NIRF Rank Predictor

Roadmap to the webapp:

1. We scraped the data from NIRF ranking website https://www.nirfindia.org and converted the data into .csv format to use it for further processing and training

2. We scraped the data from years 2017,2018,2019 and 2020

3. Data from 2017 to 2019 was used for training and the 2020 data was used for testing which is our prediction

4. We are predicting ranks for three fields :
   a) Universities
   b) Overall
   c) Engineering colleges

Below we will be decribing the technical details as to , how we created the model and used it for prediction

a) Universities :

Modelling details:

- We found the data to be having ranks only from 1-100 in all years (from 2017-20)

- Initial step was just finding out the scores for each year based on the formula:

SCORE = 0.3*TLR + 0.3*RPC + 0.2*GO + 0.1*OI + 0.1*PPN

- We stored each year's scores in some variables and started using it for some data exploration.

- Then we calculated arrays in which , we stored the increase in scores for consecutive years , in between the years 2017 to 2019

- Then we used these arrays to find the RMS increase in score .

- So we added this RMS increase array with the scores calculated for year 2019 and used it for training

- By following this method , we achived an accuracy of 98% for testing year 2020's data

- Then we tried to increase the prediction till rank 150 by using time series prediction

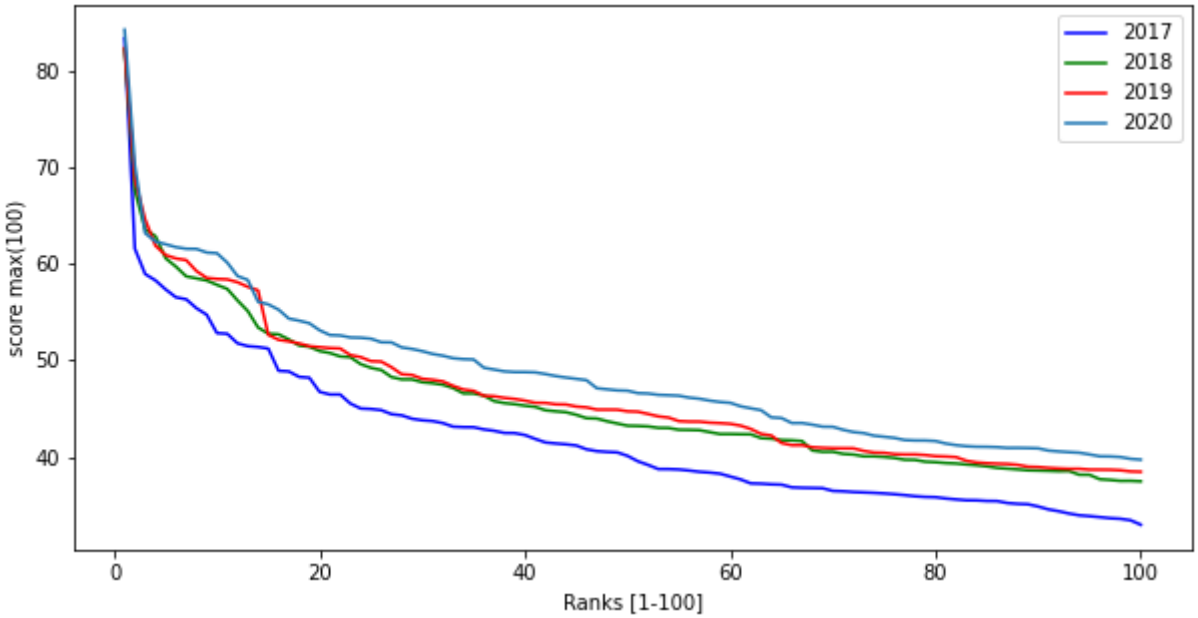
- Then we found out that that the difference betweeen scores became very less (10^{-6}) as we went above ranks 110

- We calculated the slope for the last 3 ranks and assigned it to a variable "diff"

- So , for predictions above 110 , we subtracted the last rank with the variable "diff" and went upto rank 150

- We used the polynomial regression model , with degree 5 and with no interaction

Reason for using this approach:



As the above plot is showing ,there has been a consistent improvement in the scores for ranks 1-100. So this method of finding the RMS increase in the scores worked and gave a good accuracy

We also tried simple linear models , but found it to be giving acuuracy around 84% .We even tried multiple linear regression but , again it gave less accuracy , around 78% .

b) Overall :

Modelliing details:

- We found the data to be having ranks only from 1-100 in all years (from 2017-20)
- Initial step was just finding out the scores for each year based on the formula:

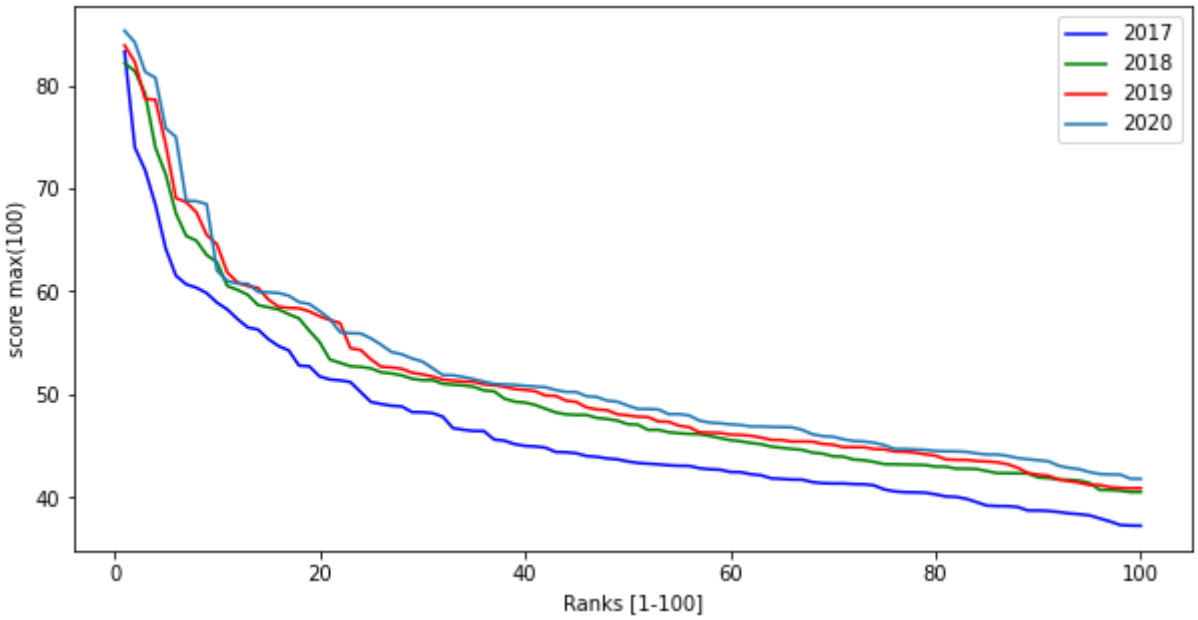
SCORE = 0.3*TLR + 0.3*RPC + 0.2*GO + 0.1*OI + 0.1*PPN

- We stored each year's scores in some variables and started using it for some data exploration.
- Then we calculated arrays in which we stored the increase in scores for consecutive years , in between the years 2017 to 2019
- Then we used these arrays to find the RMS increase in score .
- So we added this RMS increase array , with the scores calculated for year 2019 and used it for training
- By following this method , we achived an accuracy of 98.7% for testing year 2020's data
- Then we tried to increase the prediction till rank 150 by using time series prediction
- Then we found out that that the difference betweeen scores became very less (10^{-6}) as we went above ranks 110
- We calculated the slope for the last 3 ranks and assigned it to a variable "diff"

- So , for predictions above 110 , we subtracted the last rank with the variable “diff” and went upto rank 150

- We used the polynomial regression model , with degree 5 and with no interaction

Reason for using this approach:



As the above plot is showing ,there has been a consistent improvement in the scores through the years for ranks 1-100 . So this method of finding the RMS increase in the scores worked and gave a good accuracy

We also tried simple linear models , but found it to be giving acuuracy around 85% .We even tried multiple linear regression but , again it gave less accuracy , around 76% .

c) Engineering colleges :

Modelling details:

- Initial step was normalizing the data , ie. we divided each column in the parameters by the highest value of that parameter in that particular year

- Then we averaged the data from all the training years and used this averaged data for training .

- Then we calculated scores for the averaged data by the following formula

SCORE = 0.3*TLR + 0.3*RPC + 0.2*GO + 0.1*OI + 0.1*PPN
We used the computed score for training.

- Since , we used normalized data for training , we had to also normlize the testing data . But for predictions , the maximum value of each parameter of the next year is not known , so we assumed the next year's maximum to be the maximum of previous year , so we used the previous year's data for normalization

- After normalization of each parameter for testing ,we apply the same formula to compute the score and pass it to the trained model for prediction of the rank

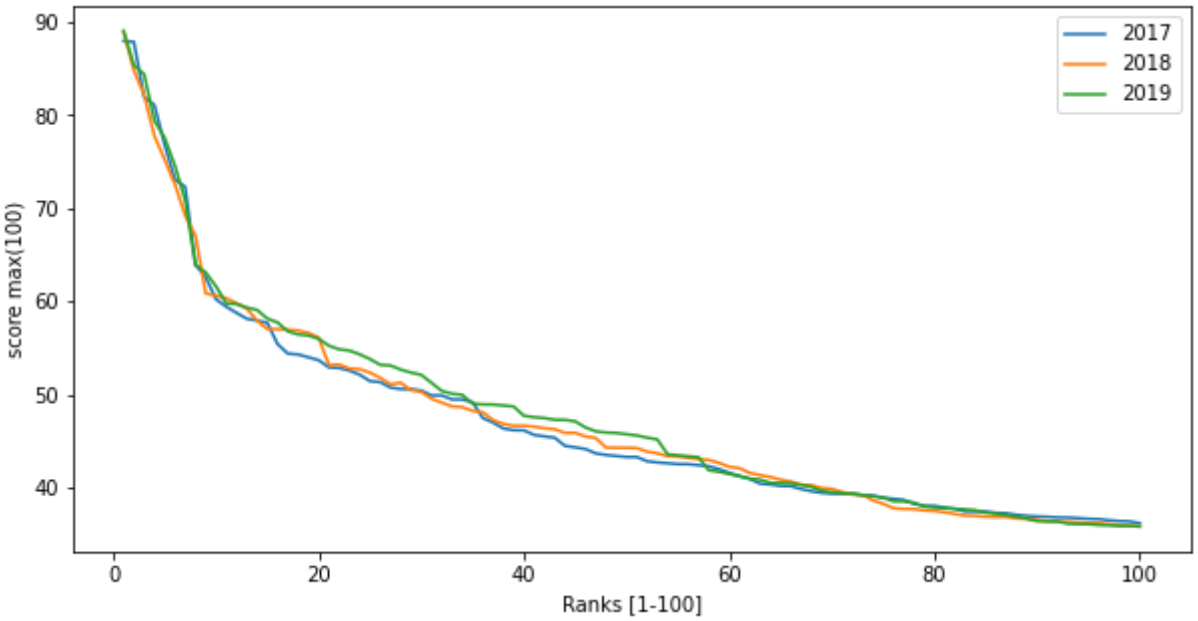
- We trained the model using polynomial regression model , with degree 3 , and without any interaction

- We had data for rank 1-100 in 2017 and 2018 , and for ranks 1-200 from year 2019 , but we used the data till rank 169

- Since we had data above 100 only from year 2019, the accuracy above 100 for prediction will become comparitively low

- So ,while predicting, we get almost accurate results until rank 140 , and we stopped there because our college rank was 132 and ranks above that were useless for us

Reason for averaging the data for training :



As we look at the above image , the data varies very less , and we can predict with 94% accuracy for ranks until 100 by averaging method. But since our training data for ranks above 100 are only from 2019 year , the ranks above 100 are a little innaccurate .

We also tried simple linear models , but found it to be giving acuuracy around 84% .We even tried multiple linear regression but , again it gave less accuracy , around 75% .

Instruction set to use the webapp:

1. Index page of the webapp consists of three links , each linking to Engineering , University and Overall accordingly.

2. In all prediction pages , the user has to give the valid scores (1-100)for TLR , RPC , GO , OI , PPN.

3. The user input values are collected at the flask server side and are predicted .

4. The predicted ranks are then rendered to their respective pages .

5. Whenever the user gives the scores , these scores are dynamically added into the table along with the rank , so that the user can tweak the values accordingly.

Requirements :

1. numpy == 1.18.1
2. Flask == 1.1.1
3. pandas == 1.0.1
4. scikit-learn == 0.23.1
5. gunicorn == 20.0.4