Predicting the chances of Admission into foreign universities

By

Himangan Sarma

06.2022

Introduction

In today's competitive market, obtaining a high-quality education from a reputed university holds a lot of importance for students and many students in India as well around the world opt to pursue their graduation or masters in a foreign university. American universities are the most famous in this regard and many students move to the States for pursuing education. However, the application process is quite different from that of India's and takes a lot of factors into play. Moreover, separate applications have to be sent to universities and each applications cost upwards of Rs10,000. So, knowing which universities are worth sending an application to is very important both financially as well as to save time and energy.
So, through the help of artificial intelligence and data obtained from previous applications, my product idea aims at solving this very problem. It will be a prediction system that will read the important details of an application (like GRE score, research experience etc) and provide a percentage value of the possibility of being admitted to a certain university.

Problem Statement

Creating a predictive machine learning model that will take in independent, relevant variables (like test scores and SOP rating) and provide a percentage value output to indicate the possibility of acceptance to a particular foreign university based on data collected from previous applications

Market / Customer base

The primary customers of this product would be the students themselves. The model may be deployed into a web-app and made available to the interested students. Since, the scores/variables that are assessed in American universities are common to most countries in the world, the target consumer base would be students all over the world who are interested in studying in the included universities. Through reliable sources on the internet, I can infer that more than 250,000 students move to America every year either in under-graduate or post-graduate programs across all disciples. Out of them, around 25,000 are Indian and this a very solid local market to begin with. The product might also be licensed out to popular institutions that coach students to get admissions into foreign universities such that the students of the institute use a special code/link to access the same.

Existing products/services

At present scenario, the most students can go for that would provide a similar value to our proposed service are counselling sessions with professionals. These are one-to-one sessions with trained professional who have had years of experience counselling a large number of students. So, with this experience and expertise comes large costs. On average a counselling session and profile evaluation comes at around 2-3 thousand rupees (which is still a low-ball figure). Although a few firms offer a free session, most of them are not specific at all and another paid session is required to get some value out of the free session. Although what we offer would be different from that to a counsellor, in terms of value-for-money our web-app would compete with existing services. If the model is tried and tested and made using proper data sources, the product may even outperform them.

Applicable Patents/Regulations

- Patents on ML algorithms developed
- Data protection and privacy regulations (Customers)
- Protection/ownership regulations
- Ensuring open-source, academic and research community for an audit of Algorithms.
- Meet the legal requirements for hosting a website
- Govt Regulations for small businesses
- Permissions from the respective universities if required

Application Constraints

The preliminary application constraints for hosting the service will be –

- Data: This will be out biggest constraint. Since our model development will be based on previous applications, we have to make sure we a large enough dataset of sample applications to work with. The data collection procedure may involve online sites, approaching previous applicants, coaching institutes or the universities itself. After that data maintenance and continuous data collection will also be needed to further better the model. We have to make use of cloud platforms to store the data gathered over the net.

- Development: Expertise in Python and the various machine learning libraries (disused in the code implementation part) will be required. Also, proper evaluation tools will also be needed in order to make sure the model is ready for deployment. A certain expected accuracy has to be set in order to make the model qualify as deployable.
- Deployment: Although our model will be developed on Python, the deployment will require expertise in web-development both front-end for the interface as well as backend for the storage and maintenance of data and the swift functioning of the web-app.
- Communication: Establishing e-mail service in the product which will send the predicted outcomes to the users on their registered email ids.
- Evaluation & Validation: Thorough understanding of dataset and verification of the results must be performed by people who have expertise in the field.
- Algorithm training and optimization must be done to minimize overfitting of the model and hyperparameter tuning

  Since our product will be a web-based application, there would not be any other costs other than the operating costs for the aforementioned constraints.

Business Opportunity

Substantial amount of web traffic can be foreseen right from the initial time of deployment if proper funnel marketing is done and our target user base come across the site. This is mainly due to the unique value that the product will provide. It presents us with a promising future in product development. If someone doesn't want to spend on career counselling or coaching institutes, our service offers a similar outcome in terms of value. Also, there may also be large EdTech companies that might want to buy the product altogether that will pay out great profits. So, the overall revenue-cost seems positive making the prototype worth investing in.

Concept Development

The product development will take place in phases roughly based on the application constraints discussed above. These are as follows

P1-Data Collection: Just like any other machine learning model, data collection is the first step and gaining the required amount of data from credible sources is essential for the development of an accurate machine learning model.

P2-Data Cleaning/Transformation: Since, we will be using data from different sources, we have to make sure all the acquired data is in the required format compatible to the ML model. Also, cleaning of the data need to be done to take care of missing or duplicate values.

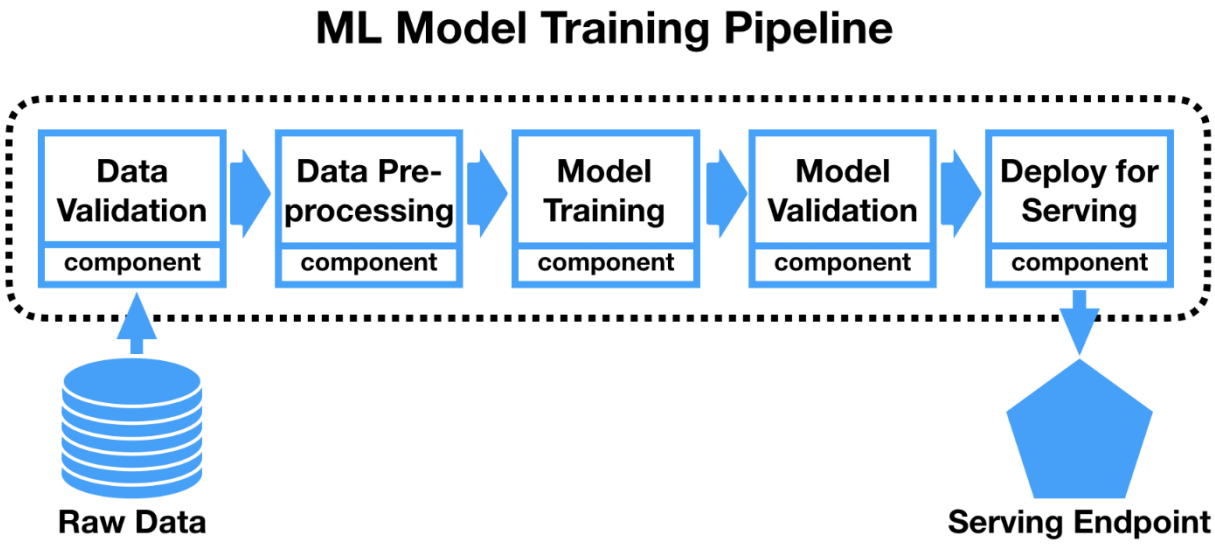
P3- Model Development: Using the obtained cleaned date, this phase will be choosing the appropriate machine learning algorithms and tuning of the necessary parameters to obtain the optimal model that would fit our dataset and objective the best. This might involve testing and trying out several different algorithms and approaches and running evaluation functions on the results obtained by all to choose the best fit.

P4-Model Evaluation: After developing our ML model, the final step would be to test the readiness of the model to be deployed for real use. Evaluation might involve running the model on some test data (with already verified outcome) and compare the result accuracy. This way if we receive a high enough accuracy of prediction (say above 90%) we can be confident in saying the model is ready. We can also do some non-technical forms of evaluation like taking some test application and showing the predictions to a counsellor or someone with experience in university admission process. Although, this might be a cumbersome task, the validation from someone of that stature will only help in ascertaining our product as market-ready.

P5-Model Deployment: Model deployment is simply the engineering task of exposing an ML model to real use i.e., making the model available via real-time APIs. This would require the ML engineers and software developers to work together to ensure the model works reliably in the organization's production environment. This presents a major challenge because there is often a discrepancy between the programming language in which a machine learning model is written and the languages your production system can understand, and re-coding the model can extend the project timeline by weeks or months

Final Product Prototype

Our final product would be a web application that would be easily application. Here the students would provide their scores like GRE, IELTS etc along with other important information like research publications, work experience etc in the form of YES/NO questions or the number of years of experience.
The information will be cleaned and transformed into the necessary format using a data pre-processing engine.
The cleaned data will be input to the ML model and it will provide a percentage value that would be the predicted chance of being accepted into a selected university.
Other than the prediction, our webapp will also provide other helpful information. This could be suggestions for alternative universities or courses or some basic visualizations on where their application stands compared to other applicants (comparing scores etc). This would be helpful to the students in moving forward with their application process.
All of this information will be recorded and sent to the user (through an automated email service) in the form a comprehensive report.
The schematic diagrams with all the above components is given below



Product details

The details regarding the application are as follows:
Working:
1) Whenever a user visits the website and enters in his/her details, the details will be sent to the machine learning module stored in cloud as input. First the input will be pre-processed using a standard data cleaning/pre-processing engine like Spark or we can design our own cleaning algorithm based on the specific needs of the product. This will make sure that the data that is being fed to our ML model is of proper formatting and clean (refer to basics of data cleaning) so that the machine learning algorithms will be of highest efficiency.

2) The ML model will consist of a model training part that would be constantly training our existing model based on new received data and a output part which will take in our provided input (processed in previous step) and generated the prediction.

3) Along with the prediction, there will be other outputs generated as previously stated. This could be visualizations like where does the score rank among other applicants or percentage values regarding how many applicants have work experience. These are just ideas and upon proper deliberation, these additional components can be decided upon based on what are important and we can easily add them to the final output

4) The output of our whole ML model would be a document with all the necessary outputs. The final step would be generating the outputs on the webpage as the result and sending out a copy of the result to the user through the automated email service.

5) The records generated upon each use is also to be stored in a database as it will help in improving the product in the long run (both for evaluation as well as for training the model)

Software/Frameworks

- For Model building: Python
  (Libraries –NumPy, pandas, matplotlib, sklearn among others based on the type of algorithms to be developed)
- For data processing: Spark engine
- For model deployment: Flask framework
- For database: MongoDB, SQL

For model building, all the necessary tools and algorithms have been discussed in the next section

Small Scale Implementation

Code Link:
https://github.com/himangan/Feynn/blob/f8385ba5507fddf6efb08833740bf6144a114c61/T0_notebook.ipynb

I have performed a small-scale implementation of the idea using a dataset that I found on Kaggle. It is a 500-record dataset that has predictions of chances of admission at masters' courses in UCLA.
The dataset contains several parameters which are considered important during the application for Masters Programs.
The parameters included are:

GRE Scores (out of 340)
TOEFL Scores (out of 120)
University Rating (out of 5)
Statement of Purpose and Letter of Recommendation Strength (out of 5)
Undergraduate GPA (out of 10)
Research Experience (either 0 or 1)
Chance of Admit (ranging from 0 to 1)

The dataset is owned by Mohan S Acharya. Link - Graduate Admission 2 | Kaggle

I have discussed my approach to exploring the dataset and creating a suitable Machine Learning model that would meet the requirements of our proposed product. This is only a small-scale representative implementation and for deployment a lot of work has to be done in terms of data collection, selection and tuning of model, evaluation etc. I only aim at demonstrating how one can use data to produce predictions and other valuable insights.

1) Data Exploration

After loading up the dataset, we look at the basic insights like shape, names of columns, percentile, mean and std of different numerical values of the variables, information regarding the datatype, presence of null values etc. These values may be helpful to us when building the report of the user that I discussed previously. For instance, using mean or top 75% value we can assess an applicant's standing based on a variable like GRE score.

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Serial No.	500.0	250.50000	144.481833	1.00	125.7500	250.50	375.25	500.00
GRE Score	500.0	316.47200	11.295148	290.00	308.0000	317.00	325.00	340.00
TOEFL Score	500.0	107.19200	6.081868	92.00	103.0000	107.00	112.00	120.00
University Rating	500.0	3.11400	1.143512	1.00	2.0000	3.00	4.00	5.00
SOP	500.0	3.37400	0.991004	1.00	2.5000	3.50	4.00	5.00
LOR	500.0	3.48400	0.925450	1.00	3.0000	3.50	4.00	5.00
CGPA	500.0	8.57644	0.604813	6.80	8.1275	8.56	9.04	9.92
Research	500.0	0.56000	0.496884	0.00	0.0000	1.00	1.00	1.00
Chance of Admit	500.0	0.72174	0.141140	0.34	0.6300	0.72	0.82	0.97

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         500 non-null    int64  
 1   GRE Score          500 non-null    int64  
 2   TOEFL Score        500 non-null    int64  
 3   University Rating  500 non-null    int64  
 4   SOP                500 non-null    float64
 5   LOR                500 non-null    float64
 6   CGPA               500 non-null    float64
 7   Research           500 non-null    int64  
 8   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
```

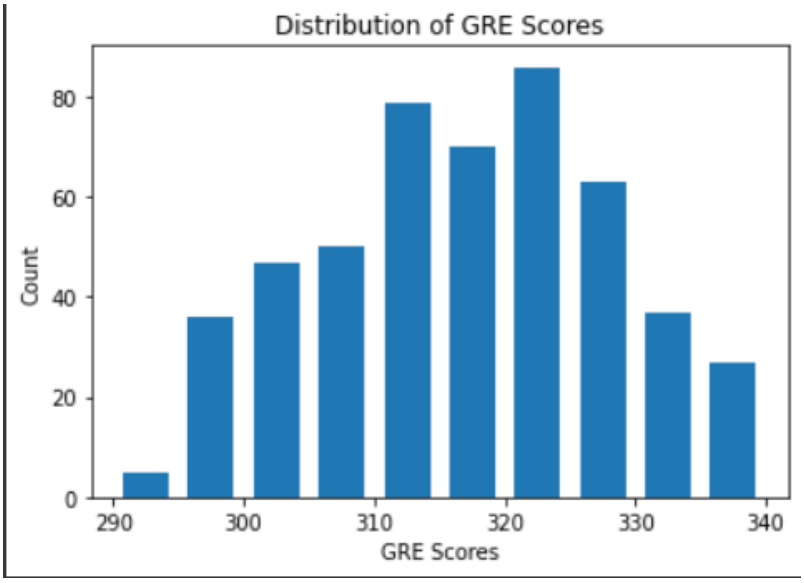
Renaming columns to simpler words

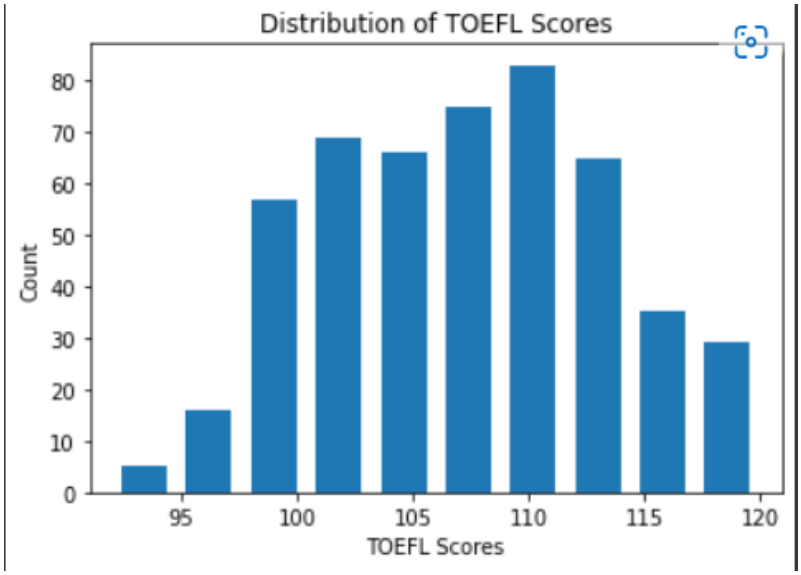
```
df = df.rename(columns={'GRE Score': 'GRE', 'TOEFL Score': 'TOEFL', 'LOR ': 'LOR', 'Chance of Admit '
df.head()
```

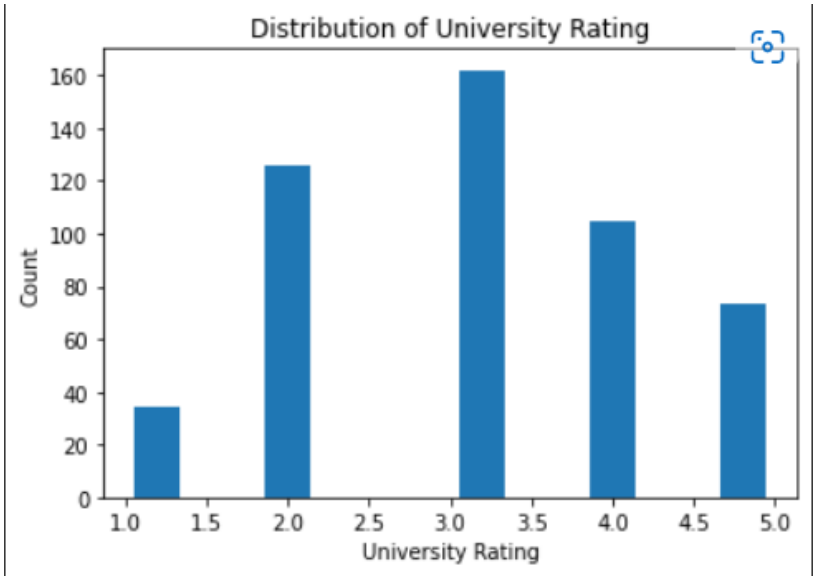
	Serial No.	GRE	TOEFL	University Rating	SOP	LOR	CGPA	Research	Probability
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

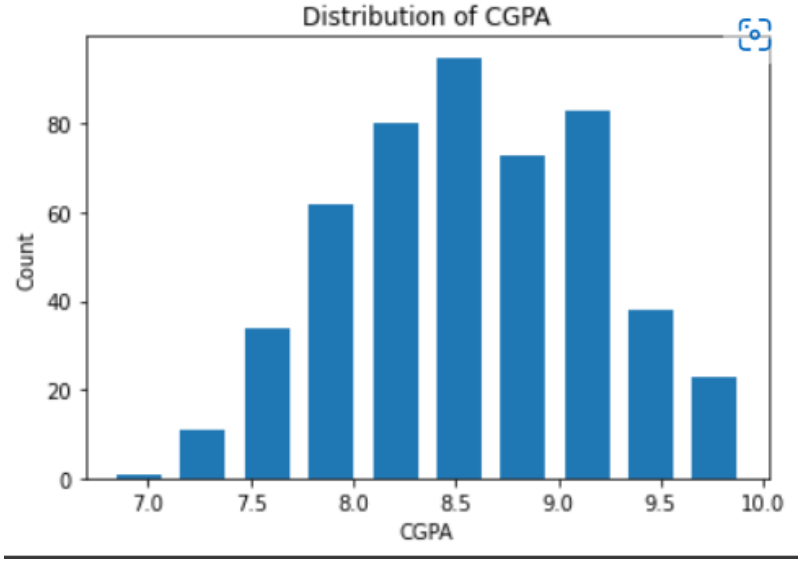
2)Data Visualizations

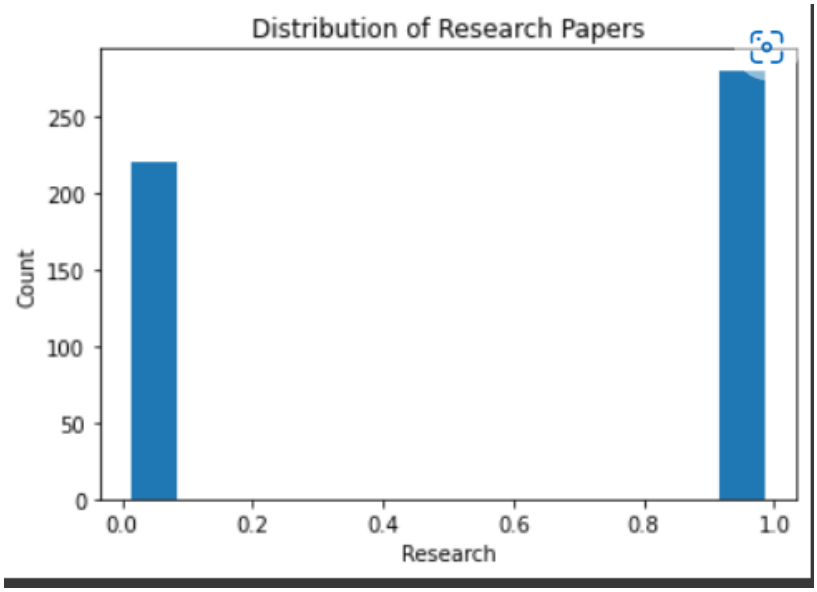
We use matplotlib to generate histograms plotting the value of the variables against the number of records in that region. We do this for all numerical variables to get a better understanding of how the values are distributed amongst all records. We could directly use these visualizations (after making them prettier) in our reports as they will be helpful to the users.





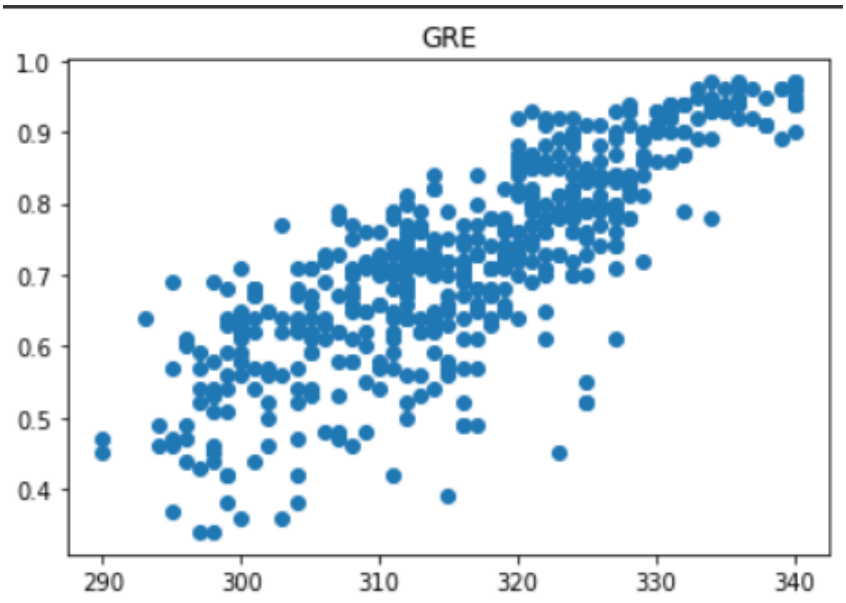

Distribution of University Rating
Count
University Rating
0
20
40
60
80
100
120
140
160
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


Distribution of CGPA
Count
CGPA
0
20
40
60
80
7.0
7.5
8.0
8.5
9.0
9.5
10.0

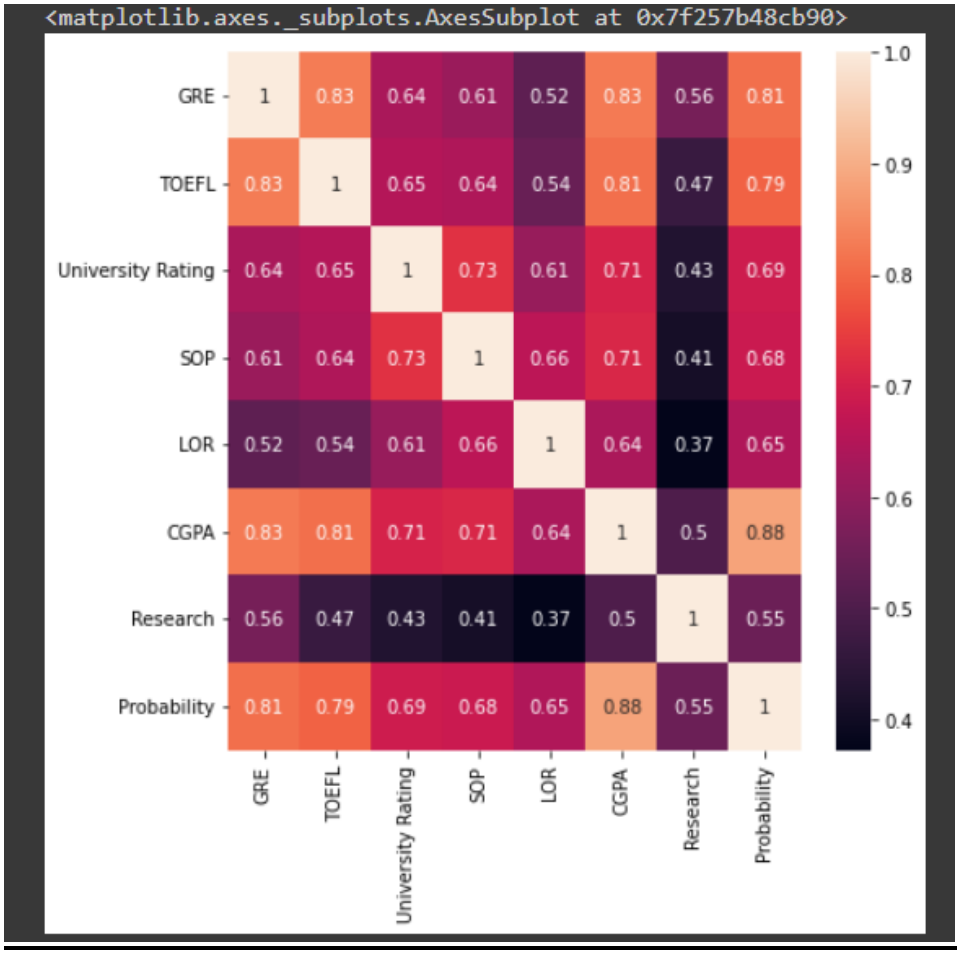

Distribution of Research Papers
Count
Research
0
50
100
150
200
250
0.0
0.2
0.4
0.6
0.8
1.0

I also plotted a Scatter plot to understand the relationship between the Variables field and the target variable (i.e., chance of admission). It will basically describe how high your chances (how close the y axis is to 1) for all values in the x-axis. An example scatter plot for GRE score is shown below

```
def plot_scatter(Data,list1):
    for ri, rd in Data.iteritems():
        plt.scatter(rd,list1)
        plt.title(ri)
        plt.show()
y=df['Probability']
plot_scatter(df,y)
```



Another interesting plot is the correlation plot which will just give us information regarding how all the variables are correlated to each other. A cell having value closer to 1.0 means the 2 variables directly across the cell are highly correlated. The correlation matrix obtained using seaborn heatmap is shown below



From the correlation heatmap we can infer that CGPA with a score of 0.88 has a high correlation with the probability of getting admitted while research although having a score of 0.55 has the least correlation out of the given features. Similarly, we can also see the correlation of the variables with each other like GRE and TOEFL scores are highly correlated while SOP has low correlation with LOR

Model Building

For model building purposes I will first divide the dataset into the features (the scores) and outcome (probability). Also, for evaluation purposes I will train the model on the first 400 records and test it on the last 100 records. This way I will evaluate the model by comparing the outcome of the model with the actual outcome in the dataset.

```
[ ] # Splitting the dataset in features and outcome
    X = df_copy.drop('Probability', axis='columns')
    y = df_copy['Probability']
```

```
[24] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=None)
     y_test=list(y_test)

[28] len(X_train)

     400

[29] len(X_test)

     100
```

Now for the main model, I have decided to test out a few regression models using different types of regression. We could also use Neural networks to perform the task but since this is just a sample implementation, I have only gone with the regression models.

I have evaluated the model using the R2 score. R2 score or *Coefficient of Determination* is the amount of the variation in the output dependent attribute which is predictable from the input independent variable (s). A higher value of r2 means the actual values of y will be close to the regression line, and similarly, if it is nearing 0, the values move away from the regression line.

Random Forest Regession

```
[30] from sklearn.ensemble import RandomForestRegressor

[31] model=RandomForestRegressor(n_estimators=800,random_state=100)
     model.fit(X_train,y_train)
     y_pred_random=list(model.predict(X_test))

[39] from sklearn.metrics import r2_score
     print("R2 score for Random Forest Regression")
     r2_score(y_test, y_pred_random)

     R2 score for Random Forest Regression
     0.7736629102203271
```

Linear Regression

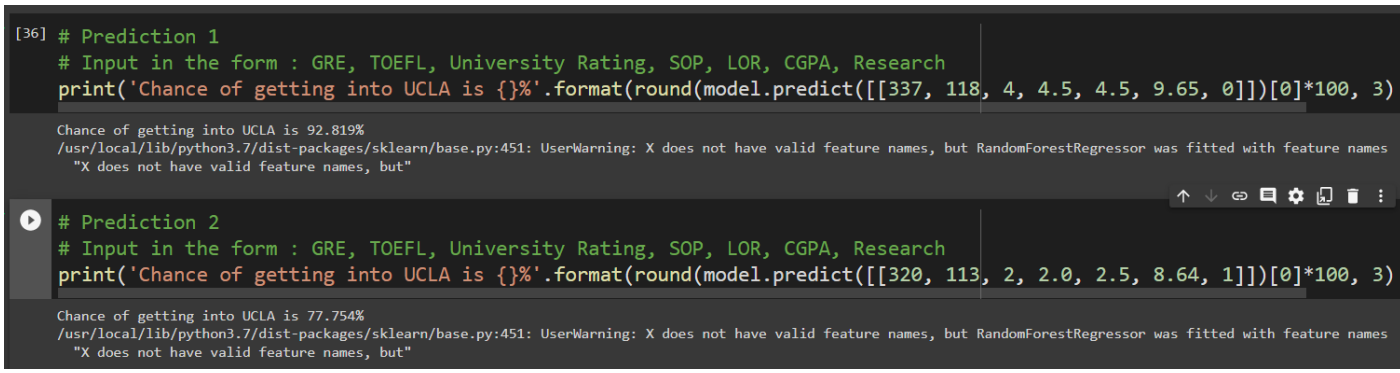
```
[37] from sklearn.linear_model import LinearRegression

[38] linearregression = LinearRegression(normalize=True)
     linearregression.fit(X_test, y_test)
     y_pred_linearregression = list(linearregression.predict(X_test))

[42] print("R2 score for Linear Regression")
     r2_score(y_test, y_pred_linearregression)

     R2 score for Linear Regression
     0.8277725181152369
```

```
Decesion Tree Regression

[44] from sklearn.tree import DecisionTreeRegressor

[45] tr=DecisionTreeRegressor()
     tr.fit(X_train,y_train)
     y_pred_decision=list(tr.predict(X_test))

[46] print("R2 score for Decesion Tree Regression")
     r2_score(y_test, y_pred_decision)

     R2 score for Decesion Tree Regression
     0.46646490604433954
```

So, from the above results we can say that Linear regression is the better regression model that can be implemented for developing the final model. This can be attributed to the fact that there is a monotonic relationship between our features and outcome (chance of admission is always greater if you have a higher score or research experience).
However, with proper tuning, scaling of data or using other machine learning algorithms (like NN) we can surely increase the r2 score to above 90%.

Output

The input to our model has to be in the form of a 2-D array having the values of all the independent variables/features in the correct sequence. The formation of the 2-D array will be taken care of by our data re-processing engine. Then it will give us a prediction in terms of a int value.
A sample output generated by given input is shown below.



Conclusion

So, this was my idea for a ML driven product that could be used students and by education companies alike. Other than providing prediction chances of being admitted the idea of this model can also be used for profile evaluation and profile comparison purposes. Overall, it has the potential to be a useful product if developed and marketed right. Since i do not have any experience in web development i could not elaborate on that side of the product greatly but with the right expertise, a prototype can be up and running without much time and money invested. Data collection for proper training still remains a constraint that will need quite some time and work to make the product perfect.