Junran Cao
Professor Amir Hossein Jafari
Intro to Data Mining – DATS 6103
12/6/2021

Individual Report

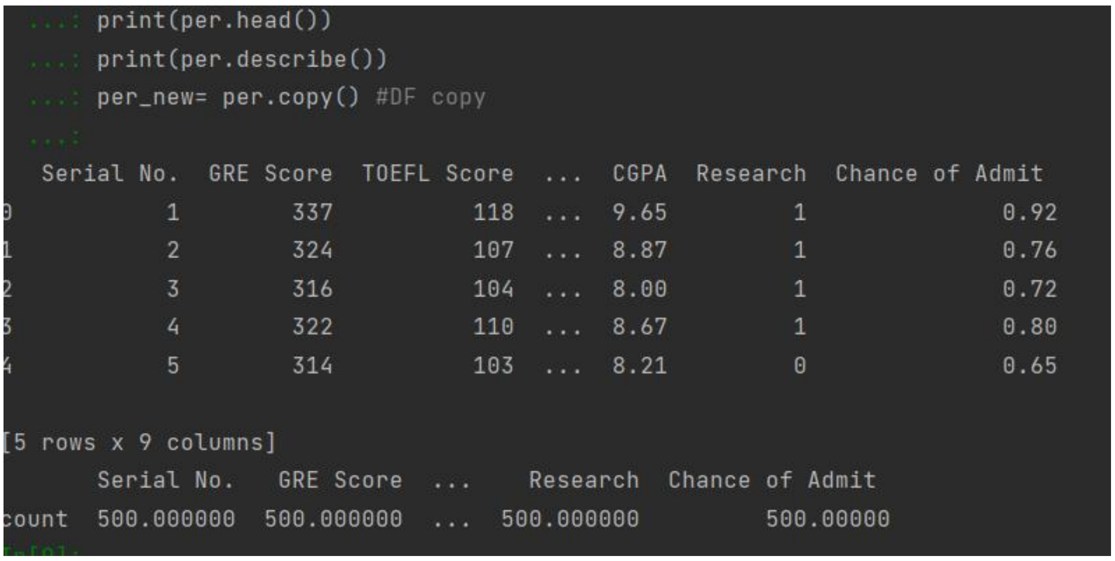
Introduction

Our project examined the factors that affected admission to the master's programs in the US to see how well they predicted the admission.

The project was into four major parts: (1) preprocessing and cleaning, (2) visualization, (3) data analysis (i.e., cross-validation, linear regression analysis, Decision Tree analysis, and Random Forest Analysis), and (4) building the GUI. My work was preprocessing and cleaning the data. In addition to that, I also drafted the Group Final report.

Individual Work

Before doing any processing and cleaning, I had known that our data set contained a variety of variables. Therefore, I decided to start the processing and cleaning by checking the structure of the data set using head() and describe() function. Figure 1 showed me how the data set looked like, providing a clue to my cleaning and preprocessing followed by.



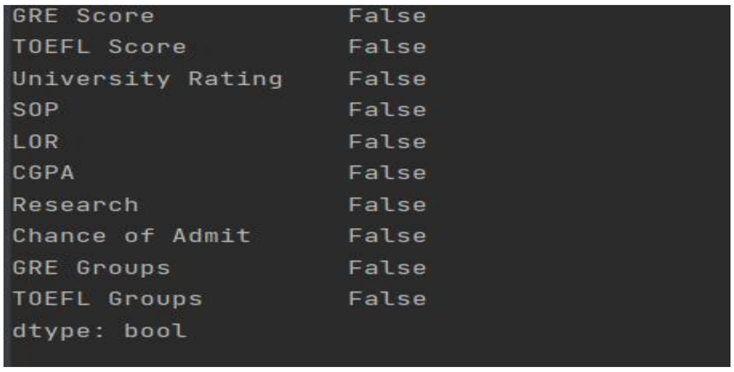
```
   ...: print(per.head())
   ...: print(per.describe())
   ...: per_new= per.copy() #DF copy
   ...:
   Serial No.  GRE Score  TOEFL Score  ...  CGPA  Research  Chance of Admit
0           1        337          118  ...  9.65         1             0.92
1           2        324          107  ...  8.87         1             0.76
2           3        316          104  ...  8.00         1             0.72
3           4        322          110  ...  8.67         1             0.80
4           5        314          103  ...  8.21         0             0.65

[5 rows x 9 columns]
       Serial No.   GRE Score  ...    Research  Chance of Admit
count  500.000000  500.000000  ...  500.000000         500.00000
```

Figure 1

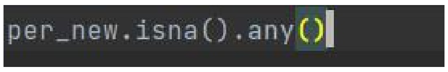
Data Cleaning:

I eliminated the White Space in 'Chance of Admit ' and 'LOR ' by renaming the variables as 'Chance of Admit' and 'LOR'. I also checked for missing values by checking for 'NA' values and found no missing values. I also checked outliers with inter-quartile range detection for the integer and float variables, which are 'GRE Score', 'TOEFL Score', 'SOP', 'LOR', 'CGPA', and 'Chance of admit'. Doing so, I used for-in loop. I first dropped the columns: 'University Rating', 'Research', 'GRE Groups', and 'TOEFL Groups'. I then used Numpy to define the inter-quartile ranges. After that, I showed in which array the outliers were. Finally, in case that the authors may accidentally entered the wrong values for categorical variables when preparing the data set, I checked the outliers for categorical variables, which are 'University Rating' and 'Research'. No outliers were detected.



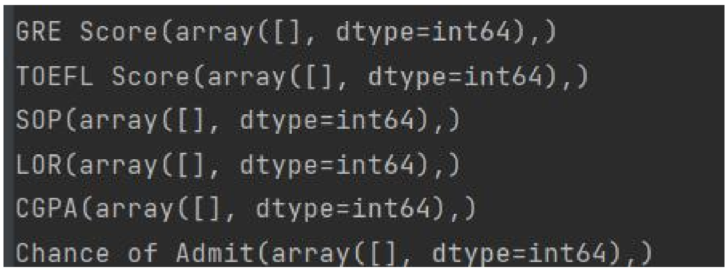
```
GRE Score                False
TOEFL Score              False
University Rating        False
SOP                      False
LOR                      False
CGPA                     False
Research                 False
Chance of Admit          False
GRE Groups               False
TOEFL Groups             False
dtype: bool
```

Figure 2



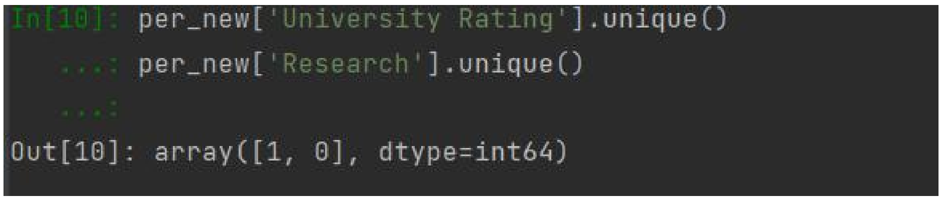
```
per_new.isna().any()
```

Figure 3



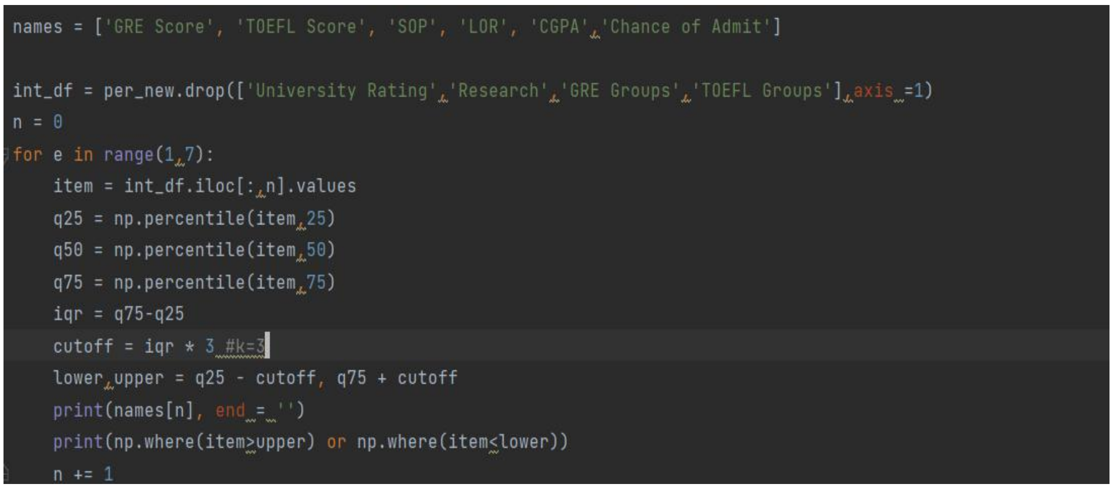
```
GRE Score(array([], dtype=int64),)
TOEFL Score(array([], dtype=int64),)
SOP(array([], dtype=int64),)
LOR(array([], dtype=int64),)
CGPA(array([], dtype=int64),)
Chance of Admit(array([], dtype=int64),)
```

Figure 4



```
In[10]: per_new['University Rating'].unique()
   ...: per_new['Research'].unique()
   ...:
Out[10]: array([1, 0], dtype=int64)
```

Figure 5



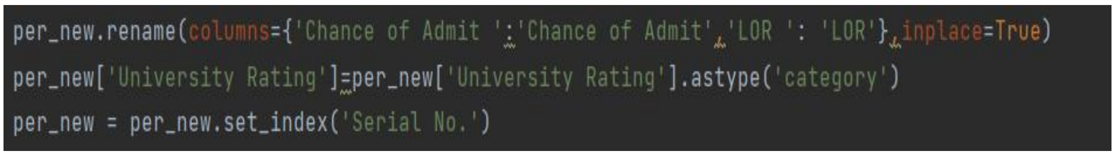
```
names = ['GRE Score', 'TOEFL Score', 'SOP', 'LOR', 'CGPA','Chance of Admit']

int_df = per_new.drop(['University Rating','Research','GRE Groups','TOEFL Groups'],axis =1)
n = 0
for e in range(1,7):
    item = int_df.iloc[:,n].values
    q25 = np.percentile(item,25)
    q50 = np.percentile(item,50)
    q75 = np.percentile(item,75)
    iqr = q75-q25
    cutoff = iqr * 3 #k=3
    lower,upper = q25 - cutoff, q75 + cutoff
    print(names[n], end = '')
    print(np.where(item>upper) or np.where(item<lower))
    n += 1
```

Figure 6

Data Transformation:

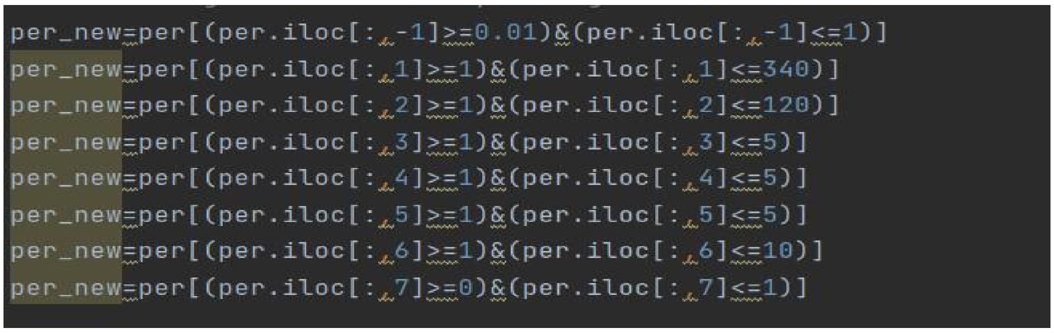
I made 'University Rating' a categorical variable and set 'Serial No.' as the index.



```
per_new.rename(columns={'Chance of Admit ':'Chance of Admit','LOR ': 'LOR'},inplace=True)
per_new['University Rating']=per_new['University Rating'].astype('category')
per_new = per_new.set_index('Serial No.')
```

Figure 7

Again, for the accuracy of the analysis. I "restricted" the data values, keeping only the valid values. Note that in the data set, we had: 'GRE score' (out of 340); 'TOEFL score' (out of 120); 'University rating' (out of 5); 'SOP' (out of 5), 'LOR' (out of 5); 'CGPA (out of 10); 'Research' (either 0 or 1), and 'Chance of admit' (from 0-1). I selected 'Chance of admit' with the values that were greater than or equal to 0.01 and less than or equal to 1; 'GRE score' with the values that were greater than or equal to 1 and less than or equal to 340; 'TOEFL score' with the values that were greater than or equal to 1 and less than or equal to 120; 'University rating' with the values that were greater than or equal to 1 and less less or equal to 5; 'SOP' with the values greater than or equal to 1 and less than or equal to 5; 'LOR' with the values that were greater than or equal to 1 and less than or equal to 5; 'CGPA' with the values that were greater than or equal to 1 and less than or equal to 10; and 'Research' with the values greater than or equal to 0 and less than or equal to 1. Figure 8 illustrates the process above.

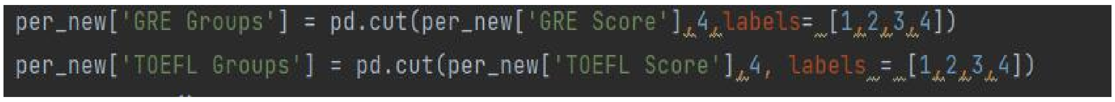


```
per_new=per[(per.iloc[:,-1]>=0.01)&(per.iloc[:,-1]<=1)]
per_new=per[(per.iloc[:,1]>=1)&(per.iloc[:,1]<=340)]
per_new=per[(per.iloc[:,2]>=1)&(per.iloc[:,2]<=120)]
per_new=per[(per.iloc[:,3]>=1)&(per.iloc[:,3]<=5)]
per_new=per[(per.iloc[:,4]>=1)&(per.iloc[:,4]<=5)]
per_new=per[(per.iloc[:,5]>=1)&(per.iloc[:,5]<=5)]
per_new=per[(per.iloc[:,6]>=1)&(per.iloc[:,6]<=10)]
per_new=per[(per.iloc[:,7]>=0)&(per.iloc[:,7]<=1)]
```

Figure 8

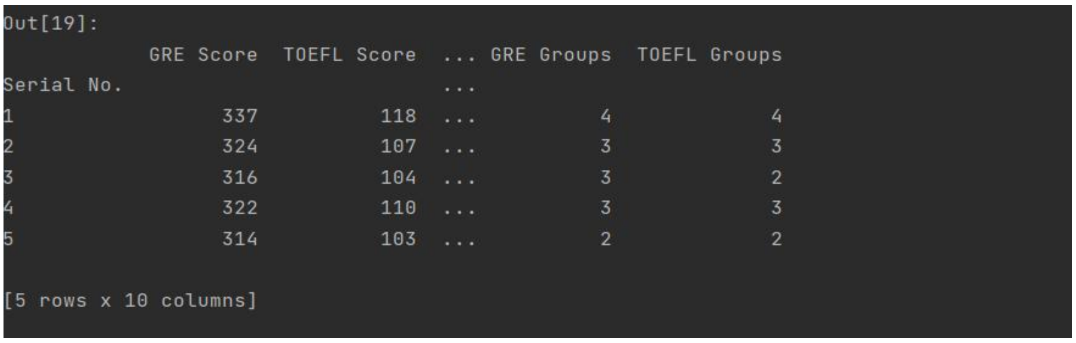
Data Integration:

To make our analysis even more accurate, I binned 'GRE score' and 'TOEFL score' into four groups (1,2,3,4). The algorithm used here is binning. Data binning is a way of assigning original data values into a bin which they fit according to their size. The original values are then replaced by values representing the corresponding intervals.[1]



```
per_new['GRE Groups'] = pd.cut(per_new['GRE Score'],4,labels=[1,2,3,4])
per_new['TOEFL Groups'] = pd.cut(per_new['TOEFL Score'],4, labels = [1,2,3,4])
```

Figure 9



```
Out[19]:
           GRE Score  TOEFL Score  ... GRE Groups  TOEFL Groups
Serial No.                         ...
1                337          118  ...          4             4
2                324          107  ...          3             3
3                316          104  ...          3             2
4                322          110  ...          3             3
5                314          103  ...          2             2

[5 rows x 10 columns]
```

Figure 10

Summary

After preprocessing and cleaning the data using Pandas and Numpy, I saw that the data set has improved a lot, showing no outliers and variable types that were coded wrong. I believed that the cleaned and preprocessed data set assisted our analysis. I learned that cleaning and preprocessing data was not solely removing outliers; instead, it was closely related to the goal of the project (i.e., how we wanted the variables to look like and how to present them

[1] *Python: Binning method for data smoothing*. GeeksforGeeks. (2019, May 20). Retrieved December 5, 2021, from https://www.geeksforgeeks.org/python-binning-method-for-data-smoothing/#:~:text=Binning%20method%20is%20used%20to,values%2C%20they%20perform%20local%20smoothing.

accurately). To improve the process, I would include noise detection, which would detect random errors or variances in the variables.

Code Score: I did not copy nor find any codes from the Internet.

References

Python: Binning method for data smoothing. GeeksforGeeks. (2019, May 20). Retrieved December 5, 2021, from https://www.geeksforgeeks.org/python-binning-method-for-data-smoothing/#:~:text=Binning%20method%20is%20used%20to,values%2C%20they%20perform%20local%20smoothing.