Executive summary:

As a student, I feel admission grades and jobs are the things that can make every student very anxious. The results for which take minimum a week or more than 6 months.

I have created models to predict quickly how they would score in the future tests in advance and the probability of getting an admit from the university of their dreams.

The third model is of job recommendation. User can put skills under required Qualifications and predict which roles he would be open to.

Execution:

Data Preparation:

The dataset is loaded into python. We will be loading the dataset with Pandas onto grade_pr.csv. Before that, we will also pass in column names for each CSV and read them using pandas

I have attached the screenshots of code for handling missing data from grade_pr.csv

```
In [1]: import pandas as pd
In [2]: df1 = pd.read_csv('C:\\Users\\Shrita\\Downloads\\datasets_for_capstone\\grade_pr.csv')
In [3]: df1
```

```
df1.shape
```

```
(1037, 20)
```

```
import numpy as np
df1.fillna(np.mean(df1),inplace = True)
df1.info()
```

There are many missing values, and I will fill the missing values by replacing it with the mean of respective columns. After using that method, I still found some missing values which I removed later.

```
df1.isnull().sum()
```

```
PPID           0
PP             0
EAL            0
SEN            0
HML           50
Re           120
Wr           122
Ma            50
Att8Est        0
Att8Act        0
Att8Diff       0
EngEst         0
EngAct         0
EngDiff        0
MathsEst       0
MathsAct       0
MathsDiff      0
EbaccEst       0
EbaccAct       0
EbaccDiff      0
dtype: int64
```

```
df1.dropna(inplace = True)
```

(915,20) is the final shape of my data.

The data is clean.

```
df1.shape
```

```
(915, 20)
```

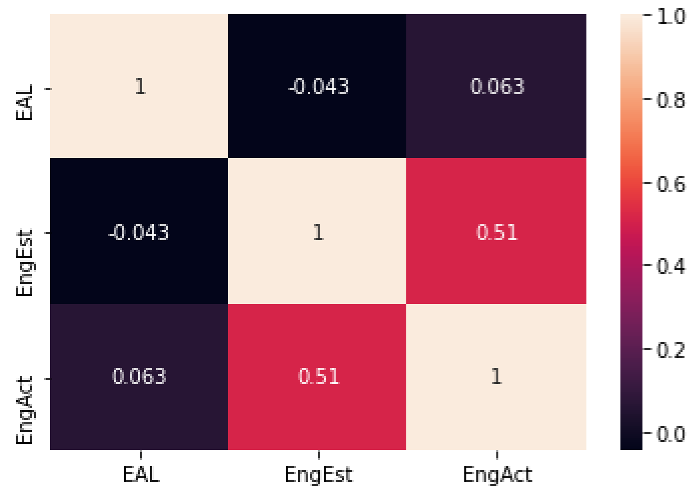
```
col = ['EAL','HML','Re','Wr','EngEst','EngAct']
df1 = df1[col]

import warnings # current version of seaborn generates a bunch of warnings that we'll ignore
warnings.filterwarnings("ignore")
import seaborn as sns
sns.heatmap(df1.corr(), annot = True)
```

Here, I am creating a Dataframe with the columns to filter out the columns.

Here is how the data is correlated

```
<AxesSubplot:>
```




```python
X = df1[['EAL','HML','Re','Wr','EngEst']]
y = df1['EngAct']
```

```python
X = pd.get_dummies(X, drop_first = True)
```

I have X which are the feature variables and y which is the dependent variable. They are now ready to go into the model

Algorithm Evaluation:

1) Linear Regression

```python
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```python
y_pred = lin_reg.predict(X_test)
```

```python
from sklearn.metrics import r2_score, mean_squared_error
import numpy as np
lr_r2 = r2_score(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)
print('Linear Regression R2 Score: {0} \nLinear Regression MSE: {1}, \nLinear Regression RMSE:{2}'.format(lr_r2, lr_mse, lr_rmse
```

```
Linear Regression R2 Score: 0.3137517356762931
Linear Regression MSE: 2.399812312813948,
Linear Regression RMSE:1.5491327615197956
```

2) SVR:

```
import warnings # current version of seaborn generates a bunch of warnings that we'll ignore
warnings.filterwarnings("ignore")
from sklearn.svm import SVR
regressor = SVR(kernel = 'rbf')
regressor.fit(X_train, y_train)
```

```
SVR(C=1.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1,
    gamma='auto_deprecated', kernel='rbf', max_iter=-1, shrinking=True,
    tol=0.001, verbose=False)
```

```
y_pred = regressor.predict(X_test)
```

```
from sklearn.metrics import r2_score, mean_squared_error
import numpy as np
lr_r2 = r2_score(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)
print('Linear Regression R2 Score: {0} \nLinear Regression MSE: {1}, \nLinear Regression RMSE:{2}'.format(lr_r2, lr_mse, lr_rmse
```

```
Linear Regression R2 Score: 0.3099026028088352
Linear Regression MSE: 2.4132727424123908,
Linear Regression RMSE:1.5534711913686687
```

3) Decision Tree

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X, y)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=0, splitter='best')
```

```
y_pred = regressor.predict(X_test)
```

```
from sklearn.metrics import r2_score, mean_squared_error
import numpy as np
lr_r2 = r2_score(y_test, y_pred)
lr_mse = mean_squared_error(y_test, y_pred)
lr_rmse = np.sqrt(lr_mse)
print('Linear Regression R2 Score: {0} \nLinear Regression MSE: {1}, \nLinear Regression RMSE:{2}'.format(lr_r2, lr_mse, lr_rmse
```

```
Linear Regression R2 Score: 0.6374743003454886
Linear Regression MSE: 1.2677534982179672,
Linear Regression RMSE:1.1259456018023106
```

4) Random Forest

```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 3000, random_state = 0)
regressor.fit(X, y)
```

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=3000,
                      n_jobs=None, oob_score=False, random_state=0, verbose=0,
                      warm_start=False)
```

```
y_pred = regressor.predict(X_test)
```

```
rom sklearn.metrics import r2_score, mean_squared_error
nport numpy as np
r_r2 = r2_score(y_test, y_pred)
r_mse = mean_squared_error(y_test, y_pred)
r_rmse = np.sqrt(lr_mse)
rint('Linear Regression R2 Score: {0} \nLinear Regression MSE: {1}, \nLinear Regression RMSE:{2}'.format(lr_r2, lr_mse, lr_rmse))
```

```
Linear Regression R2 Score: 0.6033171915870068
Linear Regression MSE: 1.3872010136874764,
Linear Regression RMSE:1.1777949794796532
```

It is clear, that Decision Tre Regression has the best performance

For Math grades,

```
df1.dropna(inplace = True)
```

```
X = df1[['PP','SEN','Ma','EAL','HML','MathsEst']]
y = df1['MathsAct']
```

```
X = pd.get_dummies(X, drop_first = True)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3 , random_state = 50)
#Splitting the test and train data
```

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X, y)
```

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=0, splitter='best')
```

```
y_pred = regressor.predict(X_test)
```

```
rom sklearn.metrics import r2_score, mean_squared_error
mport numpy as np
r_r2 = r2_score(y_test, y_pred)
r_mse = mean_squared_error(y_test, y_pred)
r_rmse = np.sqrt(lr_mse)
rint('Linear Regression R2 Score: {0} \nLinear Regression MSE: {1}, \nLinear Regression RMSE:{2}'.format(lr_r2, lr_mse, lr_rmse))
```

```
Linear Regression R2 Score: 0.7907128943232796
Linear Regression MSE: 0.7423527794956366,
Linear Regression RMSE:0.8615989667447591
```

Now we move on to the next model:

2) In this data, I will be predicting whether a student will get admission from his dream university based on his/her Gre, Ielts, LOR, Sop Scores and most importantly the university ratings.

I have used SelectKbest to select the best features, but looking at the scores I felt all the features are dependent with chances of admit.

So, in the end I have added all the features.

This is a pure regression problem.

```python
import pandas as pd
df1 = pd.read_csv('C:\\Users\\Shrita\\Downloads\\datasets_for_capstone\\ad_pr.csv')
df1
```

	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
...	...	...	...	...	...	...	...	...	...
495	496	332	108	5	4.5	4.0	9.02	1	0.87
496	497	337	117	5	5.0	5.0	9.87	1	0.96
497	498	330	120	5	4.5	5.0	9.56	1	0.93
498	499	312	103	4	4.0	5.0	8.43	0	0.73
499	500	327	113	4	4.5	4.5	9.04	0	0.84

500 rows × 9 columns

```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3 , random_state = 0)
#Splitting the test and train data
```

```python
from sklearn.feature_selection import SelectKBest, chi2
import sklearn
# configure to select all features
fs = SelectKBest(score_func=sklearn.feature_selection.f_regression, k='all')
# learn relationship from training data
fs.fit(X_train, y_train)
# transform train input data
X_train_fs = fs.transform(X_train)
# transform test input data
X_test_fs = fs.transform(X_test)
X_train_fs, X_test_fs, fs
```

```python
for i in range(len(fs.scores_)):
    print('Feature %d: %f' % (i, fs.scores_[i]))
```

```
Feature 0: 723.855183
Feature 1: 608.715383
Feature 2: 316.494754
Feature 3: 291.796145
Feature 4: 252.894744
Feature 5: 1322.325391
Feature 6: 138.313677
```

```python
X1 = df1[['CGPA', 'GRE Score','TOEFL Score']]
y1 = df1['Chance of Admit']
```

```python
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(X1, y1, test_size = 0.3 , random_state = 0)
#Splitting the test and train data
X_train
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
141	332	118	2	4.5	3.5	9.36	1
383	300	100	3	3.0	3.5	8.26	0
135	314	109	4	3.5	4.0	8.77	1
493	300	95	2	3.0	1.5	8.22	1
122	310	106	4	1.5	2.5	8.36	0
...	...	...	...	...	...	...	...
323	305	102	2	2.0	2.5	8.18	0
192	322	114	5	4.5	4.0	8.94	1
117	290	104	4	2.0	2.5	7.46	0
47	339	119	5	4.5	4.0	9.70	0
172	322	110	4	4.0	5.0	9.13	1

```python
from sklearn.model_selection import train_test_split
X_train10, X_test10, y_train10, y_test10 = train_test_split(X10, y10, test_size = 0.2 , random_state = 50)
```

```python
from sklearn.linear_model import LinearRegression
lin_reg10 = LinearRegression()
lin_reg10.fit(X_train10, y_train10)
```

```python
y_pred10 = lin_reg10.predict(X_test10)
```

```python
pd.DataFrame({"Actual": y_test10, "Predict": y_pred10})
```

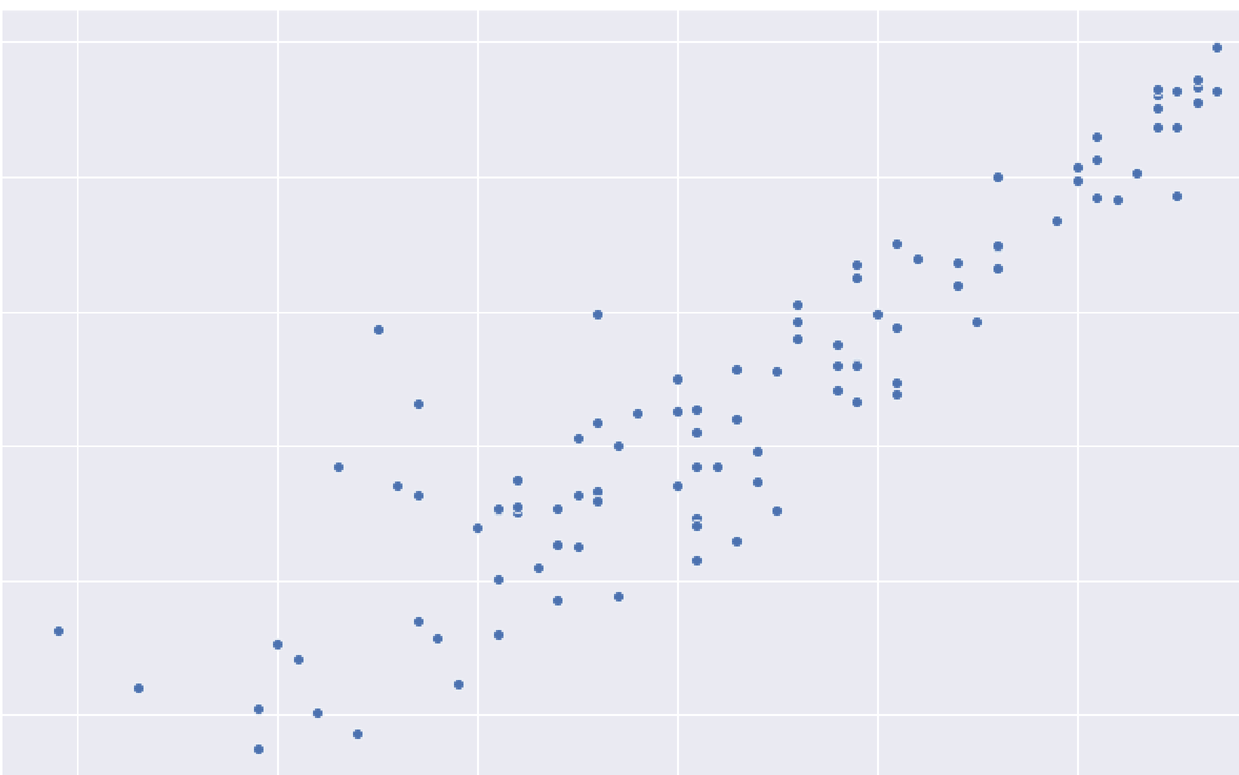
```python
from sklearn.metrics import r2_score, mean_squared_error
import numpy as np
lr_r2 = r2_score(y_test10, y_pred10)
lr_mse = mean_squared_error(y_test10, y_pred10)
lr_rmse = np.sqrt(lr_mse)
print('Linear Regression R2 Score: {0} \nLinear Regression MSE: {1}, \nLinear Regression RMSE:{2}'.format(lr_r2, lr_mse, lr_rmse
```

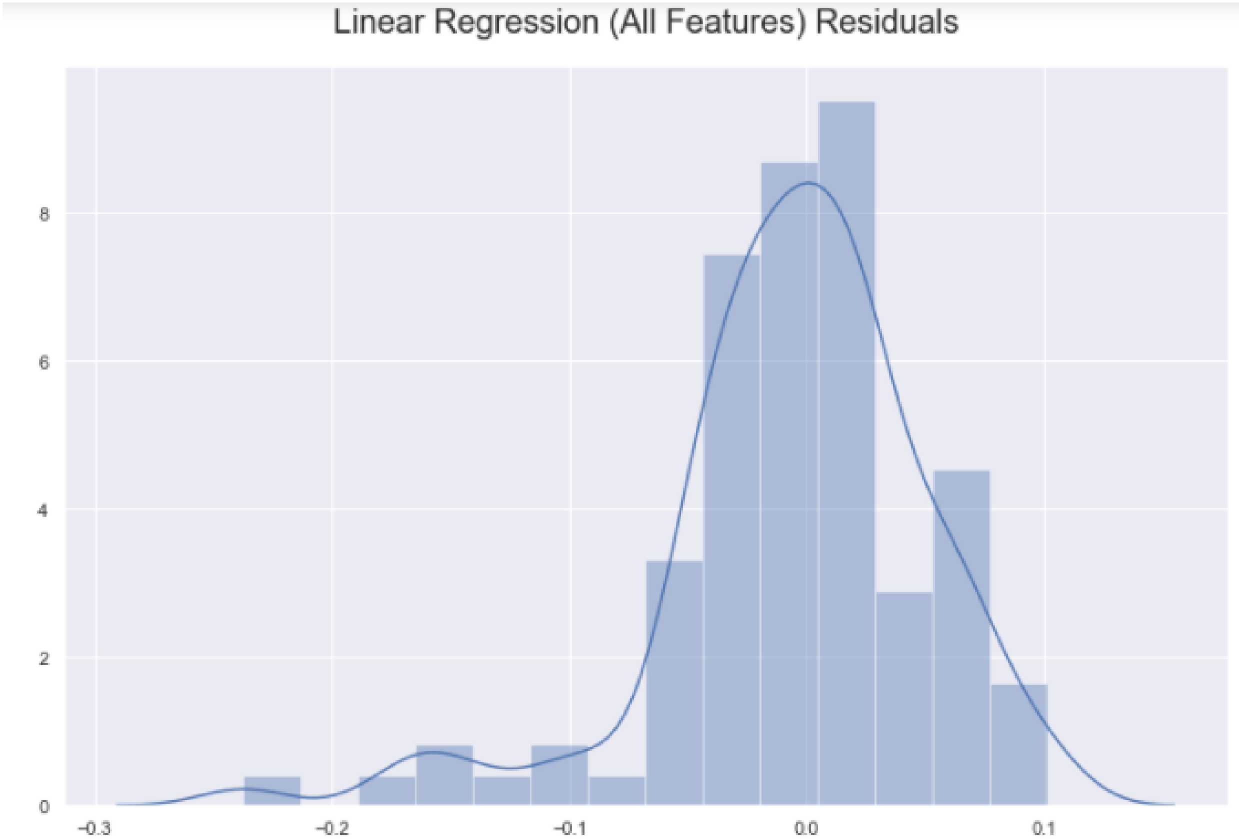
```
Linear Regression R2 Score: 0.8413731950456493
Linear Regression MSE: 0.0032409296323111297,
Linear Regression RMSE:0.05692916328483258
```

```python
pop = [[338,112,5,5,5,9.5,1]]
y_pred101 = lin_reg10.predict(pop)
```

```python
y_pred101
```

```
array([0.93670222])
```





Now we move on to the last model, which is job recommendation

I have completed my EDA in the data and learning NLP side by side

```
In [1]: import pandas as pd

In [2]: df1 = pd.read_csv('C:\\Users\\Shrita\\Downloads\\job_rec.csv')

In [3]: df1 = df1[df1['IT'] == True]

In [4]: col = ['RequiredQual', 'Eligibility', 'Title', 'JobDescription', 'JobRequirment']
        df1 = df1[col]

In [5]: df1['Title'].value_counts().head(30)

Out[5]: Software Developer                                    134
        Web Developer                                         101
        Java Developer                                         88
        Graphic Designer                                       75
        Software Engineer                                      69
        Senior Java Developer                                  69
        PHP Developer                                          65
        Senior Software Engineer                               63
        Programmer                                             56
        IT Specialist                                          55
        Senior QA Engineer                                     43
        Senior Software Developer                              41
        Android Developer                                      37
        .NET Developer                                         36
        Senior .NET Developer                                  34
        Senior PHP Developer                                   34
        iOS Developer                                          31
        Senior Web Developer                                   29
```

As you can see there are many job roles even after selecting only IT jobs.

For this I have tried to combine many columns

In [ ]:

```
def repl(title):
    tokens = title.split()
    for i,x in enumerate(tokens):
        if x == 'Senior':
            tokens = tokens[1:]
        elif x == 'Junior':
            tokens = tokens[1:]
    title2 = ' '.join(tokens)
    return title2

df1['Title'] = df1['Title'].apply(lambda x: repl(x))
```

In [ ]:

```
df1['Title'].value_counts()
```

In [ ]:

```
def repl(s):
    if 'ASP' in s:
        s = s.replace(s, '.NET Developer')
    elif '.NET' in s:
        s = s.replace(s, '.NET Developer')
    elif 'C++' in s:
        s = s.replace(s, 'C++ Developer')

    else:
        pass

    return s
df1['Title'] = df1['Title'].apply(lambda x: repl(x))
```

In [ ]:

```
df1['Title']
```

This is the final result for Data cleaning on Job_data

```
df1['Title'].value_counts().head(60)
```

```
.NET Developer                                      237
Software Developer                                  181
Java Developer                                      165
C++ Developer                                       162
Software Engineer                                   141
Web Developer                                       132
PHP Developer                                       103
Graphic Designer                                     78
Android Developer                                    60
Programmer                                           58
IT Specialist                                        56
iOS Developer                                        54
QA Engineer                                          43
Java Software Developer                              43
Database Developer                                   38
Senior Software Developer                            33
Database Administrator                               31
Software QA Engineer                                 31
Senior Developer/ Architect                          28
Network Administrator                                27
Software Engineer, Deep Submicron Department         26
Quality Assurance Engineer                           22
.Net Developer                                       21
Credit Specialist                                    18
IT Manager                                           18
Frontend Developer                                   18
Technical Support Specialist                         17
Developer                                            17
Technical Support Engineer                           15
Embedded Linux BSP Engineer                          14
Software Engineer, Design to Silicon Division        14
Embedded Software Engineer                           14
Software Development Manager                         13
Flash Developer                                      13
Software Architect                                   13
```

Algorithm Evaluation:

1) I have tried Regression models for this problem directly but they gave very less accuracy because the data needed to be transformed by using TF-IDF

Candidate Algorithm Selection (3) and Rationale:

1) For student grade prediction, It is clearly seen that out of all the models, decision tree performed better. That is why I have decided to move further with Decision Tree Regressor.
2) For student admission, which is a pure regression model, I have tried to add the best features possible
3) For job prediction, out of logistic regression and Naïve bayes, after applying TF-IDF Naïve bayes would have higher accuracy. So I will train this model on Naïve Byes regression.