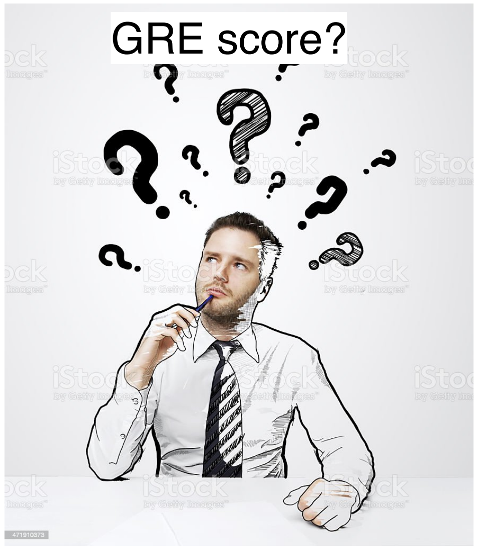
Human Factored AI

Coursera Final Project 07/10/2022

Taeho Lee

Empathize

While mentoring students who are interested in applying to US graduate school, I have got so many questions whether they can get accepted to this school with their score.



Can I get an admission from this school?

Point of View (POV) (User, Need, Insight)

Of course, the graduate school admission is not solely decided by their GRE scores or GPA, but I got interested **what if I could help students to predict and give them some sense of which school they can try to apply or not from their test results?**

Ideate / Prototype

Data collection

1. I decided to focus on applying to **US school physics department application** at this time since I am a physics student, and I wanted to try out my test result that I used for my grad school application to check if the code makes sense.

2. I collected all students information gender, race, domestic/international, GRE test scores (Verbal, Quantitative, Writing, Physics), gpa from the physics graduate school application community (https://physicsgre.com/viewtopic.php?f=3&t=145205)

3. Based on their application results, I made an excel sheet with their test results and their application results(Admitted : 1, No-admission : 0) for each school that they applied.

4. I classified the rank of the schools from 1 to 5 based on the physics ranking from US news report. (https://www.usnews.com/best-graduate-schools/top-science-schools/physics-rankings) if the school is ranked between 1 - 20, I marked as “1” in the rank section, if 21-40, I marked as “2”.

After the data collection, I made a plan to use **keras tool in the tensorflow library** in python since I can make deep learning model easily with keras.

Ideate / Prototype

Process to get to the final version

1. Worked with pandas data frame by removing gender, race, international/domestic information so that I can only use columns GRE quantitative, GRE verbal, GRE writing, GRE physics, gpa, school rank (“xdata”) as training data to train using deep learning model.

2. Created a deep learning model after making multiple nodes with Keras in the tensorflow.

3. Compiled the model

4. Trained the model 10000 times

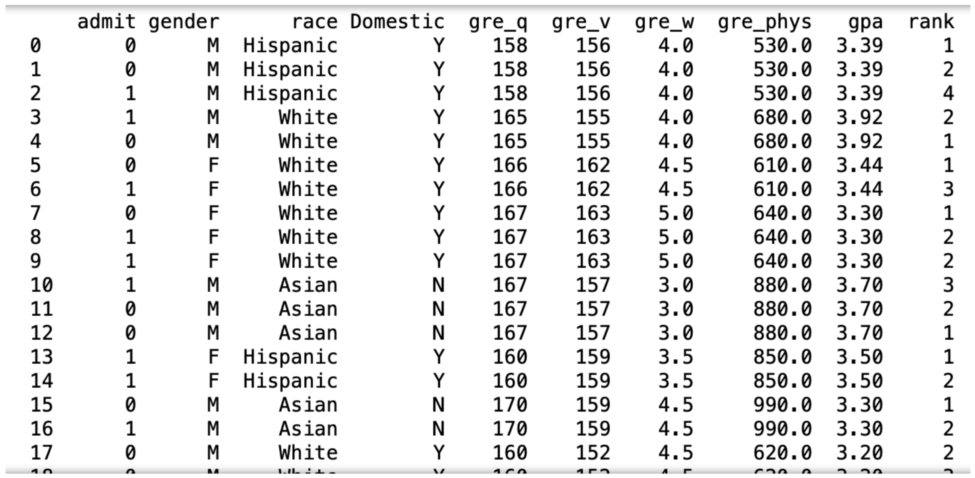
5. Tested the model with my actual scores that I used for my application to verify if the result makes sense, and also tried some random scores to see the possibility of getting in to the school with a certain rank.

Test

1. Worked with pandas data frame by removing gender, race, international/domestic information so that I can only use columns GRE quantitative, GRE verbal, GRE writing, GRE physics, gpa, school rank (“xdata”) as training data to train using deep learning model.

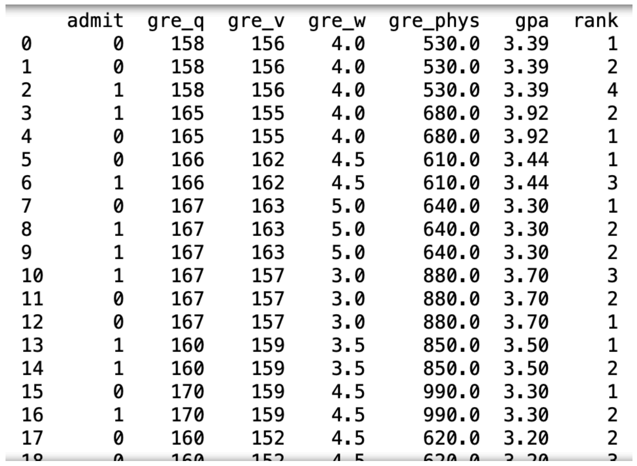
```python
import pandas as pd #import pandas library to use excel file data
import matplotlib.pyplot as plt #import matplotlib to create a plot at the final.

df = pd.read_csv('physicsgrescore2019_rev.csv') #read student information and score information
df1 = df[['admit','gre_q','gre_v','gre_w','gre_phys','gpa','rank']]
#only extract information about 'admit','gre_q','gre_w','gre_phys','gpa','rank'
df1 = df1.dropna() #preprocessing just make sure to remove any row with null values.

print(df) #print df
print(df1) #print df1
```

df



```
    admit gender     race Domestic  gre_q  gre_v  gre_w  gre_phys   gpa  rank
0       0      M Hispanic        Y    158    156    4.0     530.0  3.39     1
1       0      M Hispanic        Y    158    156    4.0     530.0  3.39     2
2       1      M Hispanic        Y    158    156    4.0     530.0  3.39     4
3       1      M    White        Y    165    155    4.0     680.0  3.92     2
4       0      M    White        Y    165    155    4.0     680.0  3.92     1
5       0      F    White        Y    166    162    4.5     610.0  3.44     1
6       1      F    White        Y    166    162    4.5     610.0  3.44     3
7       0      F    White        Y    167    163    5.0     640.0  3.30     1
8       1      F    White        Y    167    163    5.0     640.0  3.30     2
9       1      F    White        Y    167    163    5.0     640.0  3.30     2
10      1      M    Asian        N    167    157    3.0     880.0  3.70     3
11      0      M    Asian        N    167    157    3.0     880.0  3.70     2
12      0      M    Asian        N    167    157    3.0     880.0  3.70     1
13      1      F Hispanic        Y    160    159    3.5     850.0  3.50     1
14      1      F Hispanic        Y    160    159    3.5     850.0  3.50     2
15      0      M    Asian        N    170    159    4.5     990.0  3.30     1
16      1      M    Asian        N    170    159    4.5     990.0  3.30     2
17      0      M    White        Y    160    152    4.5     620.0  3.20     2
```

df1



```
    admit  gre_q  gre_v  gre_w  gre_phys   gpa  rank
0       0    158    156    4.0     530.0  3.39     1
1       0    158    156    4.0     530.0  3.39     2
2       1    158    156    4.0     530.0  3.39     4
3       1    165    155    4.0     680.0  3.92     2
4       0    165    155    4.0     680.0  3.92     1
5       0    166    162    4.5     610.0  3.44     1
6       1    166    162    4.5     610.0  3.44     3
7       0    167    163    5.0     640.0  3.30     1
8       1    167    163    5.0     640.0  3.30     2
9       1    167    163    5.0     640.0  3.30     2
10      1    167    157    3.0     880.0  3.70     3
11      0    167    157    3.0     880.0  3.70     2
12      0    167    157    3.0     880.0  3.70     1
13      1    160    159    3.5     850.0  3.50     1
14      1    160    159    3.5     850.0  3.50     2
15      0    170    159    4.5     990.0  3.30     1
16      1    170    159    4.5     990.0  3.30     2
17      0    160    152    4.5     620.0  3.20     2
```

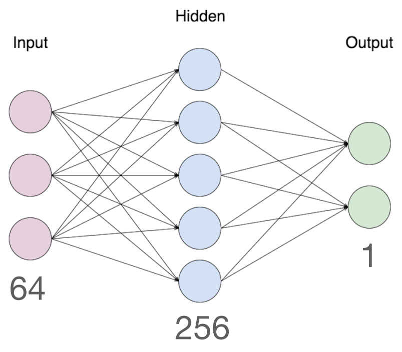
Test

2. Created a deep learning model after making multiple nodes with Keras in the tensorflow,

```python
ydata = df1['admit'].values #These are the answers for the training set.
xdata = [] #make blank list to append x_list later from the following for loop.

for i, rows in df1.iterrows(): #This is to make list in each row.
    xdata.append([rows['gre_q'],rows['gre_v'],rows['gre_w'],rows['gre_phys'],rows['gpa'],rows['rank']])
#append each row information in xdata list
import numpy as np #import numpy library
import tensorflow as tf #import tensorflow library

model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, activation='tanh'), #deep learning hidden layer with multiple node, first layer
    # second parameter in Dense is activation, tanh is used.
    tf.keras.layers.Dense(256, activation = 'tanh'), #number inside shows number of nodes.
    tf.keras.layers.Dense(1, activation ='sigmoid'), #Last node number should be 1, since we are predicting the
    #sigmoid is a activation function that gives you probability value between 0 and 1
]) #To create deep learning model
#keras is a tool in tensorflow, and we can easily make deep learning model.
```

Sequential provides training features

Hyperbolic model: tanh
(It is known that “tanh” is normally preferred and gives better result than sigmoid)

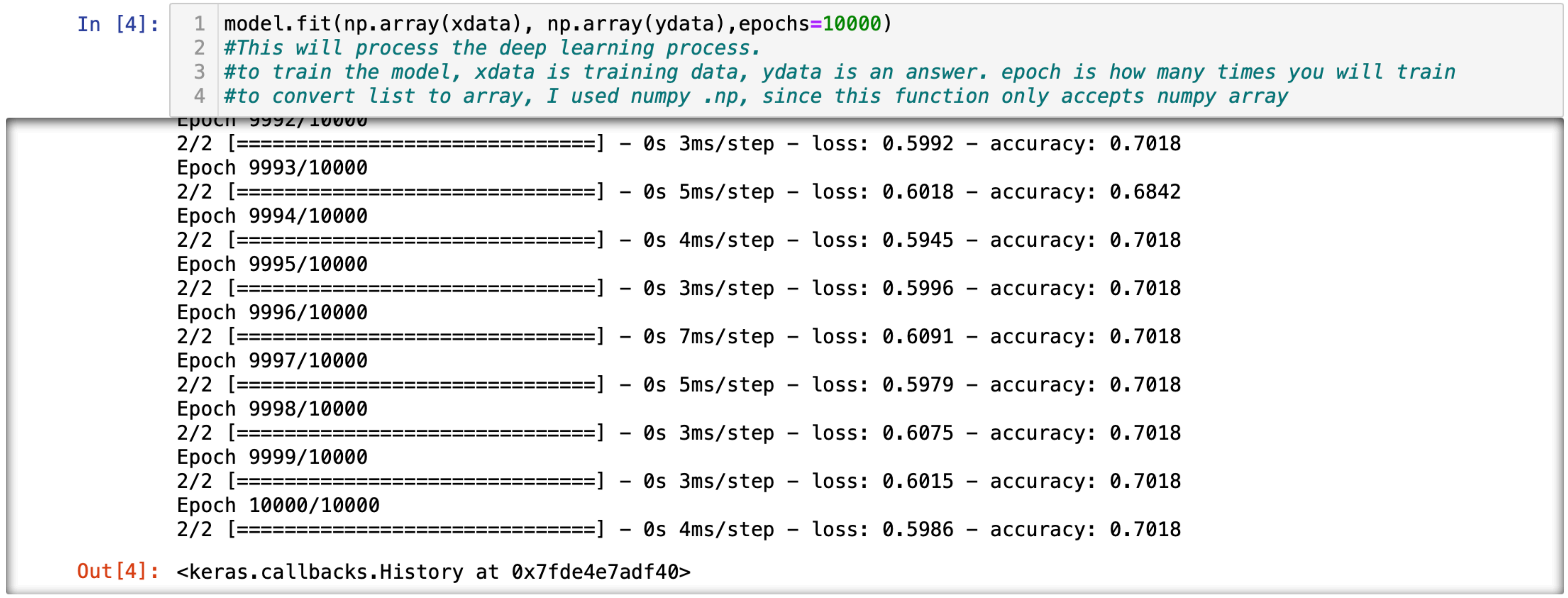


3. Compiled the model

```python
model.compile(optimizer='adam', loss='binary_crossentropy', metrics = ['accuracy'])
#model will be completed after compiling. Therefore we need compile process.
#Adam optimization gets adaptive estimation of first-order and second-order moments.
#If you want to get the probability, the binary_crossentropy is the best option because it is from 0 to 1
```

Test

4. Trained the model 10000 times



```
In [4]:
model.fit(np.array(xdata), np.array(ydata),epochs=10000)
#This will process the deep learning process.
#to train the model, xdata is training data, ydata is an answer. epoch is how many times you will train
#to convert list to array, I used numpy .np, since this function only accepts numpy array
```

```
Epoch 9992/10000
2/2 [==============================] - 0s 3ms/step - loss: 0.5992 - accuracy: 0.7018
Epoch 9993/10000
2/2 [==============================] - 0s 5ms/step - loss: 0.6018 - accuracy: 0.6842
Epoch 9994/10000
2/2 [==============================] - 0s 4ms/step - loss: 0.5945 - accuracy: 0.7018
Epoch 9995/10000
2/2 [==============================] - 0s 3ms/step - loss: 0.5996 - accuracy: 0.7018
Epoch 9996/10000
2/2 [==============================] - 0s 7ms/step - loss: 0.6091 - accuracy: 0.7018
Epoch 9997/10000
2/2 [==============================] - 0s 5ms/step - loss: 0.5979 - accuracy: 0.7018
Epoch 9998/10000
2/2 [==============================] - 0s 3ms/step - loss: 0.6075 - accuracy: 0.7018
Epoch 9999/10000
2/2 [==============================] - 0s 3ms/step - loss: 0.6015 - accuracy: 0.7018
Epoch 10000/10000
2/2 [==============================] - 0s 4ms/step - loss: 0.5986 - accuracy: 0.7018

Out[4]: <keras.callbacks.History at 0x7fde4e7adf40>
```

After training the model for 10000 times, the accuracy became 0.7018 for the prediction

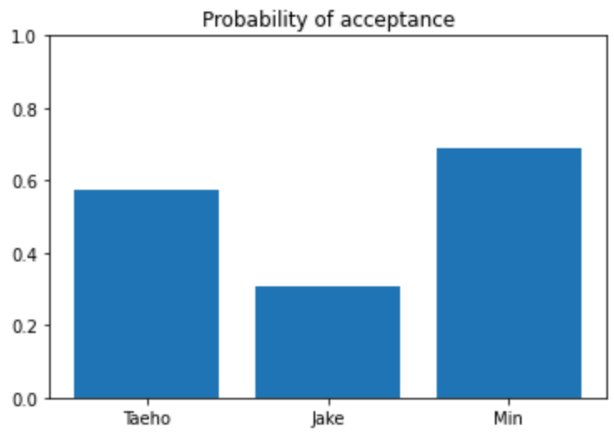
Test

5. Tested the model with my actual scores that I used for my application to verify if the result makes sense, and also tried some random scores to see the possibility of getting in to the school with a certain rank.

In [7]:
```python
#prediction

y = model.predict([[164, 149, 4.0, 830, 3.7, 3],[167, 143, 4.5, 700, 3.7, 1],[170, 167, 5.0, 990, 3.8, 1]])
print(y)

y_list = y.tolist()
y_list_new = [i[0] for i in y_list]
print(y_list_new)

students = ['Taeho','Jake','Min']
plt.ylim(bottom = 0, top=1)
plt.bar(students, y_list_new)
plt.title("Probability of acceptance")
```

```
[[0.5725241 ]
 [0.3054558 ]
 [0.68979716]]
[0.5725240707397461, 0.3054558038711548, 0.6897971630096436]
```

Out[7]:
```
Text(0.5, 1.0, 'Probability of acceptance')
```



(Taeho)
GRE Q : 164
GRE V : 149
GRE W: 4.0
GRE Phy: 830
GPA: 3.7
Rank : 3
Probability of getting accepted: 57.2 %

(Jake)
GRE Q : 167
GRE V : 143
GRE W: 4.5
GRE Phy: 700
GPA: 3.7
Rank : 1
Probability of getting accepted: 30.5 %

(Min)
GRE Q : 170
GRE V : 167
GRE W: 5.0
GRE Phy: 990
GPA: 3.8
Rank : 1
Probability of getting accepted: 68.9 %

Difficulties

Getting a right value for the number of nodes was challenging to get the most reasonable result.
I tried several values by actually changing the number of nodes and the number of training (number of epoch)

Conclusions

- By following the human-centered design principles, I was able to **design and develop an AI model to predict the probability of getting accepted to the school** within the certain rank with my current GRE test scores and GPA using deep learning technique with Keras in the tersorflow library.

- Even though graduate school application is not only determined by the test scores, I expect that this will help students who are trying to apply for us physics graduate school. This code will give students some sense of whether they can get admission from the school.

- Accuracy will be improved if I could increase the number of the data sets by collecting more data from the website.

- I learned that a lot of different factors affect to the result (the number of nodes, the number of training (epoch), the model for compiling), and we are the one who has to decide which one to use to optimize the result.

Privacy / Ethical considerations

- To get rid of bias issues, I did not include gender or nationality data into the AI system.
- I explained what type of data were collected, and how those are used using the quotation mark to make it clear for the privacy issues.