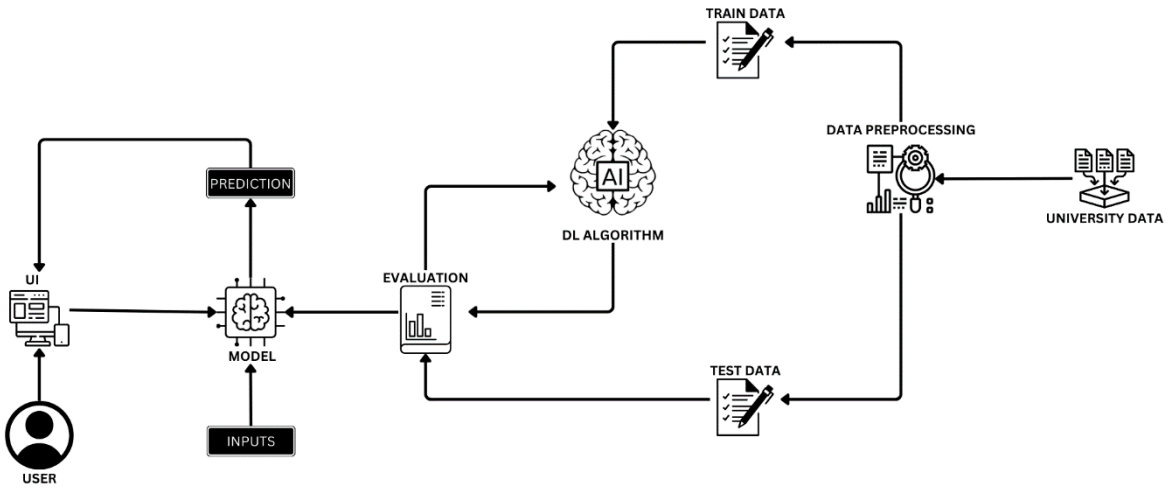
ENVISIONING SUCCESS: Predicting University Scores using Machine Learning

Which universities throughout the world are the best?

Universities receiving scores can be given for a number of significant reasons. First of all, it enables potential students and their families to choose their college of choice with knowledge based on objective standards including faculty caliber, research output, academic reputation, and student results. Second, it gives colleges insightful feedback on their strong and weak points, which can help them pinpoint areas for development and raise the caliber of their work overall. Thirdly, university rankings can affect funding and reputation as well. Generally speaking, universities that rank higher are more likely to attract top talent, receive more research grants, and enjoy better status within the academic world. In general, ranking colleges can make sure that learners obtain a top-notch education and that colleges are held responsible for their actions.

We have certain university characteristics as a dataset in this project. The Score is the dataset's target variable. The quality of education, alumni employment, faculty quality, publications, influence, citations, and patents are the factors that are used to predict this score. You can use this web tool to forecast your university score, which will help you make better decisions regarding your education. The Score Prediction system's primary goal is to forecast the university's score using a set of criteria.

Technical Architecture:



Project Flow:

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

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

- Define Problem / Problem Understanding
    - Specify the business problem
    - Business requirements
    - Literature Survey
    - Social or Business Impact.
- Data Collection & Preparation
    - Collect the dataset
    - Data Preparation
- Exploratory Data Analysis
    - Descriptive statistical
    - Visual Analysis
- Model Building
    - Training the model in multiple algorithms
    - Testing the model
- Performance Testing & 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

Prior Knowledge:

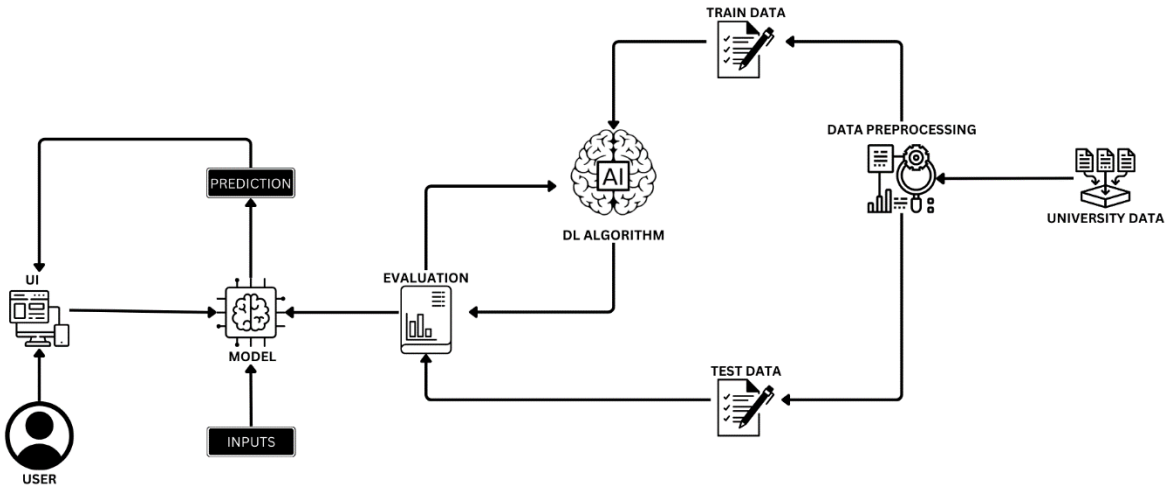
You must have prior knowledge of the following topics to complete this project.

ML Concepts:

- Supervised learning: https://www.javatpoint.com/supervised-machine-learning
- Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
- Regression analysis: https://www.javatpoint.com/regression-analysis-in-machine-learning
- Lasso Regression: https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/
- Support Vector Regression: https://www.analyticsvidhya.com/blog/2020/03/support-vector-regression-tutorial-for-machine-learning/
- Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
- Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
- Xgboost: https://www.analyticsvidhya.com/blog/2018/09/an-end-to-end-guide-to-understand-the-math-behind-xgboost/
- Evaluation metrics: https://www.analyticsvidhya.com/blog/2021/05/know-the-best-evaluation-metrics-for-your-regression-model/

Flask Basics: https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:



Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

Refer Project Description

Activity 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.

Activity 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.

Activity 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.

Milestone 2: Data Collection & Preparation

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

Activity 1: Collect the dataset

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

In this project, we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: https://www.kaggle.com/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.

Activity 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
```

Activity 1.2: Read the Dataset

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

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

```
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%	

Activity 2: Data Preparation

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

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

- Handling missing values
- Handling categorical data
- Handling 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.

Activity 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)
```

Milestone 3: Exploratory Data Analysis

Activity 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(

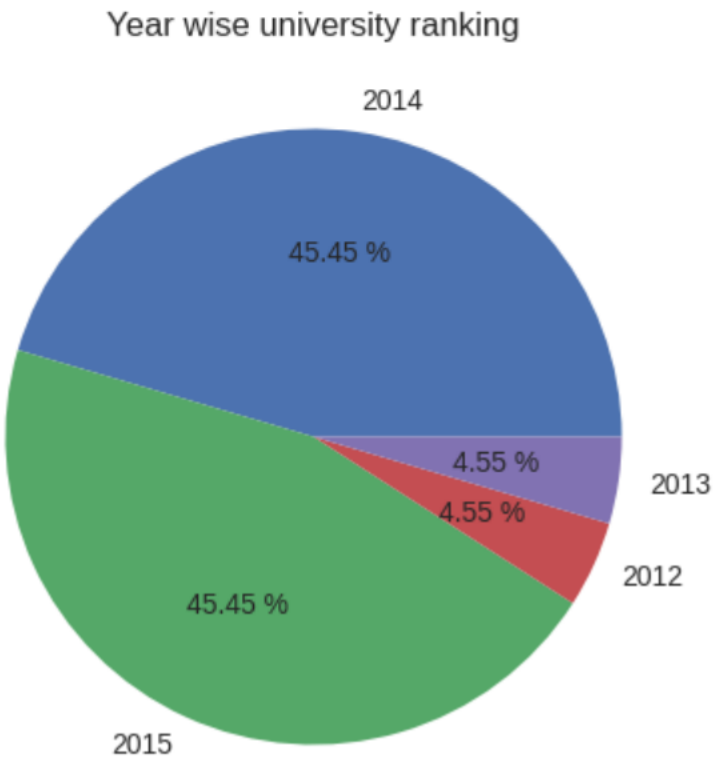
Activity 2: Visual analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions. We will be using seaborn and plotly packages from python to do so

Activity 2.1: Analysing cwurData dataset

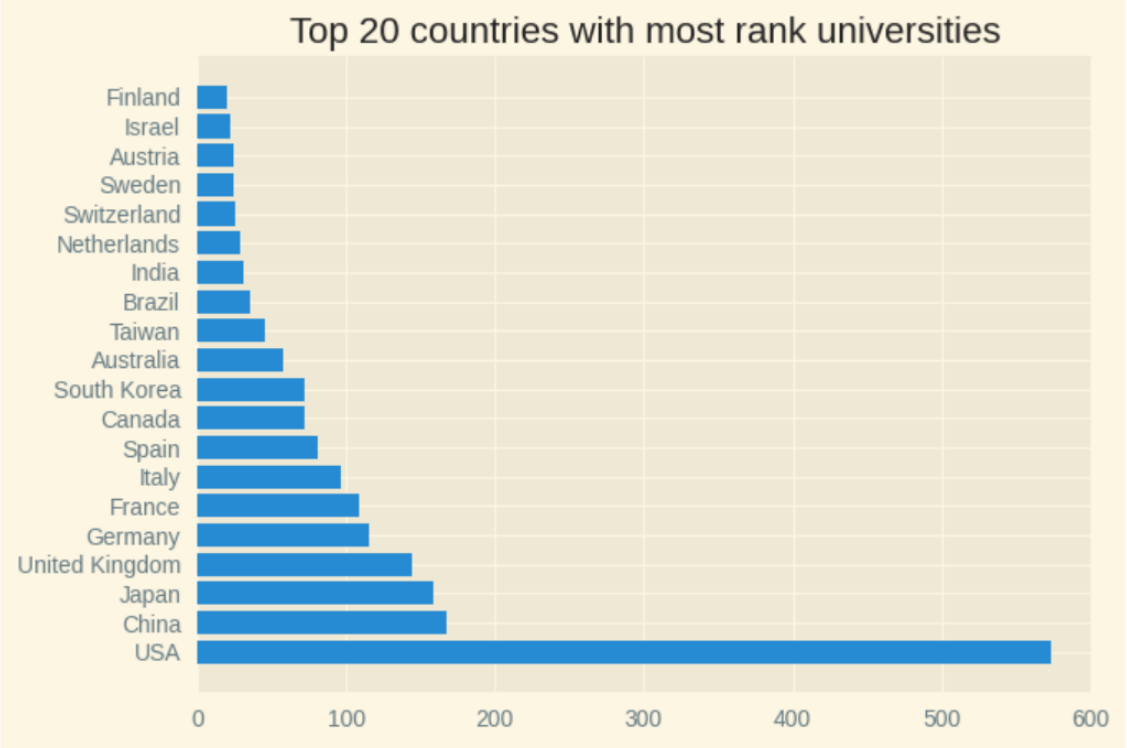
Year wise university ranking

```
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()
```



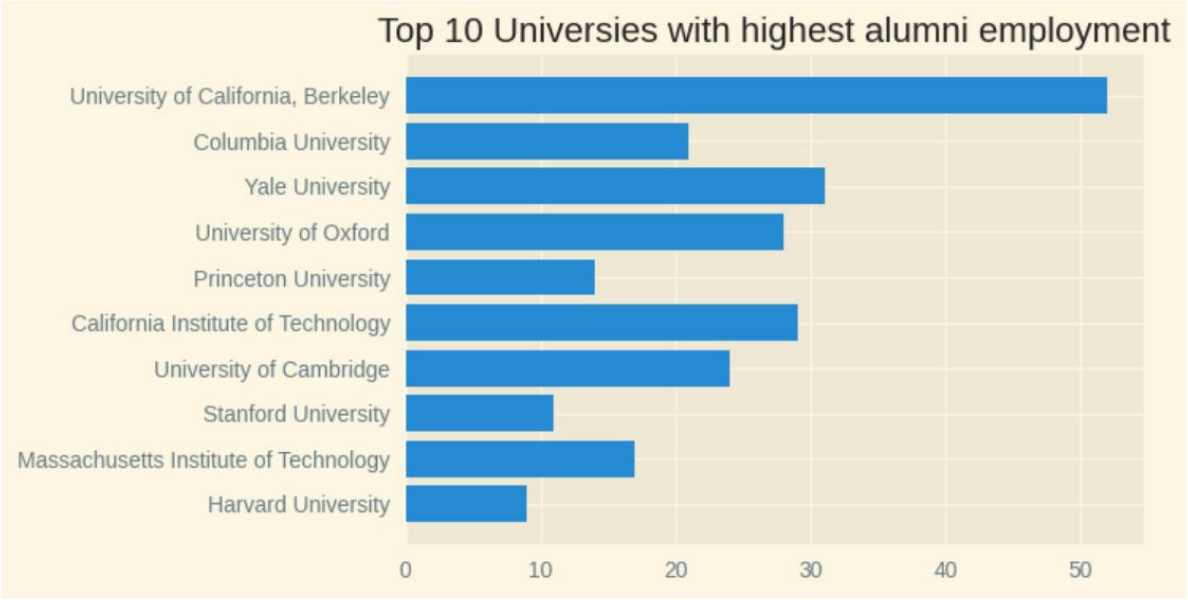
Top 20 countries with most ranked universities

```
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

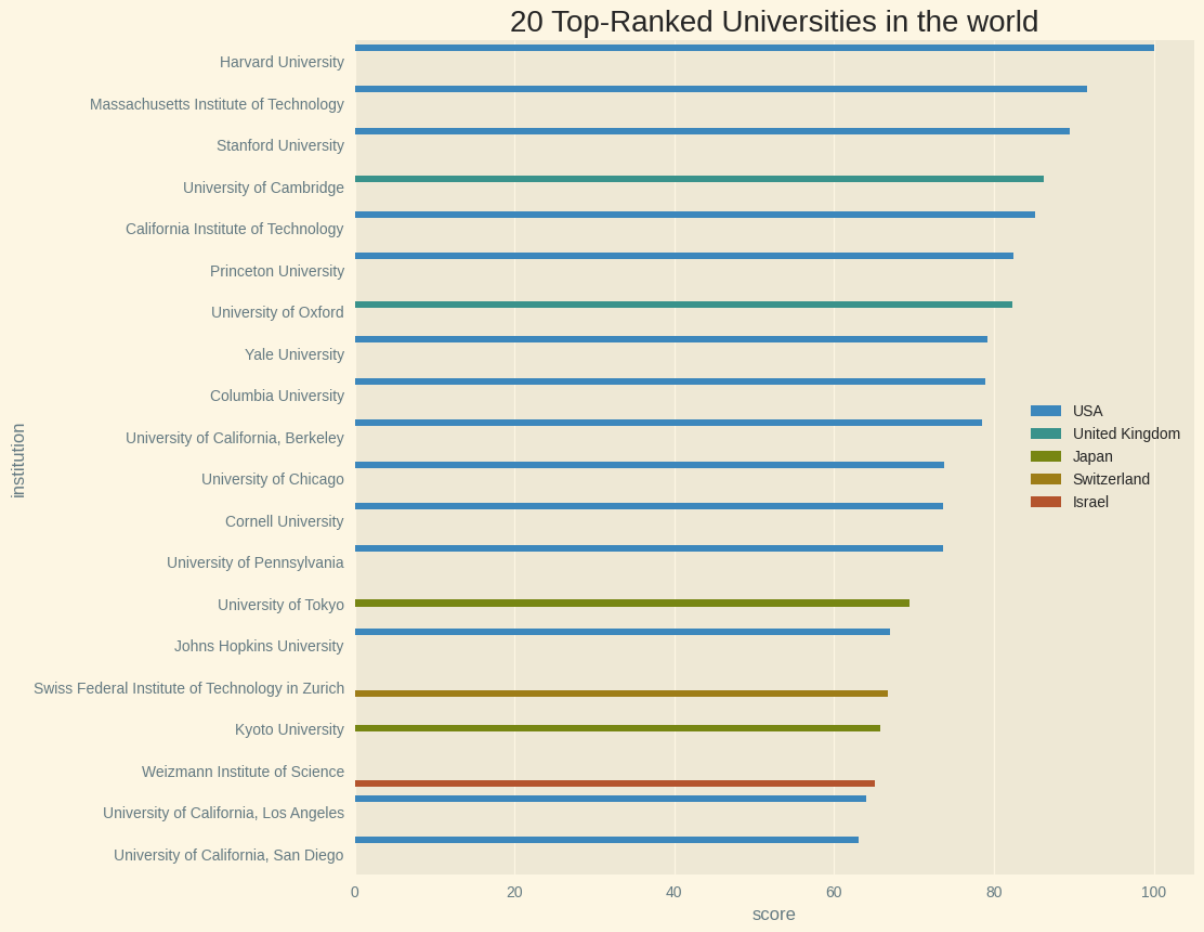
```
top_20 = cwur.head(20)
print('Top 20 universities in the world.')
top_20
```

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

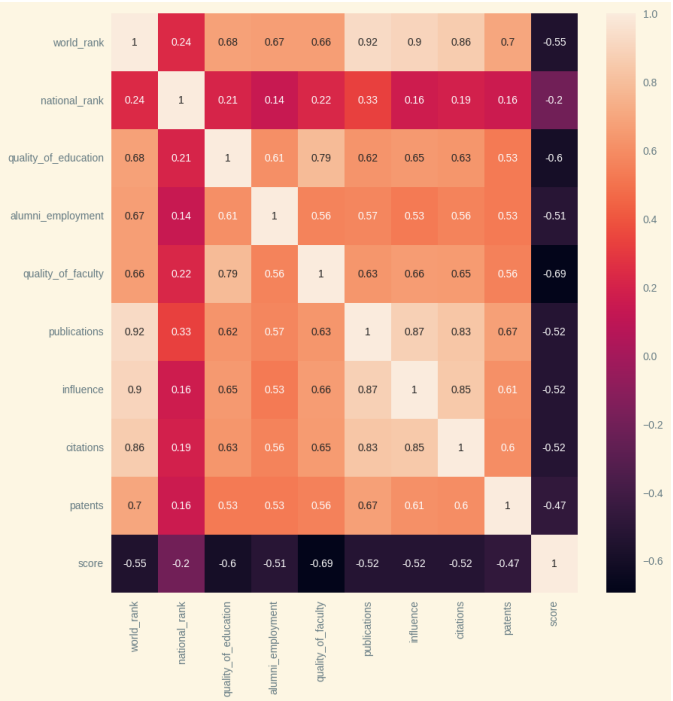
20 Top-Ranked Universities in the world

```
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

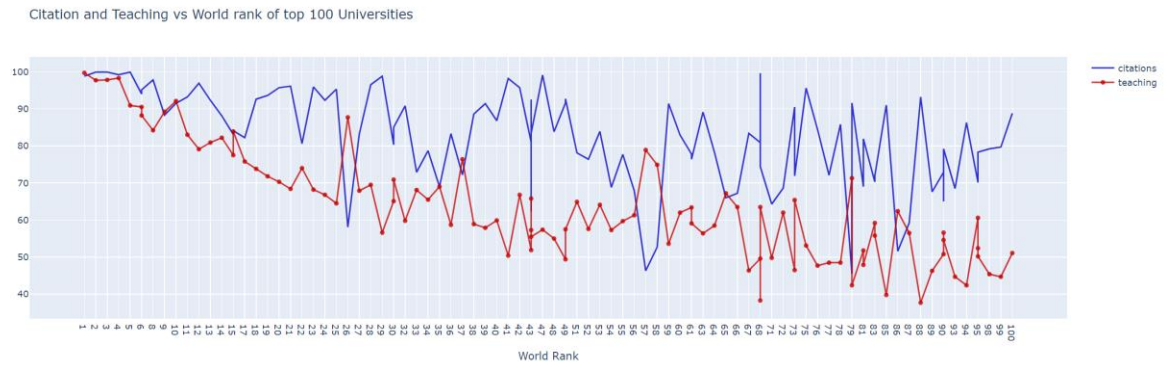
```
plt.figure(figsize=(10,10))
sns.heatmap(cwur.corr(),annot=True)
```



Activity 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)
```



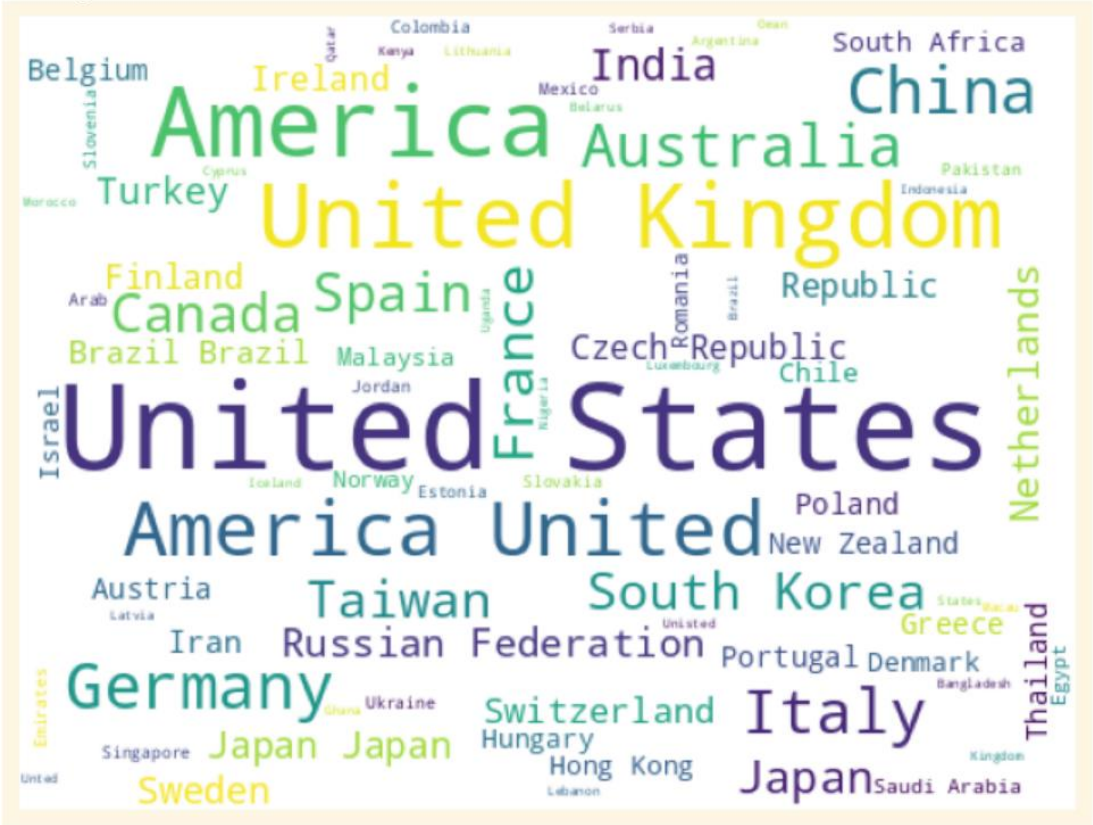
Number of Students in Universities

```
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

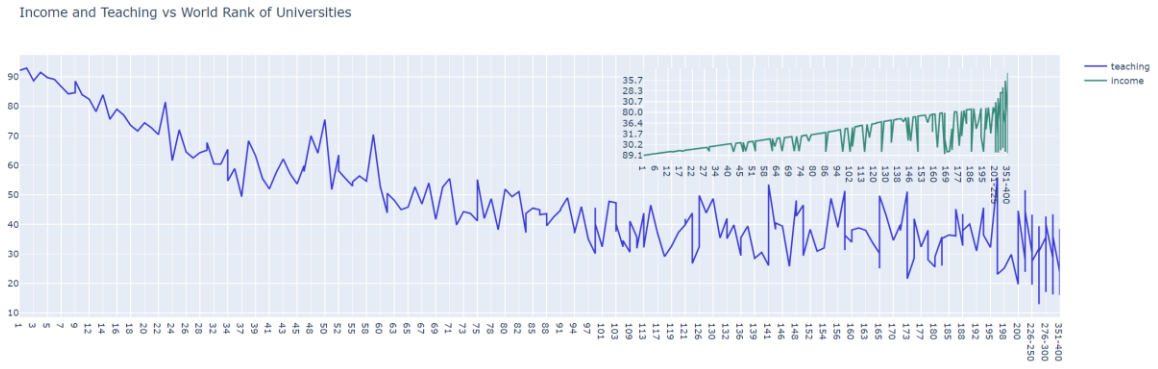
```
from wordcloud import WordCloud

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)
```



Activity 2.3: Analysing Shanghai dataset

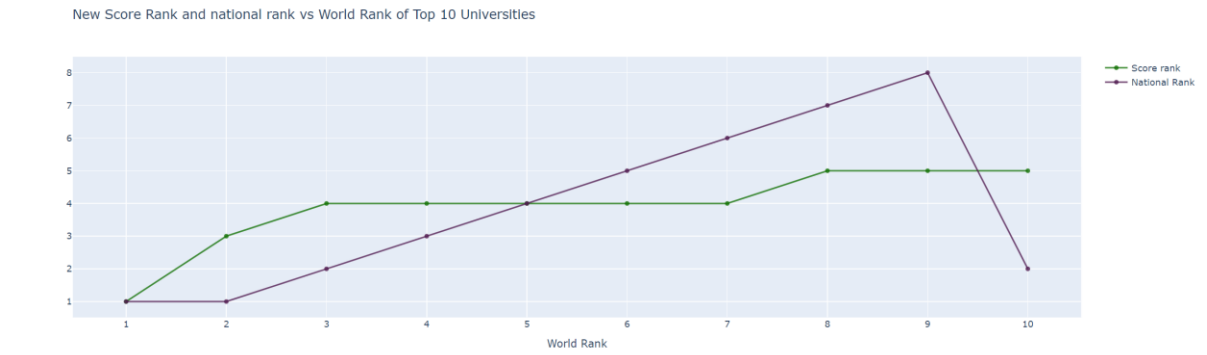
```
import numpy as np

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

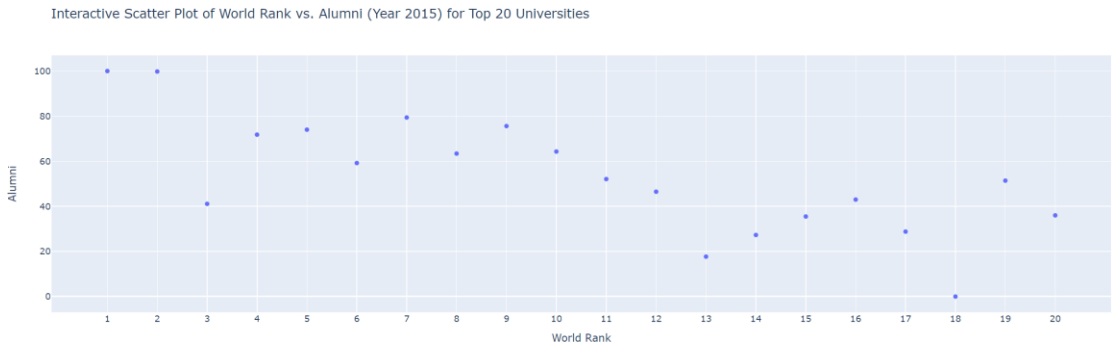
```
import plotly.graph_objs as go
from plotly.offline import iplot

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')
```



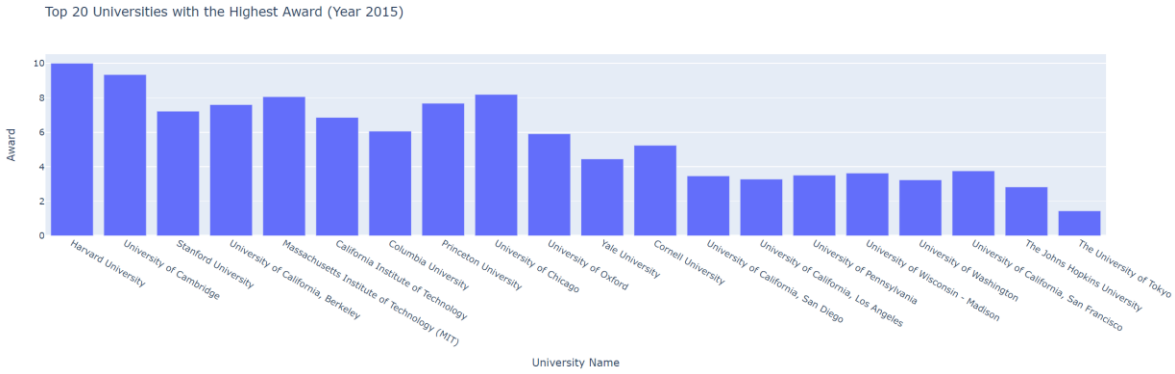
Interactive Scatter Plot of World Rank vs. Alumni (Year 2015) for Top 20 Universities

```
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()
```



Activity 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

Activity 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,)
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,train_size=0.8,random_state=0)
```

Milestone 4: Model Building

Activity 1: Training the model in multiple algorithms

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

Activity 1.1: Linear Regression model

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

Activity 1.2: Decision Tree Regression model

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

Activity 1.3: Support Vector Regression model

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

Activity 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)
```

Activity 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)
```

Activity 2: Testing the model

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

```
# Testing the model
```

```
y_pred=rf.predict([[7,8,1,2,1,2,6]])
print(y_pred)
```

```
[98.83788333]
```

Milestone 5: Performance Testing

Activity 1: Testing model with multiple evaluation metrics

Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for regression tasks including Mean absolute Error, Mean Squared Error, Root Mean Squared Error and R square Score.

Activity 1.1: Compare the model

Linear Regression model

```
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:  2.630244278526174
MSE:  25.90443884843977
RMSE:  5.089640345686497
R^2:  0.49002501497866846
```

Decision Tree Regression model

```
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:  0.8006590909090906
MSE:  2.9050970454545455
RMSE:  1.704434523663067
R^2:  0.9428079939924879
```

Support Vector Regression model

```
print ("Prediction Evaluation using Lasso 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 Lasso Regression
MAE:  1.5571247807848716
MSE:  23.017703434054262
RMSE:  4.7976768788710915
R^2:  0.546855539597443
```

Lasso Regression model

```
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:  25.901489977634238
RMSE:  5.0893506440050125
R^2:  0.4900830687490543
```

Random Forest Regression model

```
print ("Prediction Evaluation using Lasso 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 Lasso Regression
MAE:  0.5162916450216447
MSE:  1.2760579381148183
RMSE:  1.1296273447977516
R^2:  0.974878528282976
```

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

Milestone 6: Model Deployment

Activity 1: Save the best model

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

```
import pickle
filename = 'model.pkl'
pickle.dump(rf, open(filename, 'wb'))
```

Activity 2: Integrate with Web Framework

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

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

Activity 2.1: Building HTML Pages

For this project create 1 HTML file namely

- index.html

Activity 2.2: Build Python code

Import the libraries

```
1  from flask import Flask, render_template, request
2  app=Flask(__name__)
3  import pickle
4  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.

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

Render HTML page:

```
model=pickle.load(open('model.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.

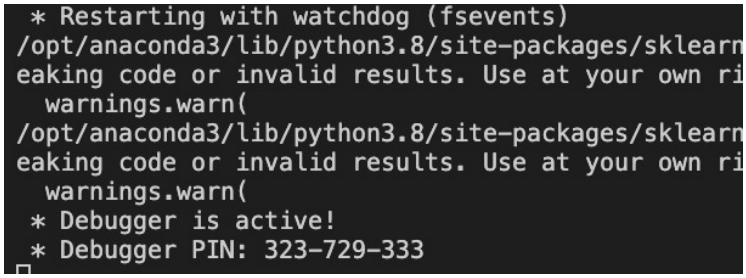
Main Function:

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

Activity 2.3: 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.

```
 * 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)
```



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


University Score Prediction
127.0.0.1:5000
Relaunch to update
University Score Prediction
Quality of Education
Alumini Employement
Quality of Faculty
Publications
Influence
Citations
Patents
Predict


University Score Prediction
127.0.0.1:5000/login
Relaunch to update
University Score Prediction
Quality of Education
Alumini Employement
Quality of Faculty
Publications
Influence
Citations
Patents
Predict
The predicted university score is 98.838

Milestone 7: Project Demonstration & Documentation

Below mentioned deliverables to be submitted along with other deliverables

Activity 1: - Record explanation Video for project end to end solution

Activity 2: - Project Documentation-Step by step project development procedure