SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DATA VISUALIZATION

CSE - 3020

PROJECT REVIEW 3

Prof. ANNAPURNA JONNALAGADDA

Slot: E2

COLLEGE ADMISSIONS ANALYSIS FROM AN INDIAN PERSPECTIVE

SUBMITTED BY:

Manan Kalpesh Shah – 18BCE2097

Akhil Thomas – 18BCE2041

School of Information Technology
VIT University
Vellore Tamil Nadu - 632 014

1.Problem Statement

We will visualize the different factors required for a student to get admission in US colleges which will make it very easy for a person with or without any knowledge to understand the different variations and factors needed. We will also be building a model which will predict if the examinee is eligible and has scored enough marks to get a college in the US for masters using the previous year's scores given by other examinees. We will be building a linear regression model to predict the chance of admission.

2.Introduction

2.1 MOTIVATION

As a student the pressure for knowing where you stand while applying for higher education at various colleges is very high. The classifier that we plan to make is made on one of the reasons where one can check their likeliness in getting their college or not, given they give the required inputs needed so that the ML can predict. The dataset that we used has all the fields that help in training the model to a most likely a 97% accuracy. The predicted output gives them a fair idea about their chances for a particular university.

2.2 SIGNIFICANCE

There are many factors that influence admission decisions. And while colleges rely on more than quantitative data to make admissions decisions, quantitative data can show us in a concrete way many things that qualitative data cannot. Even though it's tough to understand and estimate how these factors are truly judged and filtered by colleges, we do know that some of the factors such as CGPA and GRE scores can weigh heavily on determining acceptance. Metrics such as these scores can be leveraged in form of data and be analysed to gain insight into admission trends and can help students in shortlisting universities with their profiles saving time, effort and money that goes into the exhaustive application process. The predicted output can also give them a fair idea about their chances for admission to a particular university

2.3 SCOPE AND APPLICATION

With a 97% accuracy in our model, this project can be worked on and improved to be made into a GUI where different types of users can make use of it. For example, college administration who have the roles of filtering students with regarding their application, that job can be easily done with just by giving the inputs to this model and cross verifying the outputs manually for assurance which saves a lot of time and energy. The other sets of users are the students themselves as they can see where they stand in the current competition and how much they should improve in the

specific feature (example GRE marks should increase) and then work accordingly to their goal

3. Literature Survey

There have been several project and studies performed on topics related to students admission into universities. Multiple machine learning models have been used to create a system that would help the students to shortlist the universities suitable for them. A second model was created to help the colleges to decide on enrolment of the student. Nave Bayes algorithm was used to predict the likelihood of success of an application, and multiple classification algorithms like Decision Tree, Random Forest, Nave Bayes and SVM were compared and evaluated based on their accuracy to select the best candidates for the college. Limitation of this research as that it did only relied on the GRE, TOEFL and Undergraduate Score of the student and missed on taking into consideration other important factors like SOP and LOR documents quality, past work experience, technical papers of the students etc.

[1] Bayesian Networks were used by (Thi et al. (2007)) to create a decision support system for evaluating the application submitted by international students in the university. This model was designed to predict the performance of the aspiring students by comparing them with the performance of students currently studying in the university and had similar profile during their application. In this way based on the current students profile the model predicted whether the aspiring student should be granted admission to the university. Since the comparisons were made only with the students who were already admitted in the university and the data of the students who were denied admission were not included in the research this model proved to be less efficient due to the problem of class imbalance.

[2] (Abdul Fatah S; M (2012)) developed a model that can provide the list of universities/colleges where the which best suitable for a student based on their academic records and college admission criteria. The model was developed by applying data mining techniques and knowledge discovery rules to the already existing in-house admission prediction system of the university.

[3] (Mishra and Sahoo (2016))conducted a research from a university point of view to predict the likelihood of a student enrolling in the university after the have enquired about of courses in the university. They used K-Means algorithm for clustering the students based on different factors like feedback, family income, family occupation, parents qualification, motivation etc. to predict if the student will enroll at the university or not. Depending upon the similarity of the attributes among the students they were grouped into clusters and decisions were made. The objective of the model was to increase the enrolment of the students in the university.

[4] (Eberle et al. (n.d.)) used machine learning and predictive modelling to develop a model that to evaluate the admission policies and standards in the Tennessee Tech University. A well know version of the C4.5 algorithm, J48 was used to create the model. Like the models mentioned above they used the different factors of the student profile to evaluate the chances of their admission in the university. The model worked well in predicting the true positive scenarios where the student was had good profile to secure the admission, but it failed in efficiently identifying the true negatives because of which student who does not satisfy the defined criteria In research conducted by (Jamison (2017)) the yield of college admission was predicted using machine learning techniques. Yield rate can be defined as the rate at which the students who have been granted admission by the university actually enrol for the course.

[5] Multiple machine learning algorithms like Random Forest, Logistic Regression and SVM were used to create the model; the models were compared based on their performance and accuracy, Random Forest outperformed the other models with 86% accuracy and was thus used to create the system. The factors that proved to be significant in predicting successful application were also highlighted.

[6] GRADE system was developed by (Waters and Miikkulainen (2013)) to support the admission process for the graduate students in the University of Texas Austin Department of Computer Science. The main objective of the project was to develop a system that can help the admission committee of the university to take better and faster decisions. Logistic regression and SVM were used to create the model, both models performed equally well and the final system was developed using Logistic regression due to its simplicity. The time required by the admission committee to review the applications was reduced by 74% but human intervention was required to make the final decision on status if the application.(Nandeshwar et al. (2014)) created a similar model to predict the enrolment of the student in the university based on the factors like SAT score, GPA score, residency race etc. The Model was created using the Multiple Logistic regression algorithm, it was able to achieve accuracy rate of 67% only.

4. Implementation

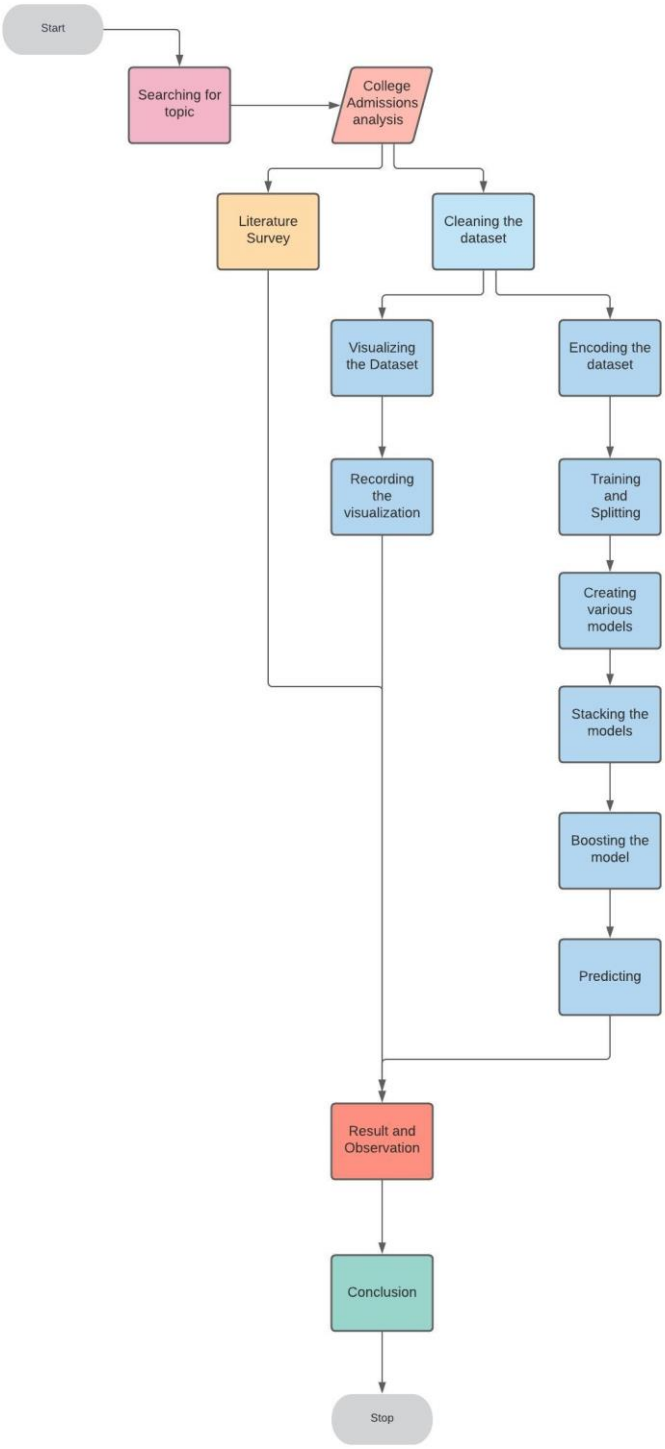
4.1 FRAMEWORK

We plan on making relevant visualizations by using tools such as Plotly, Seaborn so that when a user sees the he should have an idea of the whole dataset by just looking at the visualizations. A few examples of the charts we used are Bar, Hist, Pie, Join Plot, Box etc. For example, some of the visualizations we have done are

1. Find the most common range scores of GRE, TOFFEL, CGPA
2. The relations between every column to each other
3. The major factors affecting the chance of admit

For the Model we are planning on making multiple models and see which gives us a better accuracy

1. Linear Regression: Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable.
2. Support Vector Machine: As in classification, support vector regression (SVR) is characterized by the use of kernels, sparse solution, and VC control of the margin and the number of support vectors. Although less popular than SVM, SVR has been proven to be an effective tool in real-value function estimation.
3. Random forests: random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting
4. Extra Trees Regressor: This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.



4.2 ALGORITHM

i) Linear Regression:

Linear regression is used to predict a quantitative response Y from the predictor variable X. Mathematically, we can write a linear regression equation as:

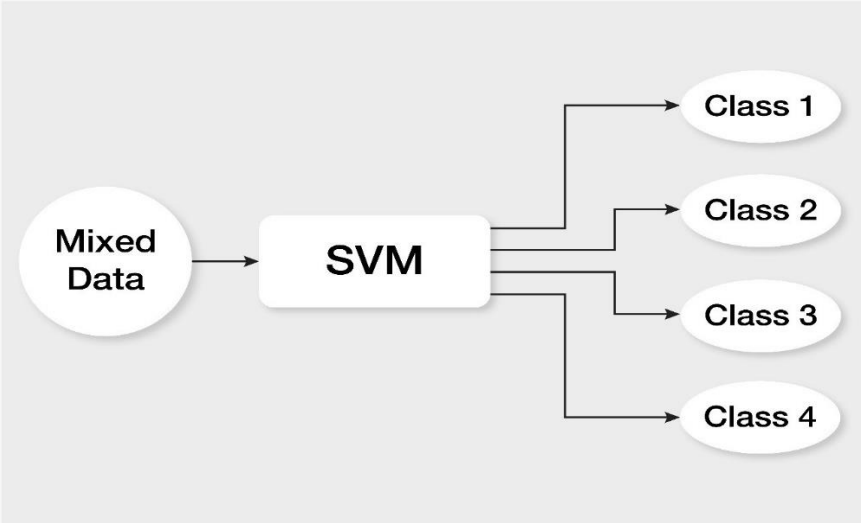
$$y = a + bx$$

Where a and b given by the formulas:

$$b\,(slope) = \frac{n\sum xy - (\sum x)(\sum y)}{n\sum x^2 - (\sum x)^2}$$

$$a\,(intercept) = \frac{n\sum y - b(\sum x)}{n}$$

ii) Support vector machine



iii) Random forests

The training algorithm for random forests applies the general technique of bootstrap aggregating, or bagging, to tree learners. Given a training set $X = x_1, \ldots, x_n$ with responses $Y = y_1, \ldots, y_n$, bagging repeatedly (B times) selects a random sample with replacement of the training set and fits trees to these samples:

For $b = 1, \ldots, B$:

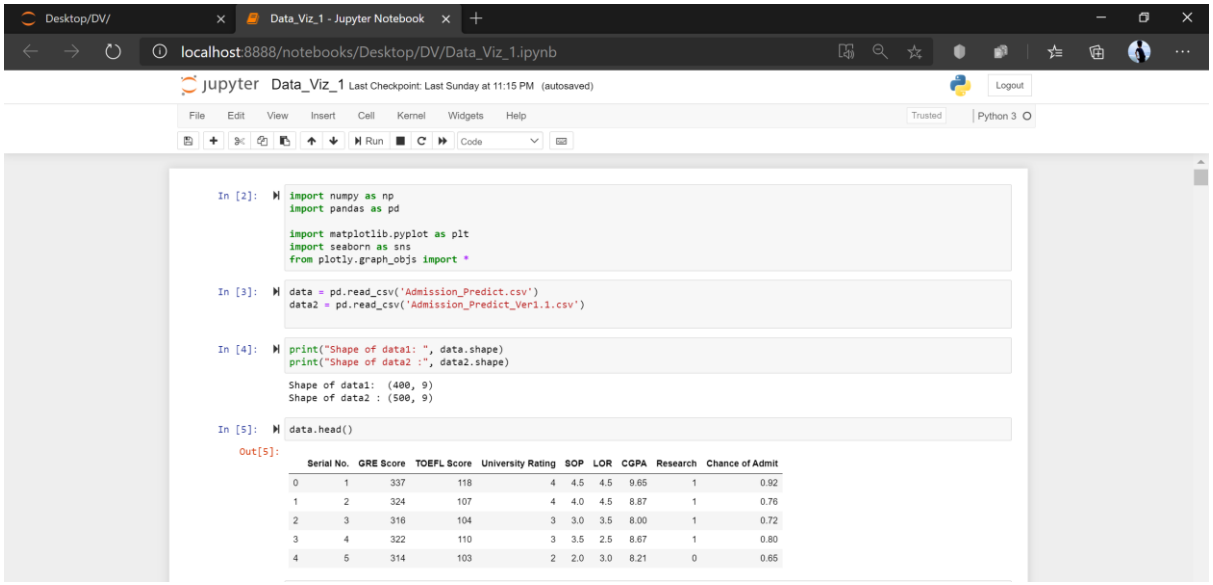
- Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.
- Train a classification or regression tree f_b on X_b, Y_b.

After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x':

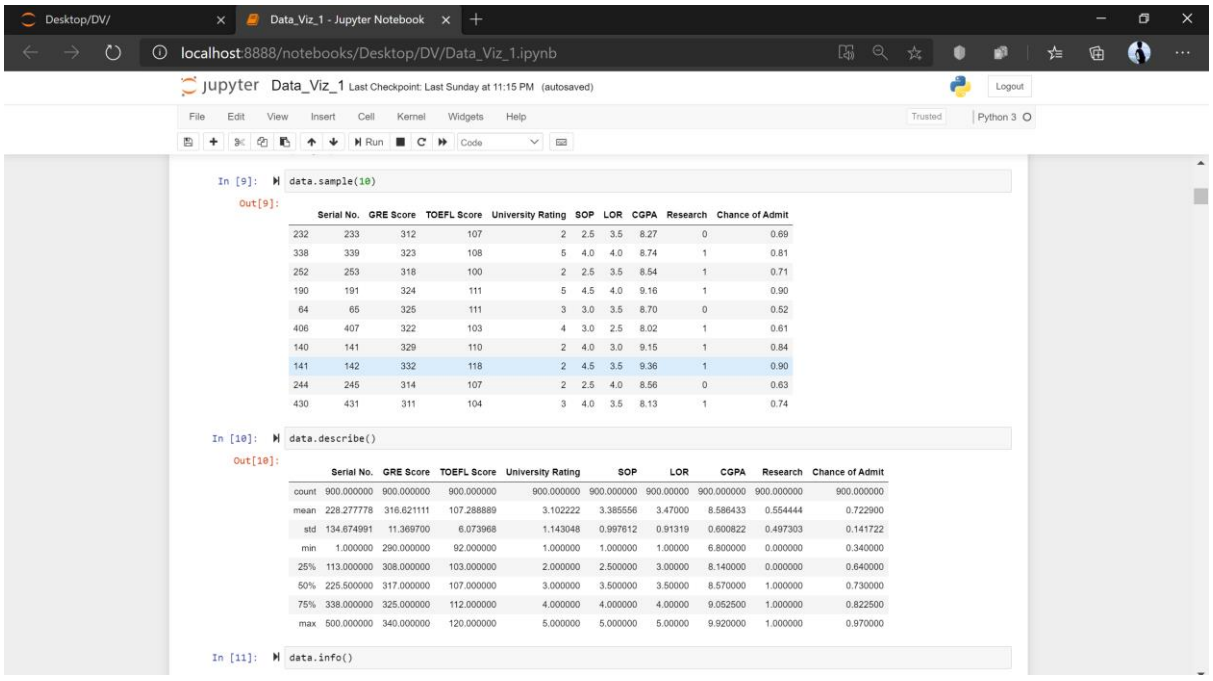
$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

or by taking the majority vote in the case of classification trees.

4.3 PROGRAM



	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
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



	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
232	233	312	107	2	2.5	3.5	8.27	0	0.69
338	339	323	108	5	4.0	4.0	8.74	1	0.81
252	253	318	100	2	2.5	3.5	8.54	1	0.71
190	191	324	111	5	4.5	4.0	9.16	1	0.90
64	65	325	111	3	3.0	3.5	8.70	0	0.52
406	407	322	103	4	3.0	2.5	8.02	1	0.61
140	141	329	110	2	4.0	3.0	9.15	1	0.84
141	142	332	118	2	4.5	3.5	9.36	1	0.90
244	245	314	107	2	2.5	4.0	8.56	0	0.63
430	431	311	104	3	4.0	3.5	8.13	1	0.74

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	900.000000	900.000000	900.000000	900.000000	900.000000	900.00000	900.000000	900.000000	900.000000
mean	228.277778	316.621111	107.288889	3.102222	3.385556	3.47000	8.586433	0.554444	0.722900
std	134.674991	11.369700	6.073968	1.143048	0.997612	0.91319	0.600822	0.497303	0.141722
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.340000
25%	113.000000	308.000000	103.000000	2.000000	2.500000	3.00000	8.140000	0.000000	0.640000
50%	225.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.570000	1.000000	0.730000
75%	338.000000	325.000000	112.000000	4.000000	4.000000	4.00000	9.052500	1.000000	0.822500
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.970000

Desktop/DV/ | Data_Viz_1 - Jupyter Notebook

localhost:8888/notebooks/Desktop/DV/Data_Viz_1.ipynb

jupyter Data_Viz_1 Last Checkpoint: Last Sunday at 11:15 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run Code

```
print("The Minimum Number for Recommendation letters among the students is", lor)
print("The Minimum Number of Research done by students is ", research)
print("The Minimum University Ratings of Different Students is ", uni_rating)
```

```
The Minimum Score for GRE is 290.00
The Minimum Score for TOEFL is 92.00
The Minimum CGPA is 6.80
The Minimum Number for Statement of Purpose is 1.0
The Minimum Number for Recommendation letters among the students is 1.0
The Minimum Number of Research done by students is  0
The Minimum University Ratings of Different Students is  1
```

In [16]:

```
gre = data['GRE Score'].max()
toefl = data['TOEFL Score'].max()
cgpa = data['CGPA'].max()
sop = np.round(data['SOP'].max())
lor = np.round(data['LOR '].max())
research = np.round(data['Research'].max())
uni_rating = np.round(data['University Rating'].max())
print("The Maximum Score for GRE is {:.2f}".format(gre))
print("The Maximum Score for TOEFL is {:.2f}".format(toefl))
print("The Maximum CGPA is {:.2f}".format(cgpa))
print("The Maximum Number for Statement of Purpose is", sop)
print("The Maximum Number for Recommendation letters among the students is", lor)
print("The Maximum Number of Research done by students is ", research)
print("The Maximum University Ratings of Different Students is ", uni_rating)
```

```
The Maximum Score for GRE is 340.00
The Maximum Score for TOEFL is 120.00
The Maximum CGPA is 9.92
The Maximum Number for Statement of Purpose is 5.0
The Maximum Number for Recommendation letters among the students is 5.0
The Maximum Number of Research done by students is  1
The Maximum University Ratings of Different Students is  5
```

In [17]:

```
sns.distplot(data['GRE Score'], color = 'blue')
plt.title('Variations of GRE score among the students', fontsize = 10)
plt.xlabel('GRE Score')
plt.ylabel('count')
plt.style.use('_classic_test')
```

Desktop/DV/ | Data_Viz_1 - Jupyter Notebook

localhost:8888/notebooks/Desktop/DV/Data_Viz_1.ipynb

jupyter Data_Viz_1 Last Checkpoint: Last Sunday at 11:15 PM (autosaved)

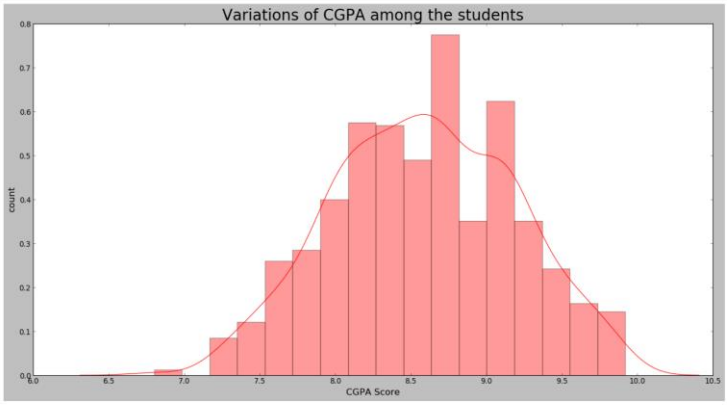
Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

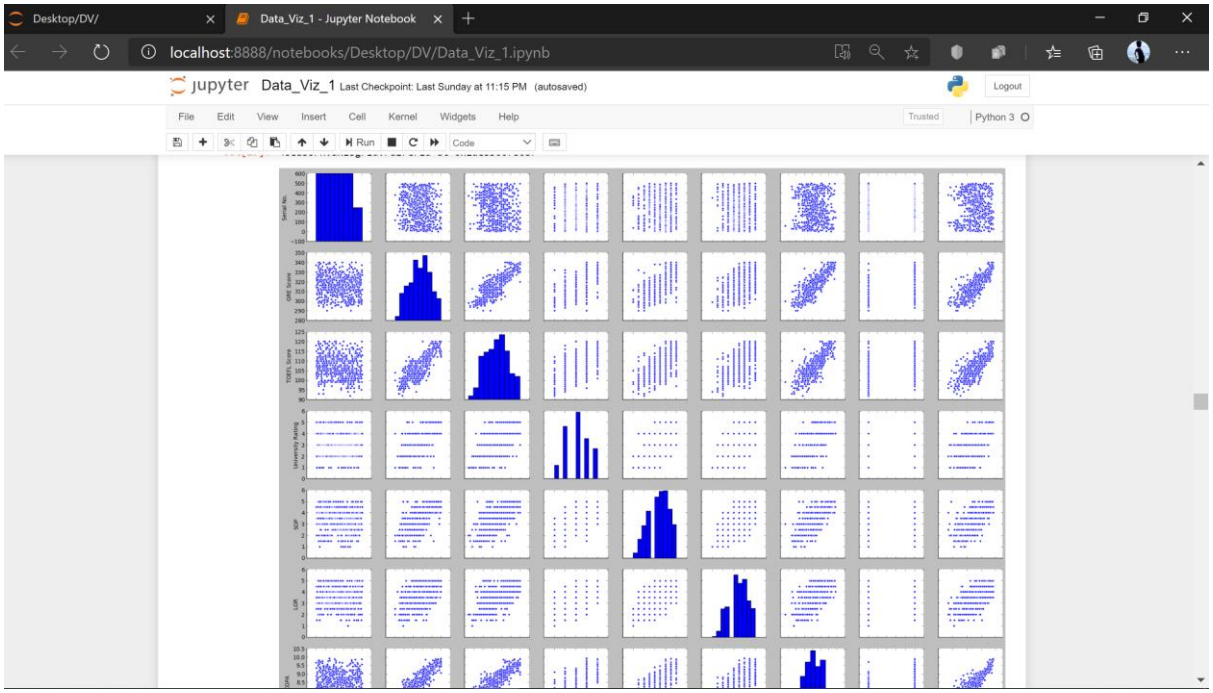
Run Code

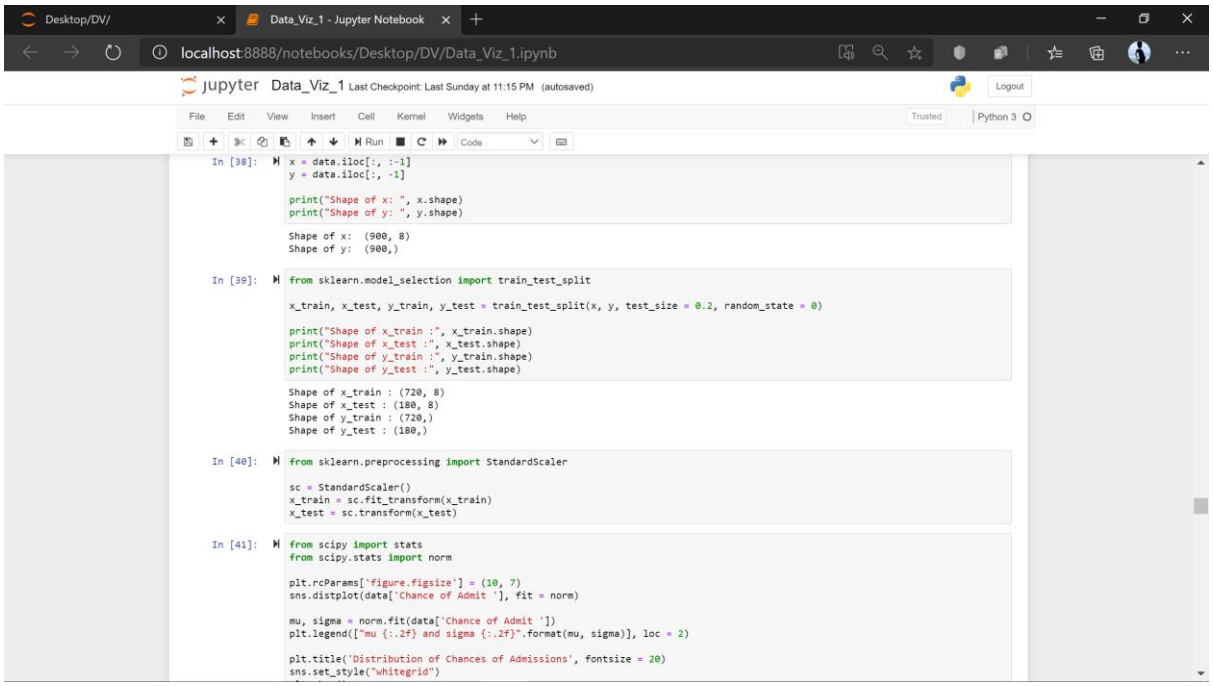
```
plt.title('Variations of CGPA among the students', fontsize = 25)
plt.xlabel('CGPA Score',fontsize = 15)
plt.ylabel('count',fontsize = 15)
plt.rcParams['figure.figsize'] = (20, 10)
plt.show()
```



In [19]:

```
sns.distplot(data['TOEFL Score'], color = 'green')
plt.title('Variations in TOEFL Score', fontsize = 25)
plt.xlabel('TOEFL Score',fontsize = 15)
```





In [38]:
```
x = data.iloc[:, :-1]
y = data.iloc[:, -1]

print("Shape of x: ", x.shape)
print("Shape of y: ", y.shape)
```

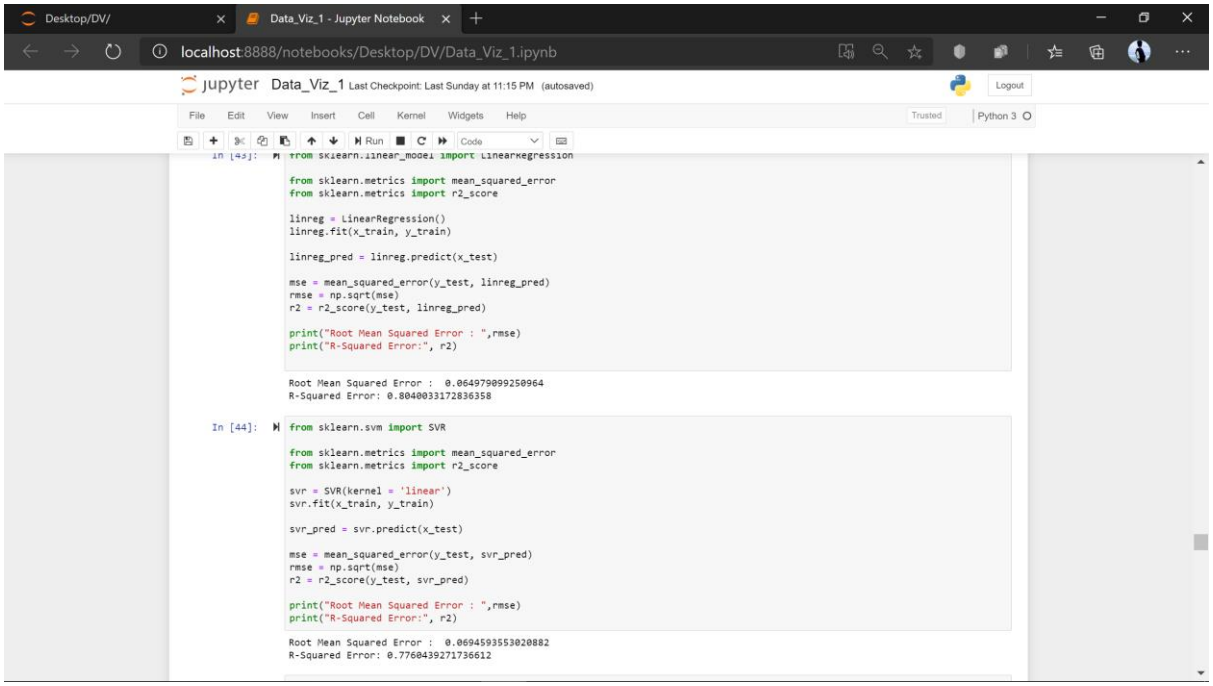
```
Shape of x:  (900, 8)
Shape of y:  (900,)
```

In [39]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)

print("Shape of x_train :", x_train.shape)
print("Shape of x_test :", x_test.shape)
print("Shape of y_train :", y_train.shape)
print("Shape of y_test :", y_test.shape)
```

```
Shape of x_train : (720, 8)
Shape of x_test : (180, 8)
Shape of y_train : (720,)
Shape of y_test : (180,)
```

In [40]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

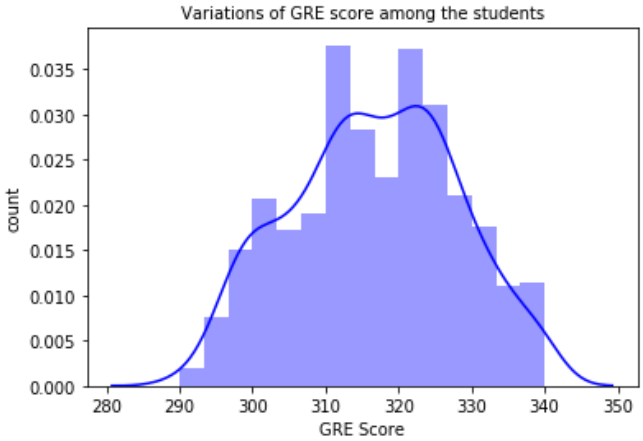
In [41]:
```
from scipy import stats
from scipy.stats import norm

plt.rcParams['figure.figsize'] = (10, 7)
sns.distplot(data['Chance of Admit '], fit = norm)

mu, sigma = norm.fit(data['Chance of Admit '])
plt.legend(["mu {:.2f} and sigma {:.2f}".format(mu, sigma)], loc = 2)

plt.title('Distribution of Chances of Admissions', fontsize = 20)
sns.set_style("whitegrid")
```



5. Result Analysis

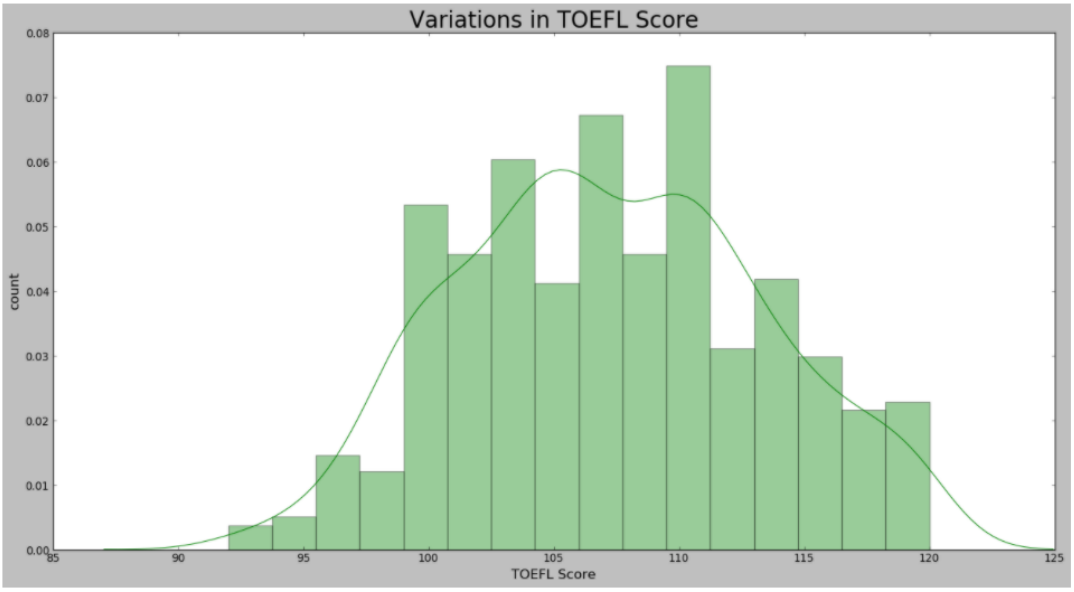
In [17]:

```
sns.distplot(data['GRE Score'], color = 'blue')
plt.title('Variations of GRE score among the students', fontsize = 10)
plt.xlabel('GRE Score')
plt.ylabel('count')
plt.style.use('_classic_test')
plt.rcParams['figure.figsize'] = (20, 10)
plt.show()
```



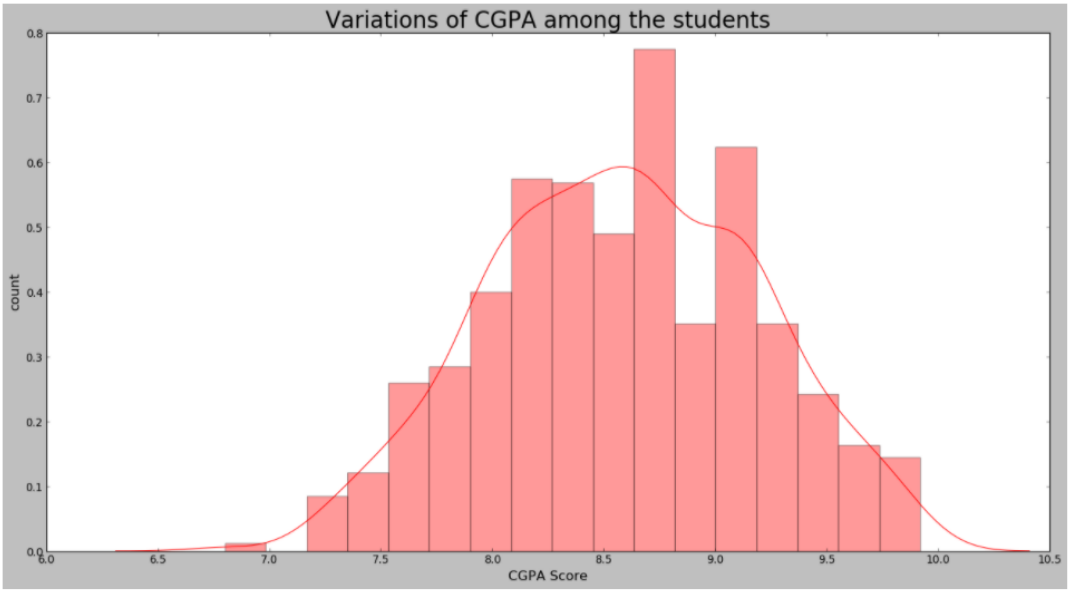
Variations of TOEFL score

In [19]:
```
sns.distplot(data['TOEFL Score'], color = 'green')
plt.title('Variations in TOEFL Score', fontsize = 25)
plt.xlabel('TOEFL Score',fontsize = 15)
plt.ylabel('count',fontsize = 15)
plt.rcParams['figure.figsize'] = (20, 10)
plt.show()
```



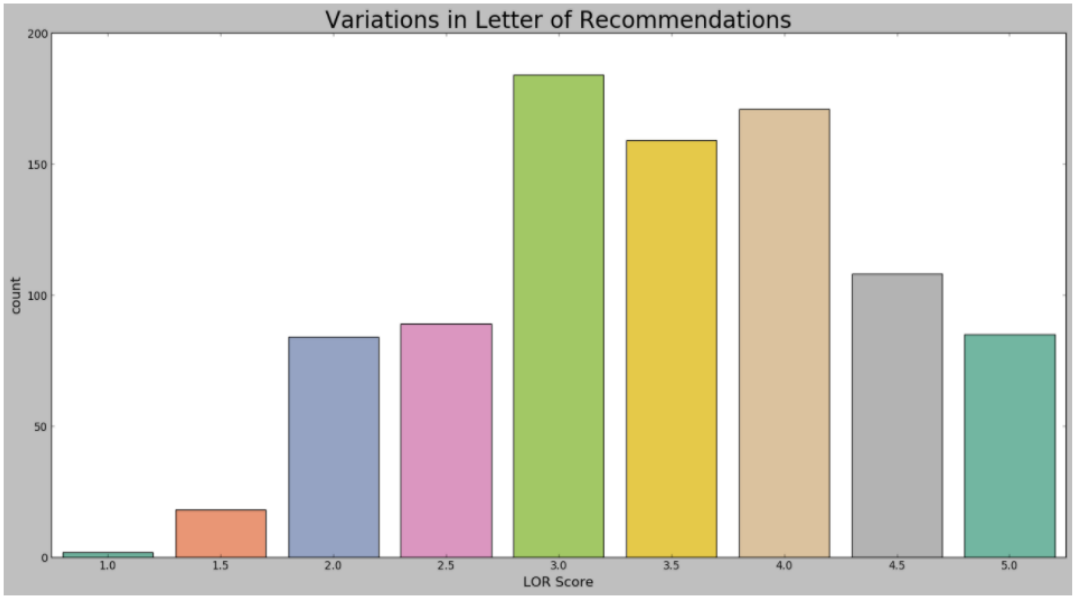
Variation of CGPA among students

In [18]:
```
sns.distplot(data['CGPA'], color = 'red')
plt.title('Variations of CGPA among the students', fontsize = 25)
plt.xlabel('CGPA Score',fontsize = 15)
plt.ylabel('count',fontsize = 15)
plt.rcParams['figure.figsize'] = (20, 10)
plt.show()
```



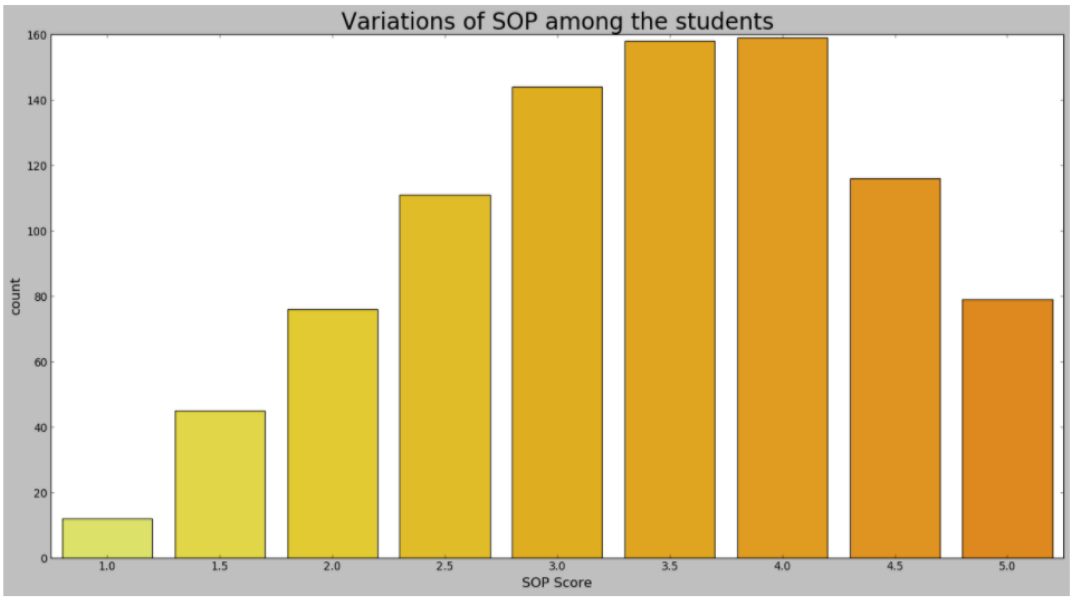
Variation in letter of recommendations

In [20]:
```
plt.rcParams['figure.figsize'] = (20, 10)


sns.countplot(data['LOR '], palette = 'Set2')
plt.title('Variations in Letter of Recommendations', fontsize = 25)
plt.xlabel('LOR Score',fontsize = 15)
plt.ylabel('count',fontsize = 15)
plt.show()
```



Variation of SOP among students

In [21]:
```
plt.rcParams['figure.figsize'] = (20, 10)
sns.countplot(data['SOP'], palette = 'Wistia')
plt.title('Variations of SOP among the students', fontsize = 25)
plt.xlabel('SOP Score',fontsize = 15)
plt.ylabel('count',fontsize = 15)
plt.show()
```

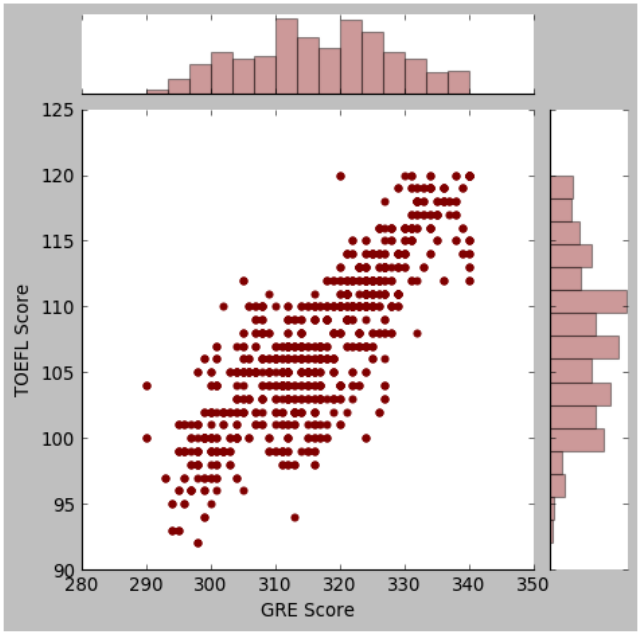


Some Observations of the above graphs:

- From the variations of GRE scores among students' graph, we can see that the GRE scores varies from 290-340 and the most common range is from 310-330
- From the Variations of TOEFL score graph we can observe that the TOEFL scores varies from 92-120 and the common range lying from 106-110
- From the Variation of CGPA among students' graph, we can observe that the cgpa varies from 6.7-9.8(approx.) and the common range of the cgpa is 8.5-9.
- From the Variation in letter of recommendations we can see that the most common range of the LOR is around 3-4
- From the Variation of SOP among students we can see that the most common variation of the SOP is 3-4

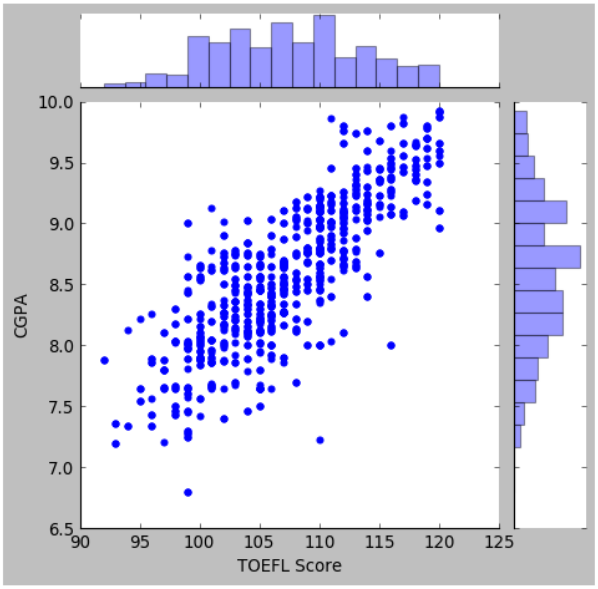
Joint plot for GRE scores

```
In [23]: sns.jointplot(x=data['GRE Score'],y=data['TOEFL Score'],color='maroon')
         plt.rcParams['figure.figsize'] = (20, 10)
         plt.show()
```



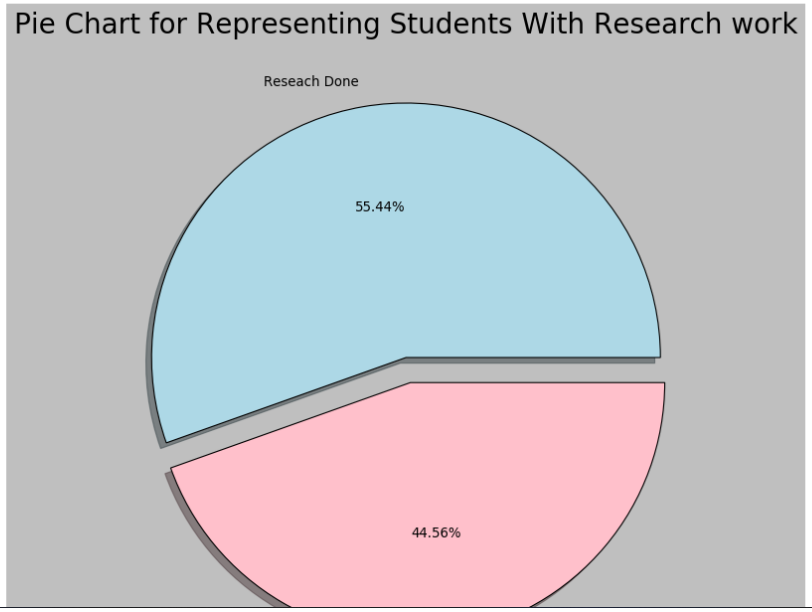
Joint plot for TOEFL scores

```
In [24]: sns.jointplot(x=data['TOEFL Score'],y=data['CGPA'],color='blue')
         plt.rcParams['figure.figsize'] = (20, 10)
         plt.show()
```



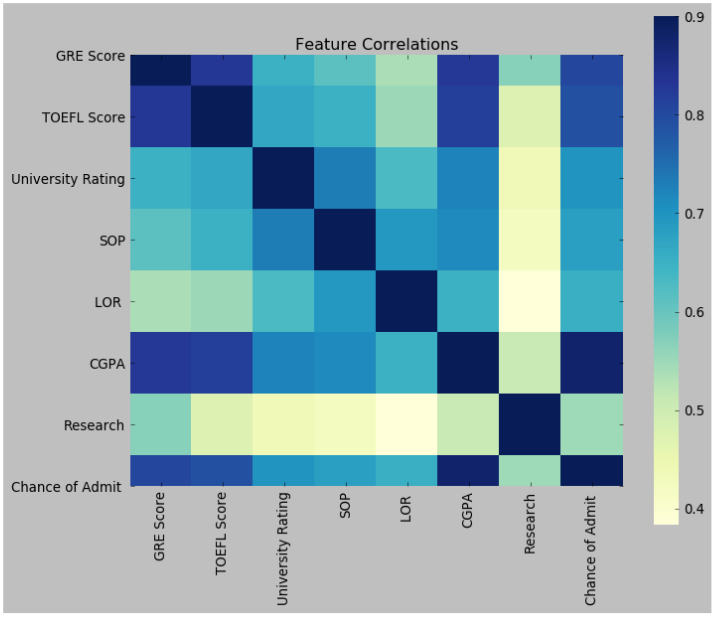
Pie chart for representing student with research work

```
In [26]: size = [499, 401]
         labels = "Reseach Done", "NO Reseach Done"
         colors = ['lightblue', 'pink']
         layout = Layout(
             paper_bgcolor='rgba(0,0,0,0)',
             plot_bgcolor='rgba(0,0,0,0)'
         )
         explode = [0, 0.1]
         plt.pie(size, colors = colors, explode = explode, labels = labels, shadow = True, autopct = '%.2f%%')
         plt.title('Pie Chart for Representing Students With Research work', fontsize = 25)
         plt.axis('off')
         plt.show()
```



Heatmap of the dataset

```
In [31]: fig, ax = plt.subplots(figsize = (10,8))
         _= sns.heatmap(corrmat, vmax=.9,cmap="YlGnBu", square=True, ax=ax)
         ax.set_title('Feature Correlations')
         plt.show(fig)
```

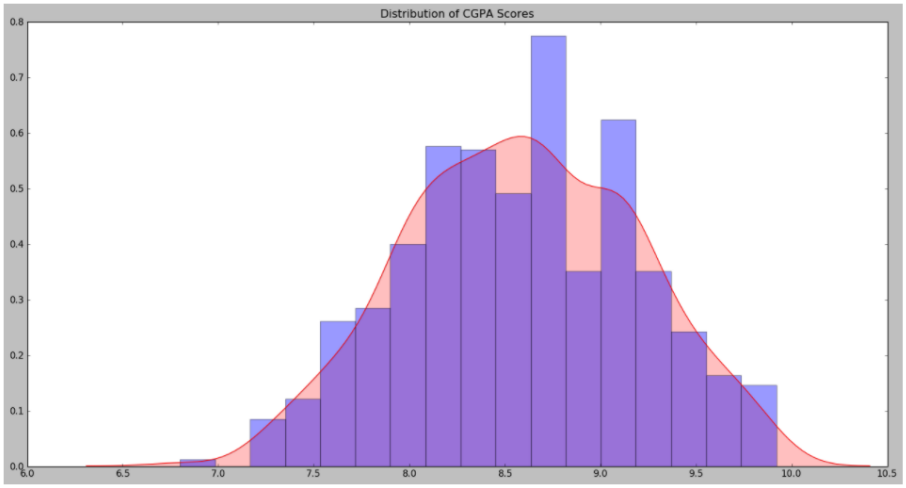


Some observations:

- From the pie chart it is evident that that more than half has done a research paper when then apply for a college.
- From the heatmap we can observe the correlation of every attribute with one another

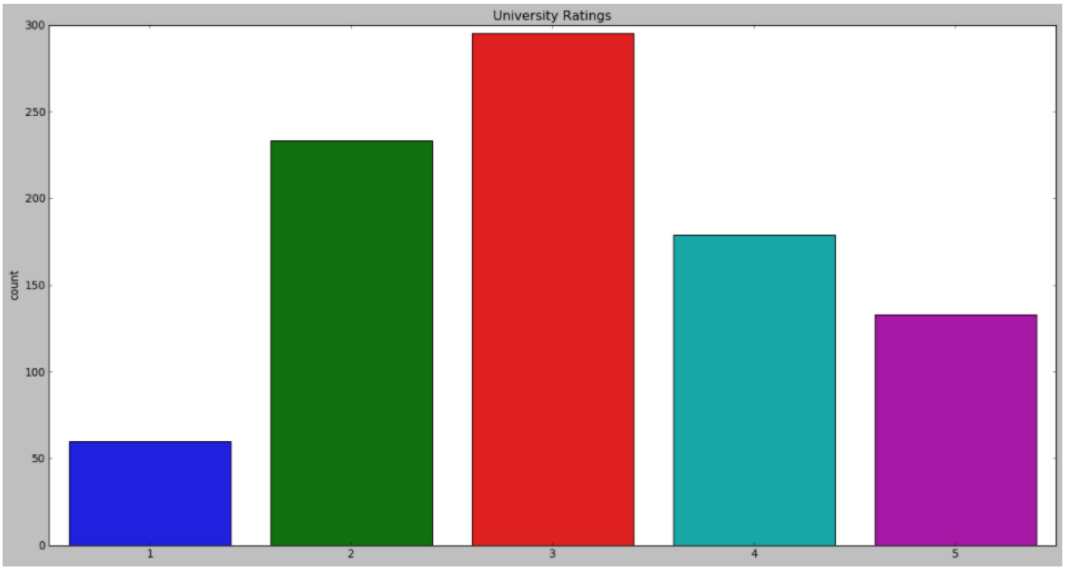
Distribution of CGPA scores

```
In [33]: fig, ax = plt.subplots()
         _ = sns.distplot(data.CGPA, kde=True, kde_kws={'color':'red', 'shade':True, 'lw':1}, ax=ax)
         ax.set_xlabel('')
         ax.set_title("Distribution of CGPA Scores")
         plt.show(fig)
```



University ratings

In [36]:
```
temp = pd.DataFrame(data['University Rating'].value_counts())
temp.columns = ['count']
temp.index.name = 'university_rating'
temp.reset_index(inplace=True)
temp

fig, ax = plt.subplots()
_ =sns.barplot(data = temp, y='count', x = 'university_rating', ax = ax)
ax.set_title("University Ratings")
ax.set_xlabel("")
plt.show(fig)
```



Training the model for prediction

In [38]:
```
x = data.iloc[:, :-1]
y = data.iloc[:, -1]

print("Shape of x: ", x.shape)
print("Shape of y: ", y.shape)
```

```
Shape of x:  (900, 8)
Shape of y:  (900,)
```

In [39]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 0)

print("Shape of x_train :", x_train.shape)
print("Shape of x_test :", x_test.shape)
print("Shape of y_train :", y_train.shape)
print("Shape of y_test :", y_test.shape)
```

```
Shape of x_train : (720, 8)
Shape of x_test : (180, 8)
Shape of y_train : (720,)
Shape of y_test : (180,)
```

Models –

a) LinearRegression

In [43]:
```
from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

linreg = LinearRegression()
linreg.fit(x_train, y_train)

linreg_pred = linreg.predict(x_test)

mse = mean_squared_error(y_test, linreg_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, linreg_pred)

print("Root Mean Squared Error : ",rmse)
print("R-Squared Error:", r2)
```

```
Root Mean Squared Error :  0.064979099250964
R-Squared Error: 0.8040033172836358
```

b) SVR

In [44]:
```
from sklearn.svm import SVR

from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

svr = SVR(kernel = 'linear')
svr.fit(x_train, y_train)

svr_pred = svr.predict(x_test)

mse = mean_squared_error(y_test, svr_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, svr_pred)

print("Root Mean Squared Error : ",rmse)
print("R-Squared Error:", r2)
```

```
Root Mean Squared Error :  0.0694593553020882
R-Squared Error: 0.7760439271736612
```

c) RandomForestRegressor

In [45]:
```
from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

rfr = RandomForestRegressor()
rfr.fit(x_train, y_train)

rfr_pred = linreg.predict(x_test)

mse = mean_squared_error(y_test, rfr_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, rfr_pred)

print("Root Mean Squared Error : ",rmse)
print("R-Squared Error:", r2)
```

```
Root Mean Squared Error :  0.064979099250964
R-Squared Error: 0.8040033172836358
```

d) XGBRegressor

In [47]:
```
from xgboost.sklearn import XGBRegressor

from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

xgb = XGBRegressor()
xgb.fit(x_train, y_train)

xgb_pred = xgb.predict(x_test)

mse = mean_squared_error(y_test, xgb_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, xgb_pred)

print("Root Mean Squared Error : ",rmse)
print("R-Squared Error:", r2)
```

```
Root Mean Squared Error :  0.028381054164711494
R-Squared Error: 0.9626097479857129
```

e) ExtraTreesRegressor

In [48]:
```
from sklearn.ensemble import ExtraTreesRegressor

from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

etr = ExtraTreesRegressor()
etr.fit(x_train, y_train)

etr_pred = etr.predict(x_test)

mse = mean_squared_error(y_test, etr_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, etr_pred)

print("Root Mean Squared Error : ",rmse)
print("R-Squared Error:", r2)
```

```
Root Mean Squared Error :  0.024566350065802524
R-Squared Error: 0.9719855165132695
```

f) Stacked and boosting predictions

In [49]:
```
# stacked predictions

stacked_predictions = np.column_stack((linreg_pred, svr_pred, rfr_pred))

# specifying the meta model
meta_model = LinearRegression()
meta_model.fit(stacked_predictions, y_test)

# final predictions
stacked_predictions = (meta_model.predict(stacked_predictions))

mse = mean_squared_error(y_test, stacked_predictions)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, stacked_predictions)

print("Root Mean Squared Error : ",rmse)
print("R-Squared Error:", r2)
```

```
Root Mean Squared Error :  0.06477825118243952
R-Squared Error: 0.805213082172756
```

In [51]:
```
Boosted_predictions = stacked_predictions*0.2 + xgb_pred*0.3 + etr_pred*0.5

mse = mean_squared_error(y_test, Boosted_predictions)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, Boosted_predictions)

print("Root Mean Squared Error : ",rmse)
print("R-Squared Error:", r2)
```

```
Root Mean Squared Error :  0.02695479998697617
R-Squared Error: 0.9662733213667997
```

6. Conclusion Future Work

From this project we have visualized and shown the visualizations of the dataset. This makes it very easy to read and understand may it be someone with knowledge regarding US college admission or not. We have also shown the co-relation of what all attributes affect the chance of admit feature. To make the prediction we trained several models such as LinearRegression, SVR and more which gave us an accuracy of about 77-80. To increase the accuracy, we boosted up our model. The model we created after doing the stacking up and overall boosting gave us a 97% accuracy which is really good for the size of the dataset we have used. Thus, the max accuracy we achieved was 97%.

As the future work we plan on doing the following

- Integrating the model with Django and deploying it
- Scrapping a bigger dataset
- Getting a more recent dataset for keeping up with the current stats
- Training the model so that we might achieve higher accuracy with the new datasets

7. References

[1] https://psycnet.apa.org/record/2004-12430-002

[2] Jamison, J. (2017). Applying Machine Learning to Predict Davidson College ' s Admissions Yield, pp. 765–766.

[3] Mane, R. V. (2016). Predicting Student Admission decisions byAssociation Rule Mining with Pattern Growth Approach, pp. 202–207.

[4] Abdul Fatah S; M, A. H. (2012). Hybrid Recommender System for Predicting College Admission, pp. 107–113.

[5] MasterPortal (2017). MasterPortal. URL: http://www.mastersportal.eu/countries/82/united-states.html

[6] Mishra, S. and Sahoo, S. (2016). A Quality Based AutomatedAdmission System for Educational Domain, pp. 221–223.

[7] https://medium.com/analytics-vidhya/a-fresh-look-at-graduate-admissions-dataset-d39e4d20803e