Assignment 4 - Problem 1

Divyanshi Kamra, Naman Agrawal, Tushar Nandy, Prasann Viswanathan

April 15, 2020

Abstract

Linear Regression performs the task of predicting a dependent variable(y) based on one or more independent variables(x). Plotted in 2D, it gives a straight line which best fits the given data points (PH117 feels). Our job is to find the appropriate m and c in the following:

$$y_{fit} = mx + c \tag{1}$$

1 Our Analysis

We performed an exploratory data analysis for the first problem and stuck to using data visualization as our primary tool as much as we could. So, does CGPA really matter? (Nervous anticipation) The answer is a mix of yes and no. Universities do notice an impressive CGPA (8.5 and above according to Tushar) but that doesn't sure it for a higher CGPA nor deny it for a lower. Refer to our graphs below.[1]

2 Graphs and Data

Contrary to classic data science methods, we have used SciPy lingress method to keep it as simple as possible.[2]

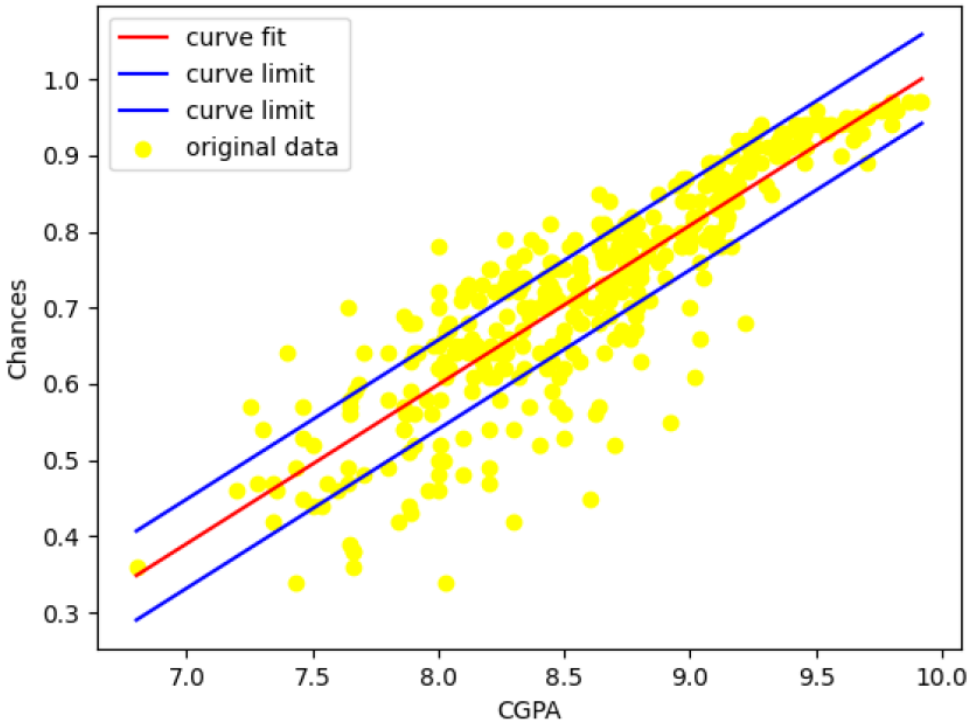

Figure 1: Linear regression best fit line

The yellow plot is the original CGPA vs Probability as provided by the csv file. The red line is the best fit line. The blue lines are at an arbitrary distance 0.05 above and below it.

Roughly around the 0.8 admisson chance, plots begin to streamline. This motivated us to validate our statement using a more beautiful plot: The heat map. Note that we multiplied the chances by 5 to fit the map better. It doesn't affect the trend and displays accurate clustering and streamlining of data points near the ends.

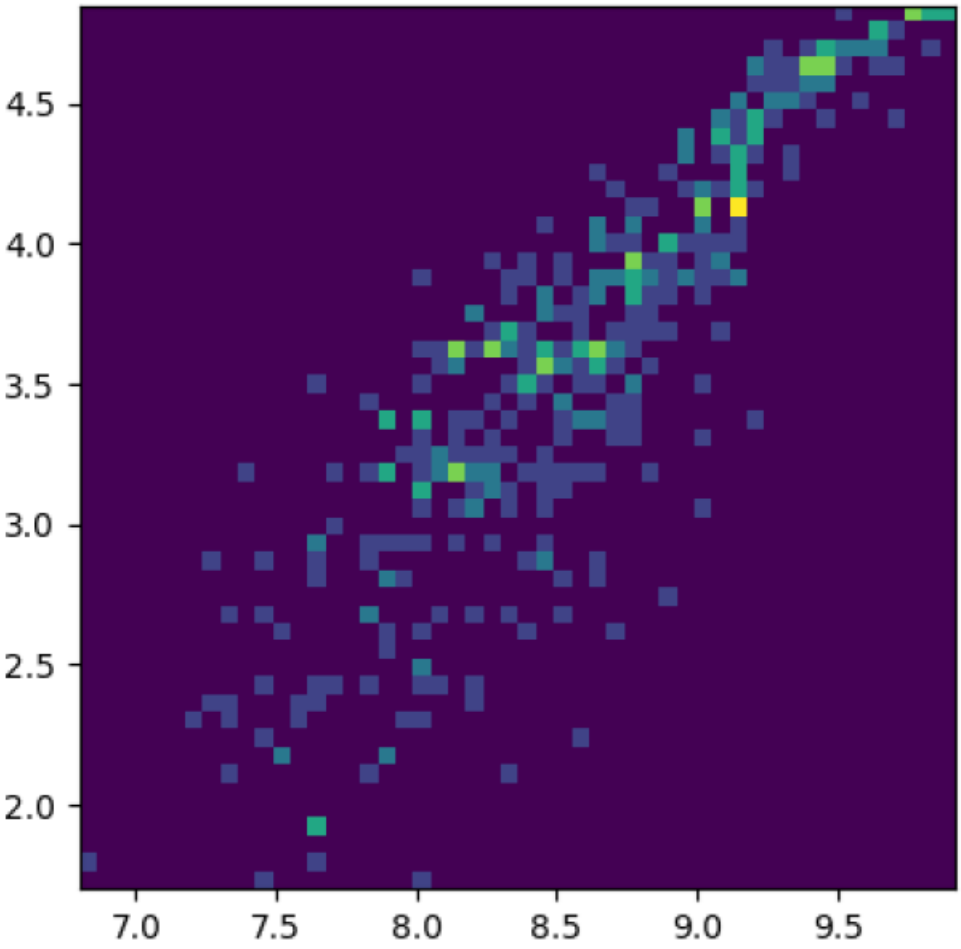

Figure 2: Heat Map

1. $m - fit = 0.20855$
2. $c - fit = -1.07151$
3. RMS error = 2.9602
4. RMS error of last 10 values = 0.02046 (less than a hundredth)

3 Code used

Please note that before running the code yourselves, sort the CGPA in ascending order and change the column name "Chance of Admit " to "ChanceofAdmit" [3]

```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
from scipy import stats
from sklearn import metrics

path = "Admission_Predict.csv"
df = pd.read_csv(path)

y = np.array(df['ChanceofAdmit'])
x = np.array(df['CGPA'])

y_end = y[390:]
print(len(x))
slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)

y_fit = slope*x + intercept
std_err *= 10
y_max= y_fit + std_err
y_min = y_fit - std_err

y_fit_end = y_fit[390:]

plt.scatter(x, y, color='yellow', label='original data')

plt.plot(x, y_fit, color='red', label='curve fit')
plt.plot(x, y_max, color='blue', label='curve limit')
plt.plot(x, y_min, color='blue', label='curve limit')

plt.xlabel('CGPA')
plt.ylabel("Chances")
plt.legend()
```

```
plt.show()

y = y*5

heatmap, xedges, yedges = np.histogram2d(x, y, bins=50)
extent = [xedges[0], xedges[-1], yedges[0], yedges[-1]]

plt.clf()
plt.imshow(heatmap.T, extent=extent, origin='lower')
plt.show()

total_MS = metrics.mean_squared_error(y, y_fit)
print(total_MS)
print(f"RMS error of the fit is: {np.sqrt(total_MS)}")

end_ms = metrics.mean_squared_error(y_end, y_fit_end)
print(f"rms error for the last 10 values: {np.sqrt(end_ms)}")
```

4 Conclusion

We've concluded that a good CGPA is a solid advantage but a below par CGPA could also be well padded with projects, LORs and a well written SOP to boost your chance to selection. Which is why data points are spread out at lower CGPAs.

References

First Source
https://towardsdatascience.com/linear-regression-using-gradient-descent-in-10-lines-of-code-642f995339c0

Second Source
https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy..linregress.html

Third Source
https://towardsdatascience.com/a-beginners-guide-to-linear-regression-in-python-with-scikit-learn-83a8f7ae2b4f