# UNIVERSITY ADMISSION PREDICTION

## Introduction

Specific preparation plays a crucial part in your life. Thus education preparation students often have multiple questions about universities which they can get admission and scholarship and accommodation. One of the main concerns is getting admitted to their dream university[1]. It's seen that students still choose to obtain their education from universities that are known internationally. And when it comes to international graduates, the United States of America is the first preference of the majority of them. With most world-renowned colleges, Wide variety of courses available in each discipline, highly accredited education and teaching programs, student scholarships, are available for international students.

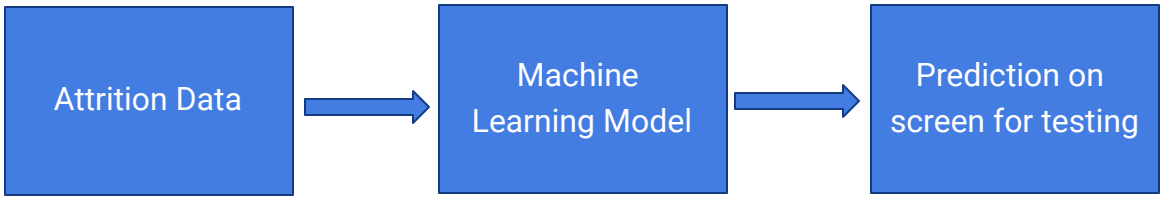
Many colleges in the U.S. follow similar requirements for student admission. Colleges take different factors into account, such as the ranking on aptitude assessment and academic record review[3]. The command over the English language is calculated on the basis of their performance in the English skills test, such as TOEFL and IELTS. The admission committee of universities takes the decision to approve or reject a specific candidate on the basis of the overall profile of the applicant application.

# Literature Survey

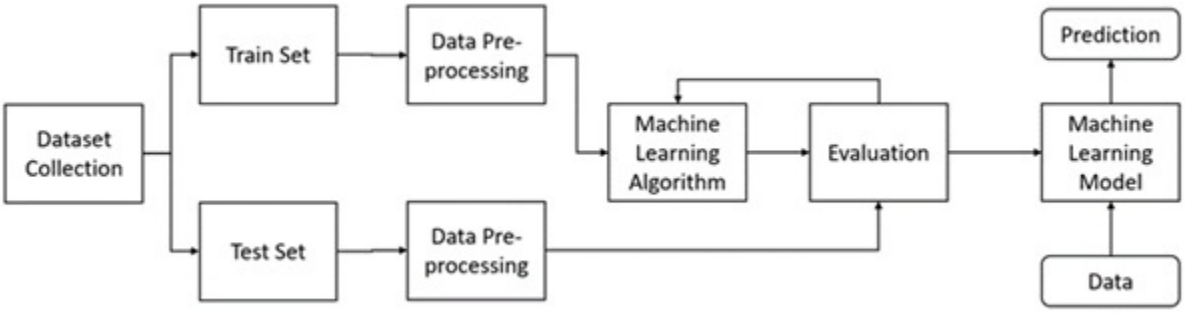
This section includes the literature review of previous research on the assessment of student enrolment opportunities in universities. Numerous programs and studies have been carried out on topics relating to university admission used many machine learning models which helps the students in the admission process to their desired universities. Previous research done in this area used Naive Bayes algorithm which will evaluate the success probability of student application into a respective university but the main drawback is they didn't consider all the factors which will contribute in the student admission process like TOEFL/IELTS, SOP, LOR and under graduate score. Bayesian Networks Algorithm have been used to create a decision support network for evaluating the application submitted by foreign students of the university[5]. This model was developed to forecast the progress of prospective students by comparing the score of students currently studying at university. The model thus predicted whether the aspiring student should be admitted to university on the basis of various scores of students. Since the comparisions are made only with students who got admission into the universities but not with students who got their admission rejected so this method will not be that much accurate.

## Theoretical Analysis:

### Block Diagram:



## Machine Learning Workflow:




## Project Flow:

- User interacts with the UI (User Interface) to enter the current attrition data.
- Entered data are analyzed and predictions are made based on interpretation that whether employee will be attrited or not.
- Predictions are popped onto the UI.

## Data Collection:

The given data set is related to Taxi Fares. It was taken from the website [kaggle.com](kaggle.com)
The website provides various datasets from various domains.

# Data pre-processing

## Importing Libraries:

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

```
from sklearn.preprocessing import MinMaxScaler
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.linear_model.logistic import LogisticRegression
from sklearn.metrics import r2_score
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.svm import SVR
from sklearn import metrics
```

● Pandas: It is a python library mainly used for data manipulation.
● NumPy: This python library is used for numerical analysis.
● Matplotlib and Seaborn: Both are the data visualization library used for plotting graph which will help us for understanding the data.
● Accuracy score: used in classification type problem and for finding accuracy it is used.
● R2 Score: Coefficient of Determination or $R^2$ is another metric used for evaluating the performance of a regression model. The metric helps us to compare our current model with a constant baseline and tells us how much our model is better.
● Train_test_split: used for splitting data arrays into training data and for testing data.
● joblib: to serialize your machine learning algorithms and save the serialized format to a file.
● Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering, and dimensionality reduction via a consistence interface in Python.

## Importing Data set

```
#read_csv is a pandas function to read csv files
data = pd.read_csv('Admission_Predict.csv')
```

```
#head() method is used to return top n (5 by default) rows of a DataFrame or series.
data.head()
```

| | 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 |

● You might have your data in .csv files, .excel files or .tsv files or something else. But the goal is the same in all cases. If you want to analyse that data using pandas, the first step will be to read it into a data structure that's compatible with pandas.

● Let's load a .csv data file into pandas. There is a function for it, called read_csv().We will need to locate the directory of the CSV file at first (it's more efficient to keep the dataset in the same directory as your program).

● Path names on Windows tend to have backslashes in them. But we want them to mean actual backslashes, not special character

## Splitting Data into Train and Test:

● When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test dataset. For this, you will a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.

● Now split our dataset into train set and test using train_test_split class from scikit learn library.

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.30,random_state=101)
#random_state acts as the seed for the random number generator during the split
```

## Model Building:

Training and testing the model:
There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have it may be Classification algorithms or Regression algorithms.
Example:

1. Linear Regression.
2. svm
3. Random Forest Regression / Classification.

You will need to train the datasets to run smoothly and see an incremental improvement in the prediction rate.

**Now we apply Logistic regression algorithm on our dataset.**

**Linear Regression** algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

```
from sklearn.linear_model import LinearRegression
mr=LinearRegression()
mr.fit(x_train,y_train)
y_pred_mr=mr.predict(x_test)
y_pred_mr
```

```
array([[0.50247361],
       [0.62464445],
       [0.63475544],
       [0.64174442],
       [0.68477608],
       [0.69643254],
       [0.80908088],
       [0.49036396],
       [0.86194413],
       [0.81198032],
       [0.79444956],
       [0.69222579],
```

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_mr)
```

```
0.739196581119832
```

# Support Vector Machine

“Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well.

```
from sklearn.svm import SVR
svr=SVR()
#n_estimators no of decision trees
svr.fit(x_train,y_train)
```

```
[ ] y_pred=rf.predict(x_test)
    from sklearn.metrics import r2_score
    r2_score(y_test,y_pred)
```

```
0.7001671952643624
```

```
[ ] from sklearn.svm import SVR
    clf = SVR()
    clf.fit(x_train, y_train)
    predictions=clf.predict(x_test)
    predictions=clf.predict(x_test)
    from sklearn.metrics import r2_score
    R2=r2_score(y_test,predictions)

    print(f'R-square= {round(R2*100,2)}% ')
```

```
R-square= 67.5%
```

## Random Forest Regressor

A Random Forest is an ensemble technique capable of performing both regression and classification tasks with the use of multiple decision trees and a technique called Bootstrap and Aggregation, commonly known as **bagging**. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This part is called Bootstrap.

```
[ ]  from sklearn.ensemble import RandomForestRegressor
     reg_rf = RandomForestRegressor()
     reg_rf.fit(x_train, y_train)
     y_pred = reg_rf.predict(x_test)
```

```
     from sklearn import metrics
     print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
     print('MSE:', metrics.mean_squared_error(y_test, y_pred))
     print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
MAE: 0.05333083333333337
MSE: 0.005491728083333338
RMSE: 0.07410619463535649
```

```
[ ]  metrics.r2_score(y_test, y_pred)
```

```
0.7017184401421837
```

## Predict the values:

Once the model is trained, it's ready to make predictions. We can use the predict method on the model and pass x_test as a parameter to get the output as pred. Notice that the prediction output is an array of real numbers corresponding to the input array.

```
y_pred_mr=mr.predict(x_test)
y_pred_mr
```

```
array([[0.50247361],
       [0.62464445],
       [0.63475544],
       [0.64174442],
       [0.68477608],
       [0.69643254],
       [0.80908088],
       [0.49036396],
       [0.86194413],
       [0.81198032],
       [0.79444956],
       [0.69222579],
       [0.89075942],
       [0.9858619 ],
       [0.57590268],
       [0.75185599],
       [0.69784091],
       [0.6627926 ],
       [0.50179126],
       [0.65109389],
       [0.56572551],
       [0.65806908],
       [0.76724508],
       [0.61221596],
       [0.46283659],
       [0.80452384],
       [0.76183198],
       [0.71704841],
       [0.51951218],
```

## Evaluation:

Finally, we need to check to see how well our model is performing on the test data. There are many evaluation techniques are there. For this, we evaluate r2_score produced by the model.

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_mr)
```

```
0.739196581119832
```

## Saving a model:

Model is saved so it can be used in future and no need to train it again

```
[ ]  import pickle
     pickle.dump(lr,open('university.pkl','wb'))
     model=pickle.load(open('university.pkl','rb'))
```

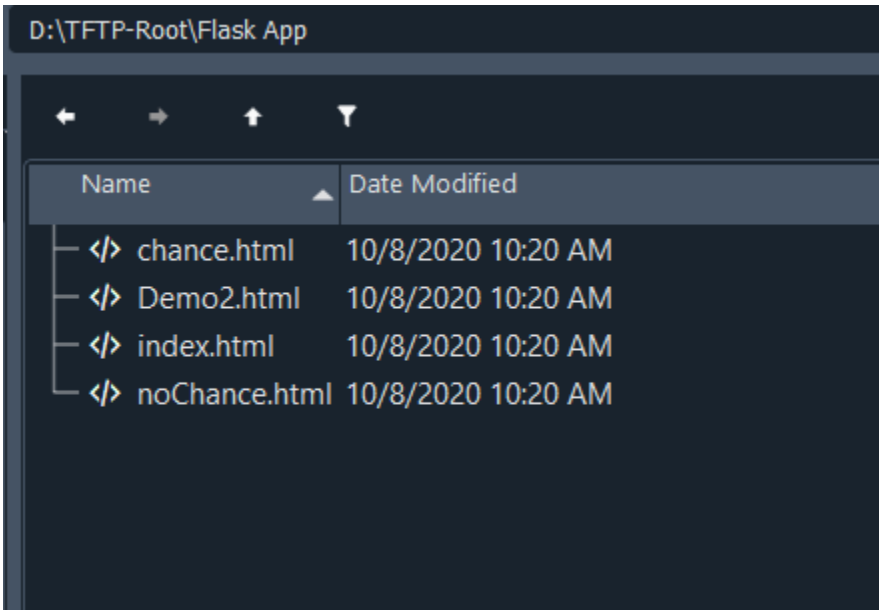
## Application Building:

Creating a HTML File, flask application.

- Build python code
- Importing Libraries
- Routing to the html Page
- Showcasing prediction on UI
- Run The app in local browser.

### Project Structure:

Create a Project folder that contains files as shown below




- We are building a Flask Application that needs HTML pages stored in the templates folder
- Templates folder contains index.html

**Task 1:**

**Importing Libraries**

```
from flask import Flask,render_template,request
import pickle
import numpy as np
```

**Task 2:**

**Routing to the html Page**

```
@app.route('/') # default route
def home():
    return render_template("ccstr.html")


@app.route('/predict',methods=['post'])
def predict():
    cement=float(request.form['Cement'])
    black=float(request.form['BlastFurnaceSlag'])
    fly=float(request.form['FlyAsh'])
    wat=float(request.form['Water'])
    sup=float(request.form['Superplasticizer'])
    cg=float(request.form['CoarseAggregate'])
    fg=float(request.form['FineAggregate'])
    ag=float(request.form['Age'])

    print(cement,black,fly,wat,sup,cg,fg,ag)
    a=np.array([[cement,black,fly,wat,sup,cg,fg,ag]])

    result=rfr.predict(a)

    return render_template('ccstr.html',x=result)
```

**Task 3:**

**Main Function**

```
if __name__ == '__main__':
    app.run(port=8000) # you are running your app
```

**Activity 3: Run the application**

● Open the anaconda prompt from the start menu.
● Navigate to the folder where your app.py resides.
● Now type “python app.py” command.
● It will show the local host where your app is running on http://127.0.0.1:5000/
● Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
● Enter the values, click on the predict button and see the result/prediction on the web page.

# Output Screens:

## UNIVERSITY ADMISSION PREDICTION SYSTEM

**Enter your details and get probability of your admission**

Enter GRE Score 337

Enter TOEFL Score 118

Select University no
- ○ 1
- ◉ 2
- ○ 3
- ○ 4
- ○ 5

Enter SOP 4

Enter LOR 4

Enter CGPA 9.65

Research
- ○ Research
- ◉ NO Research

Predict

**Predicting Chance of Admission**

A Machine Learning Web App using Flask.

Prediction : You have a chance

### Findings and Suggestions

Through Exploratory Data Analysis,
1.The Accuracy for linear regression is 73.9%.
2.The Accuracy for svm  is 70.0%.
3.The Accuracy for Random Forest Regressor is 70.0%

## Conclusion

The main goal of this work is to create a Machine Learning model which could be used by students who want to pursue their education in the US. Many machine learning algorithms were utilized for this research. Linear Regression model compared to other ones. Students can use the model to assess their chances of getting admission into a particular university with an average accuracy of 79 percent. A GUI was developed to make the program, from a non-technical perspective, usable and user-friendly.

## Reference

1.www.kaggle.com
2.www.quora.com
3.www.wikkipedia.com