Assignment 4 - Main Problem

Divyanshi Kamra, Naman Agrawal, Tushar Nandy, Prasann Viswanathan

June 2, 2020

Abstract

I'd like to start by saying that we had completely forgotten that the submission was 12 noon and not 12 midnight so thank you Sanju for that reminder message or else we'd be completely f***ed right now. In this article, we will briefly study what linear regression is and how it can be implemented for multiple variables using Scikit-Learn, which is one of the most popular machine learning libraries for Python.

$$y_{fit} = m1x1 + m2x2 + m3x3... + v \tag{1}$$

1 How We Approached

Since this problem was fairly straight forward it did not require much discussion and deliberation. On the 15_{th}, we started by doing singular research on the problem through various sources looking to find the best ones and we shared those on our team group, hence boosting everyone's knowledge on regression and how it's applied. [1] We all obtained max PH117 feels from whatever we gathered (Nightmares of photoelectric effect ensue) and realized this is just line fitting to get the minimum error.[2] We learnt important terms such as test set, train set, actual and predicted values graph and also the various errors that occur. As a result of this newfound knowledge, performing both parts of this assignment were greatly simplified and required minimum effort (and maximum interest). [3]

2 Bugs Faced

As far as I'm concerned i think the majority of my time was spent figuring out how to read a csv file without a constant error message popping out. I remember how Sanju saved the "Chance of Admit" column as "Chance of Admit ", and i spent quite some time (More than what I can say without embarrassing myself) on that issue.
This is important, add an 'r' before the file path, this helps the program read the file. It's easy to miss.
Apart from this what stressed us out was why is the error percentage so large (Around 9*percent* error seems like a lot, doesn't it?) On further research did we come to realise that the error depended on various factors and that it was okay to have a seemingly large error. This is explained in the conclusion.

3 Why We Settled For This Model

Why does anyone settle for any programming model? Efficiency? Simplicity? Same for us. Before this we had tried linear regression from the absolute basics as shown by our Assignment4.1, however the math was tedious to apply from scratch; AND if python offers a vast set of libraries for our use and ease; why not use them? Also I'd like to add that application of this model was made extremely easy thanks to the source that we referred. [4]

4 Important Snippets of Code

Here we've included the code snippets that in particular produce some visual output, as the rest just consisted of package calling and calculations. However, if you're interested the entire code is available right here [5]. Apart from the code, we've placed each corresponding graph here too which will give a visual for you to better understand what s going on. Here I'd like to mention that we ran the program thrice as

1. Train and test on the data provided by our mentor.
2. Train 80 perc on ours test 20 perc on our csv and theirs.
3. Train 100 perc on ours and test on theirs.

4. Train 80 perc on our and their data, test on our csv.

Here's the code:

```
#This helps read the csv file and separate the attribute variables from
#label variables, attributes are in X and labels in Y
df = pd.read_csv(r"C:\Users\HOME\Desktop\IntelligentAgents
"\Admission_Predict_LA.csv")
df.drop(columns=['Serial No.'], inplace=True)
headers = list(df.columns)
target = headers.pop()
X = df[headers].values
y = df[target].values

plt.figure(figsize=(15,10))
plt.tight_layout()
seabornInstance.distplot(df[target])
```

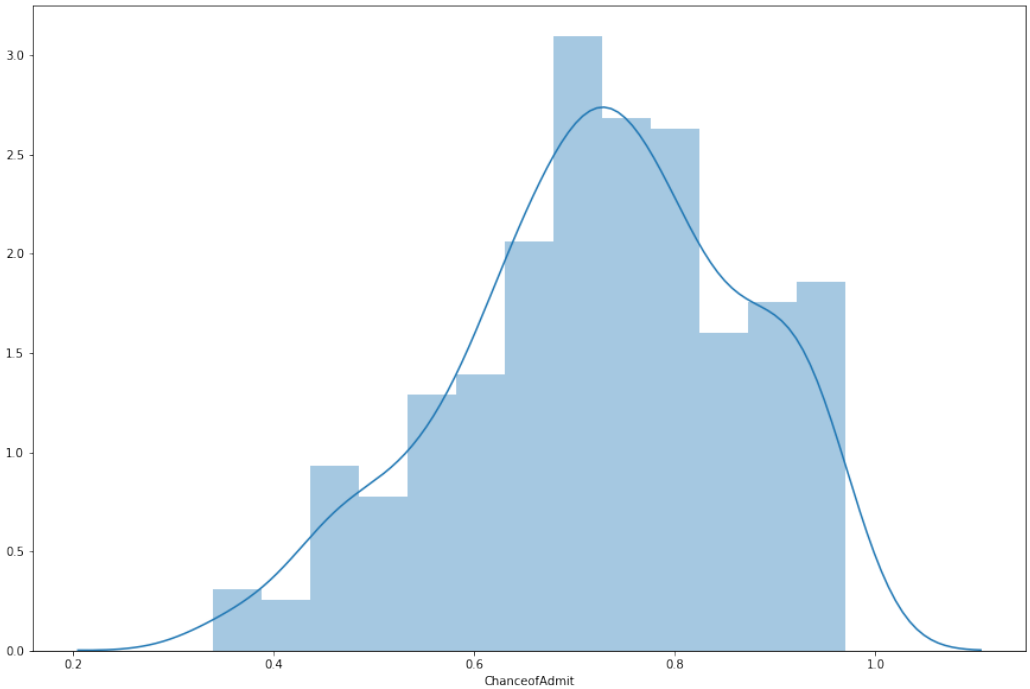

Figure 1: Distance plot to help visualize mean

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
#0.2 is an arbitrary test size,
#you can play around with this value to get desired results.

regressor = LinearRegression()
regressor.fit(X_train, y_train)

coeff_df = pd.DataFrame(regressor.coef_, headers, columns=['Coefficient'])
print(coeff_df)

y_pred = regressor.predict(X_test)
```

	Coefficient
GREScore	0.002071
TOEFLScore	0.001955
UniversityRating	0.006572
SOP	-0.004728
LOR	0.021184
CGPA	0.124632
Research	0.024144

Figure 2: Coefficient of each variable(How much it affects)

We spent a good twenty seconds laughing at the coefficients received as the old saying CGPA is most important, or CGPA is like the 1 ahead of all other ones and zeroes blah (Classic first day convocation hall speech), turns out to be true. CGPA has the highest effect on your chance of admission. Enough incentive to turn maggu? xD. (Note that SOP is least valued (hard luck to those with good writing skills), in fact has a negative coefficient xD)

```
show = final.head(25)
show.plot(kind='bar',figsize=(16,10))
plt.grid(which='major', linestyle='-', linewidth='0.5', color='green')
plt.grid(which='minor', linestyle=':', linewidth='0.5', color='black')
plt.show()
#Note that although there were 20percent of 400 (80) test cases,
#we are graphing only 25 as 80 would make the graph a mess
```

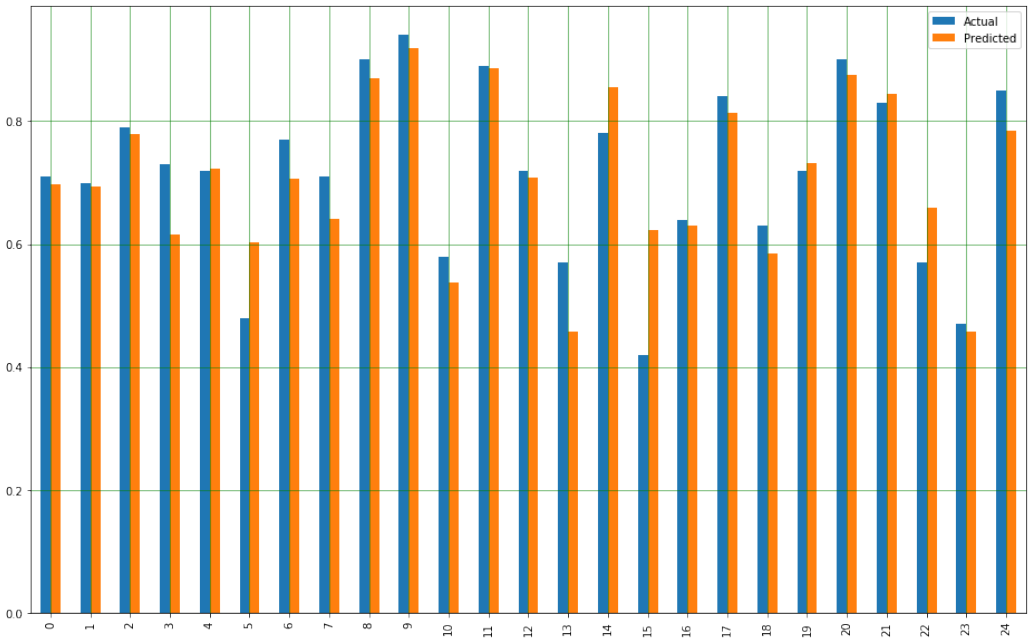

Figure 3: Data of actual and predicted values shown in a bar graph

```
#Errors
MAE = metrics.mean_absolute_error(y_test, y_pred)
RMSE = np.sqrt(metrics.mean_squared_error(y_test, y_pred))
print(f"Mean Absolute Error: {MAE}")
print(f"Root Mean Squared Error: {RMSE}")
print(np.mean(y))
print(f"Error % is {RMSE/(np.mean(y)) * 100}")
```

```
Mean Absolute Error: 0.050087548438179 53
Root Mean Squared Error: 0.069319236650335 42
0.72435
Error % is 9.569853889740514
```

Figure 4: Errors that we calculated

5 Making Sense of Some Trends

After observing nyc's project we've realised that for better understanding of why each coefficient's value is as we've obtained, we need to look at the way each attribute independently affects the chance of admission. Hence here we're showing the graphs as well as mentioning our understanding of the trends.

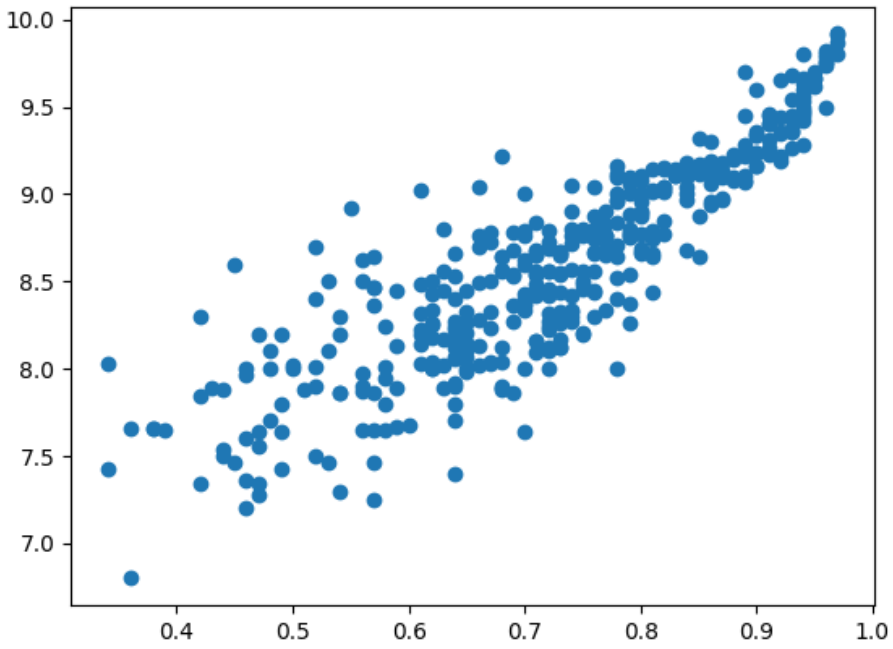

Figure 5: CGPA trends

As expected CGPA has the highest effect. We've done an extensive report on CGPAs trends in Assignment4.1 including heat maps and plotting the best fit line as well. Please be sure to check that too.

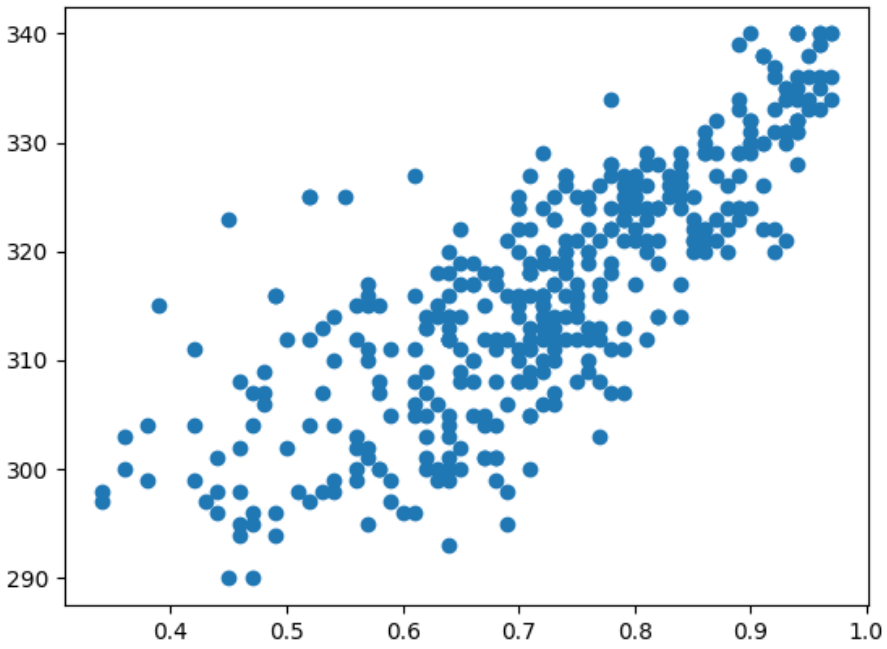

Figure 6: GRE trends

GRE has a trend similar to CGPA, however it is a bit more spread out; i.e to say that for the same chance of admission there's a wide range of GRE scores that could be possible.

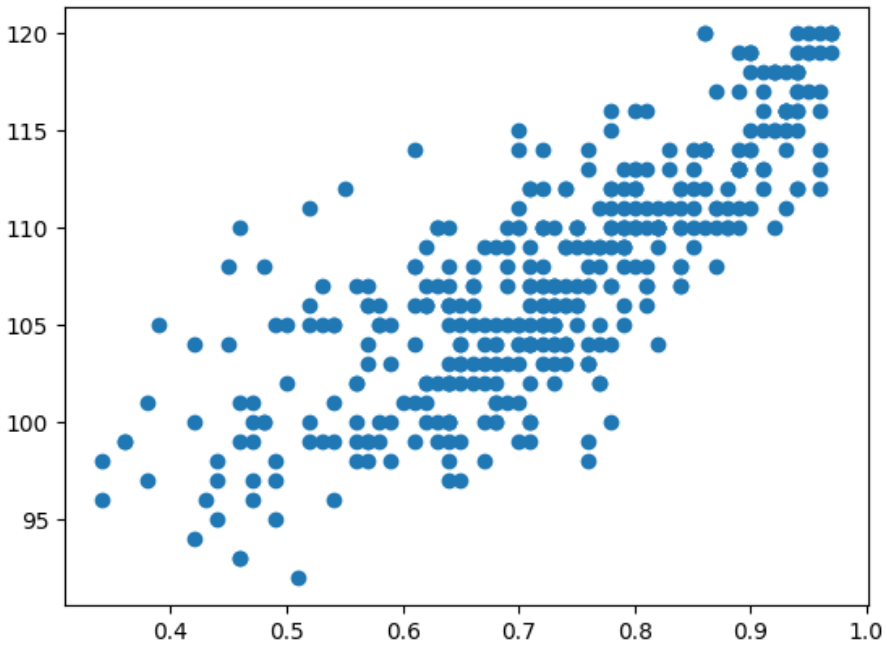

Figure 7: TOEFL trends

TOEFL trends are very similar to GRE score trends, in fact an observation could be made here that attributes with a wider range (score from 1-100) seem to better plot chance of admission. This may or may not be true but that's what we felt on face level.

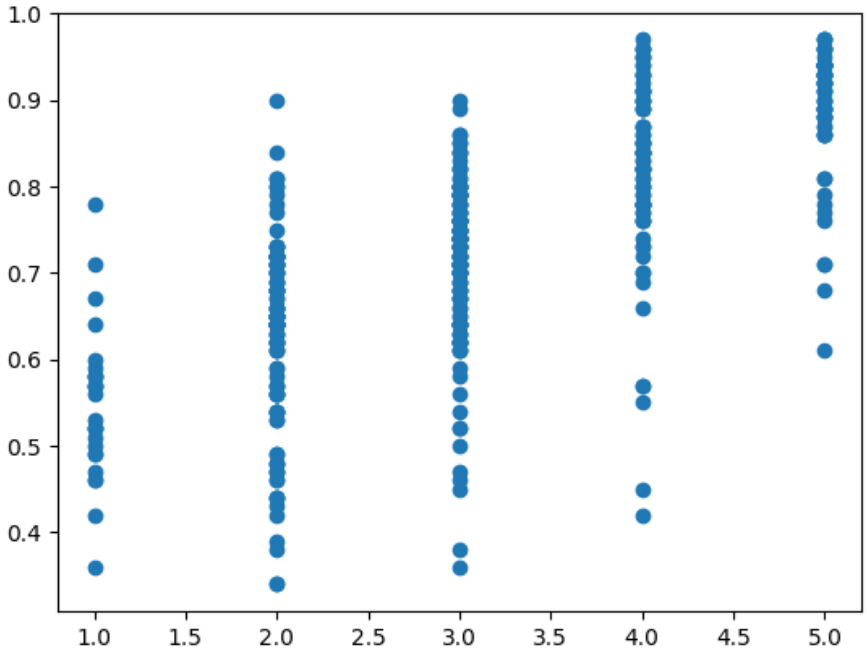

Figure 8: University Rating trends

It's surprising to note how universities with 5 rating have astoundingly high chances of admission (Is IITB 5 rated? I really need to know xD) They probably account for Ivy League institutes so the trend explains itself. On face level it seems that only 5 rated ones have a huge edge compared to their 1-4 counterparts.

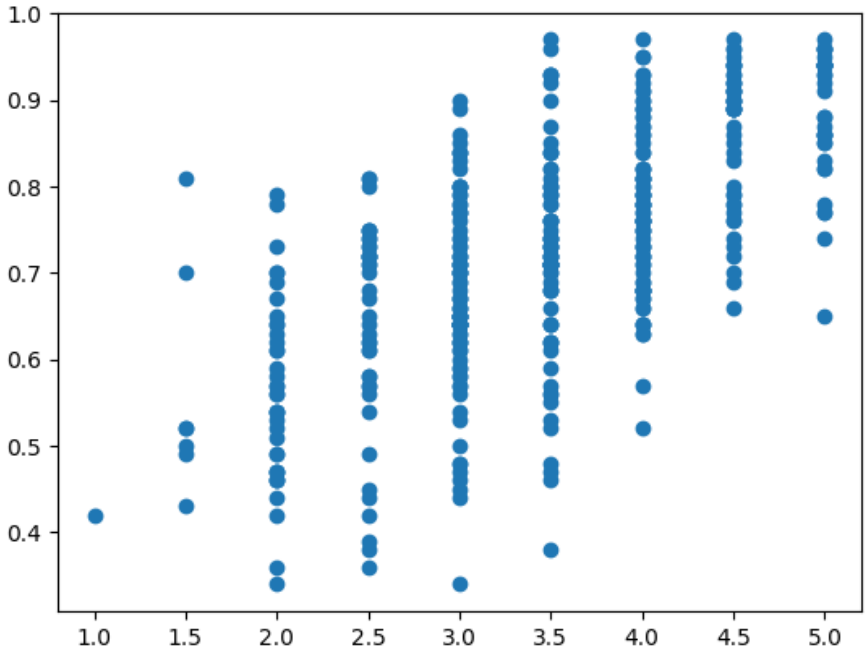

Figure 9: LOR trends

First of all, the complete lack of extremely low scores on the LOR is straight forward; no institute would write a bad LOR on purpose right? They would want their students to pursue further ed. from the best institutes. However, the discrepancy in scores is probably because they can't lie on their LOR. Hence we see a majority of LOR scores between 3-4. LOR also fairly exhibits an upward moving trend.

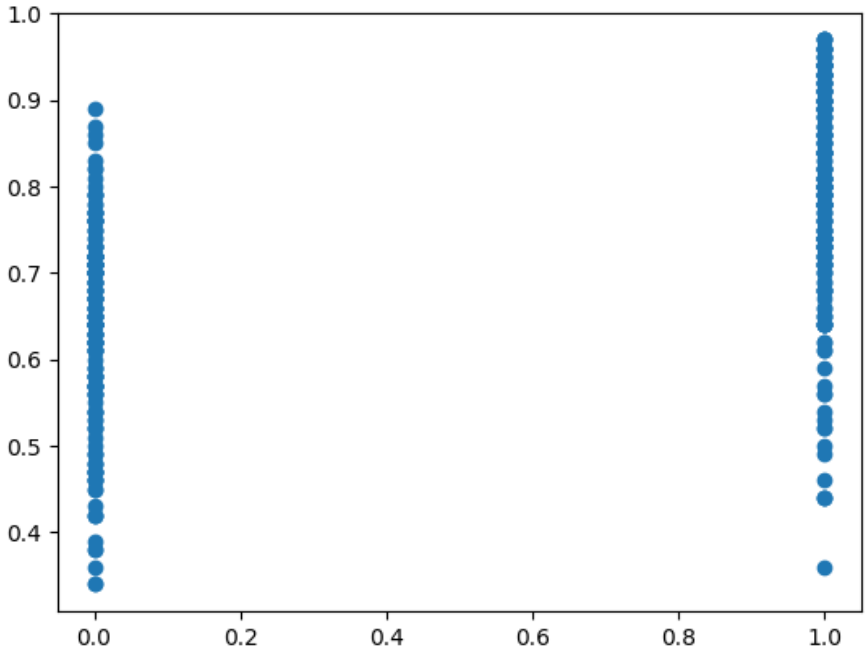

Figure 10: Research trends

Research doesn't plot Chance of admission well at all. However it is safe to assume that having done some research does boost your chances of admission over counterparts with similar CGPAs and TOEFL, GRE scores. WE also observe a majority of points above 0.65 probability among those who have pursued research.

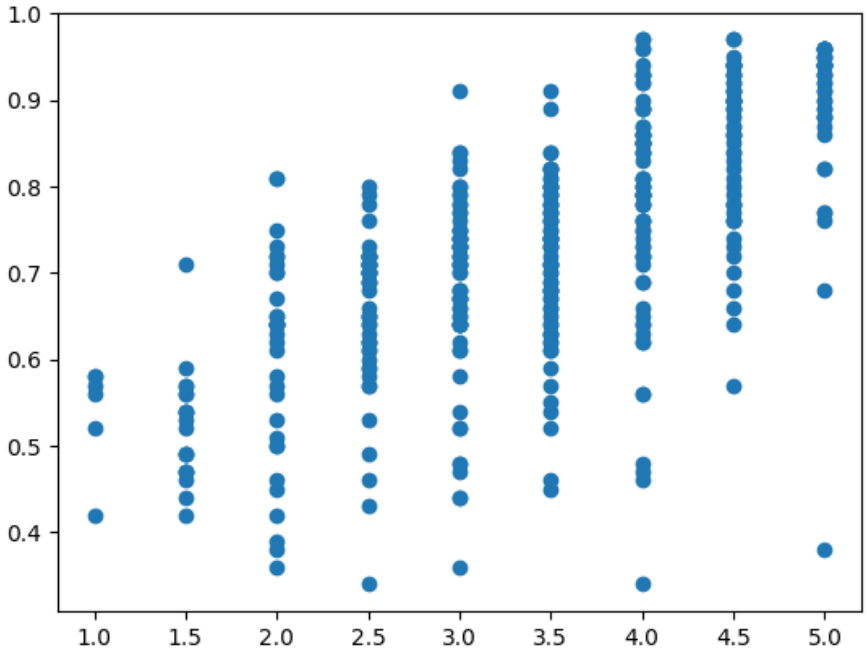

Figure 11: LOR trends

Here comes the attribute we all dreaded xD. In fact on face level SOP seems to properly map chance of admission, as one would expect. But then why the negative coefficient?
This is very well explained by our teammate Tushar-

"When the SOP dilemma occurred, I was blown out of my mind. I thought I failed as an analyst, but I think I may have come with something okay-ish. I arranged the data in ascending order of Chances and calculated the coefficients for 1st 70percent of people(after sorting based on CGPA); This gave me a positive coefficient for $SOP = +0.000896$. In the same run, the max coefficient belonged to $CGPA = 0.103356$ (Note the difference from the earlier result) The last indexed student in the chances column(The last student for this run) had a CGPA of: 9.0. Hence, SOP till this point is a positive factor, but still has negligible effect. Then, I ran the test for 80percent of students, with the same sorted csv. Coefficient for SOP was$=-0.000354$;

Coefficient for highest contributing factor (CGPA)=0.101128; Highest CGPA is 9.06.

"I saw almost every data point from 70percent range up til the end; and as chances increased, the CGPA increased but people also got poor SOP scores.

"My regressor is a fool.

"Since it is linear regression, it has evaluated coefficients for each factor independently. Because, the sop scores fall drastically sometimes even with high chances of admission, it sort of imposes a penalty for a high SOP score.

"Now, something I want to recall: I remember seeing a good enough streamlining and clustering in the heat map of CGPA when chances increased (towards higher CGPAs) but the spread was very erratic and 'ugly' at lower CGPAs. There were some people with splendid chances also who got a low CGPA.

"Eureka!(?) xD

"When your CGPA isn't good, you make a good SOP for compensation but when your CGPA is sky-rocketing, who cares about a damn SOP?

"The independent assignment of coefficients, I believe, is what motivated AI developers to move ahead of LR and reach neural networks, where almost every factor is inter-related and I eagerly look forward to that!"

6 How the Other Team's Data Affected Our Model

The other team's data affected our model as we'd expected. The errors received etc were fairly the same. As aforementioned we split the testing in 3 separate cases and tried all three of them using the code as it was. Here are our results in the form of graphs of actual and predicted values. Along with them we've added the errors received as well. [6]

1. case 1:

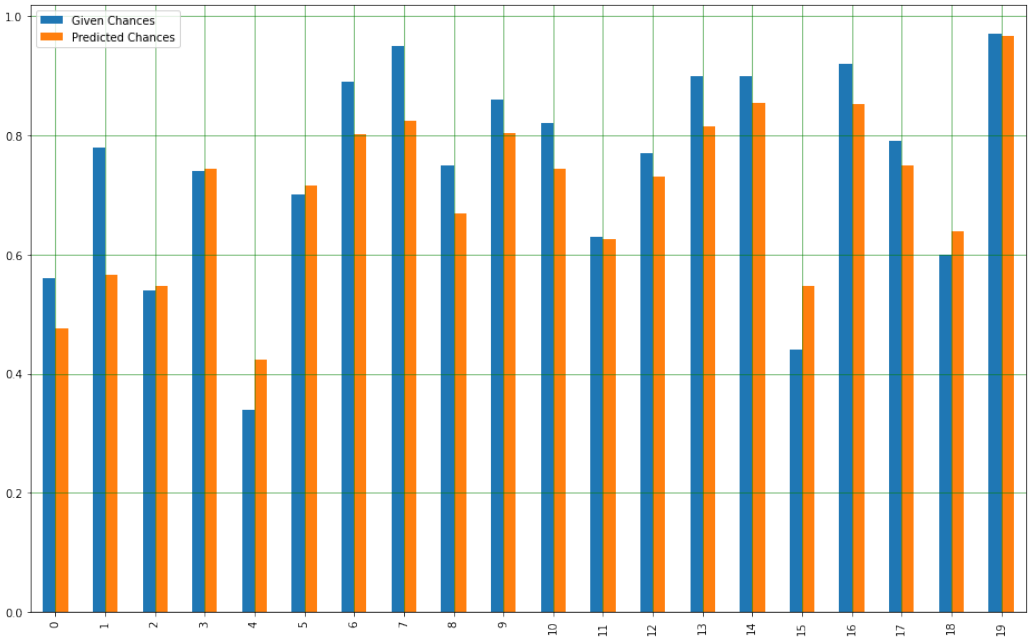

Figure 12: 1st case bar graph

	Given Chances	Predicted Chances
0	0.56	0.475375
1	0.78	0.565538
2	0.54	0.546554
3	0.74	0.743961
4	0.34	0.423650
5	0.70	0.716105
6	0.89	0.802887
7	0.95	0.825268
8	0.75	0.669407
9	0.86	0.804722
10	0.82	0.744698
11	0.63	0.625305
12	0.77	0.731019
13	0.90	0.815949
14	0.90	0.855097
15	0.44	0.547151
16	0.92	0.852013
17	0.79	0.749748
18	0.60	0.638978
19	0.97	0.967605

Mean Absolute Error: 0.06308828372290108

Root Mean Squared Error: 0.08022145066225163

0.7425

Error % is 10.804235779427827

2. case 2:

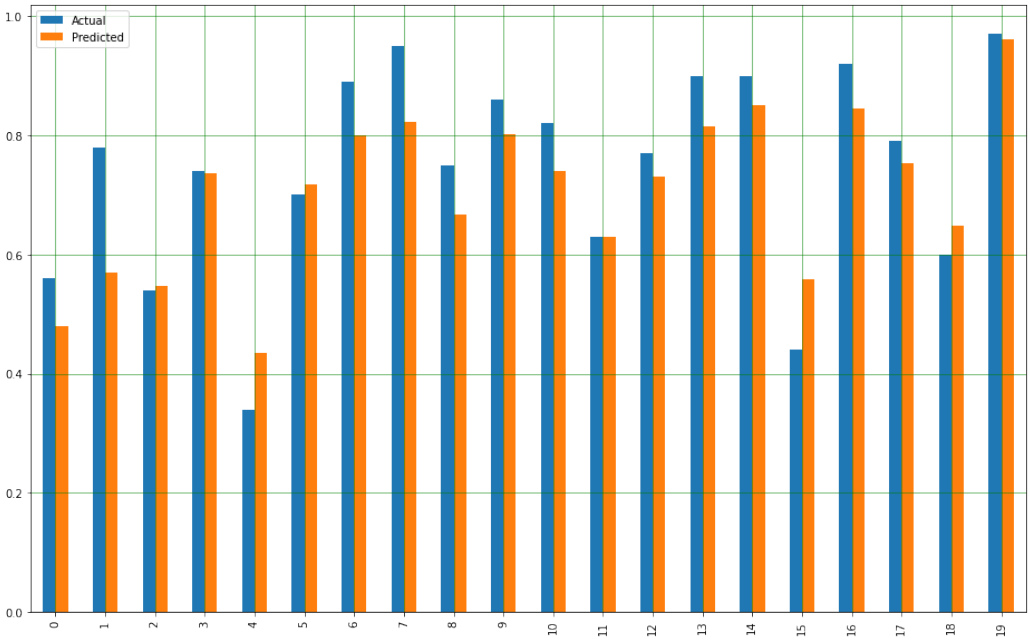

Figure 14: 2nd case bar graph

	Actual	Predicted
0	0.56	0.480193
1	0.78	0.569432
2	0.54	0.546286
3	0.74	0.736600
4	0.34	0.434654
5	0.70	0.716827
6	0.89	0.800278
7	0.95	0.823006
8	0.75	0.667406
9	0.86	0.802547
10	0.82	0.740922
11	0.63	0.630069
12	0.77	0.730843
13	0.90	0.814657
14	0.90	0.850356
15	0.44	0.558728
16	0.92	0.845556
17	0.79	0.753454
18	0.60	0.648507
19	0.97	0.960664

Mean Absolute Error: 0.065457771143795 45

Root Mean Squared Error: 0.08223446404860096

0.72435

Error % is 11.352863125367703

3. case 3:

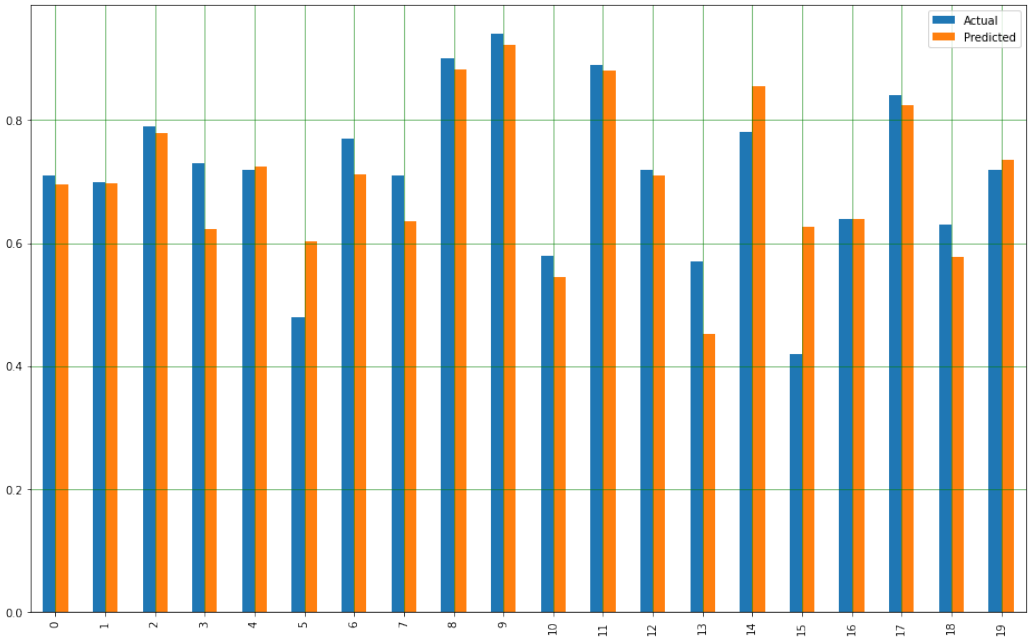

Figure 16: 3rd case bar graph

	Actual	Predicted
0	0.71	0.695304
1	0.70	0.698042
2	0.79	0.778938
3	0.73	0.623213
4	0.72	0.724316
5	0.48	0.603329
6	0.77	0.712494
7	0.71	0.634852
8	0.90	0.882798
9	0.94	0.922536
10	0.58	0.544360
11	0.89	0.881141
12	0.72	0.710205
13	0.57	0.453001
14	0.78	0.854528
15	0.42	0.626153
16	0.64	0.638810
17	0.84	0.823948
18	0.63	0.577300
19	0.72	0.735778

Mean Absolute Error: 0.048358149678442 78

Root Mean Squared Error: 0.07176409336227581

0.72435

Error % is 9.907378113104963

The errors remained around the same range i.e 9-11 percent.

7 Conclusion

We can see that our model returns around ten percent error, roughly speaking. This means that our algorithm was not very accurate but can still make reasonably good predictions. There are many factors that may have contributed to this inaccuracy, for eg:

Need more data: We need to have a huge amount of data to get the best possible prediction. Bad assumptions: We made the assumption that this data has a linear relationship, but that might not be the case. Visualizing the data may help you determine that. Poor features: The features we used may not have had a high enough correlation to the values we were trying to predict.

In this project we implemented multiple linear regression using SciKit library. I hope you enjoyed learning. Thank you for reading our presentation.

References

Reading for regression
https://en.wikipedia.org/wiki/Linear_regression

Reading for regression2
https://www.geeksforgeeks.org/ml-linear-regression/

Reading for regression3
https://towardsdatascience.com/introduction-to-machine-learning-algorithms-linear-

SciKit Learn Algo
https://towardsdatascience.com/a-beginners-guide-to-linear-regression-in-python-wi

Our code
https://github.com/tusharnandy/Team_LA/blob/master/assgn-4/Assgn4_Problem3.py

Source for graphs
https://github.com/tusharnandy/Team_LA/tree/master/assgn-4/new_test_metrics