Machine Learning Homework

2.1.b

In my observation, I found CGPA is the best contributive feature, I use Pearson correlation to calculate seven columns

```
{'GRE_score': 1.0884036862476284e-117,
 'TOFEL_score': 6.729926762328514e-109,
 'University_rating': 5.866255627650183e-72,
 'SOP': 2.8859074534541132e-70,
 'LOR ': 3.069932320299405e-60,
 'CGPA': 3.396544858710999e-165,
 'Research': 3.595493545839702e-40}
```

The smaller the value, the stronger the correlation, Research is the stronger correlation. However, Research is bool value that is hard to fit continuous value so we choose CGPA.

In another observation we can find:

[18]:
```
df_X[0:2]
```

[18]:

	Serial_id	GRE_score	TOFEL_score	University_rating	SOP	LOR	CGPA	Research	Chance_of_Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76

0 and 1 have the same eigenvalues, except for CGPA, but the results are very different.

That see another example:

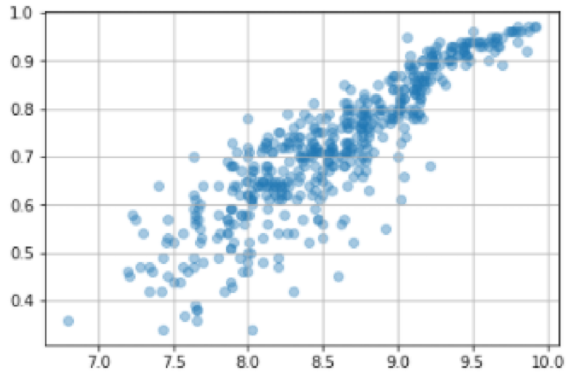
[19]:
```
df_X[5:7]
```

[19]:

	Serial_id	GRE_score	TOFEL_score	University_rating	SOP	LOR	CGPA	Research	Chance_of_Admit
5	6	330	115	5	4.5	3.0	9.34	1	0.90
6	7	321	109	3	3.0	4.0	8.20	1	0.75

We can draw the plot show CGPA and score

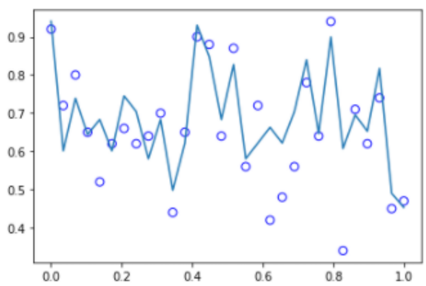
20]:
```
plt.scatter(df_X['CGPA'],df_X['Chance_of_Admit '],
            alpha = 0.4, cmap = 'Reds')
plt.grid()
```



In this figure you can see CGPA(X) is proportional to chance_of_admint(Y).

1]:
```
plt.scatter(list_x, y_test[:30], facecolor="none", edgecolor="b", s=50, label="training data")
plt.plot(list_x, y_pred[:30])
```

1]:
```
[<matplotlib.lines.Line2D at 0x1c9cac50dd8>]
```



5]:
```
print('polynomial Feature rmse result:', rmse(y_test, y_pred))
```

```
polynomial Feature rmse result: 0.06589686406416656
```

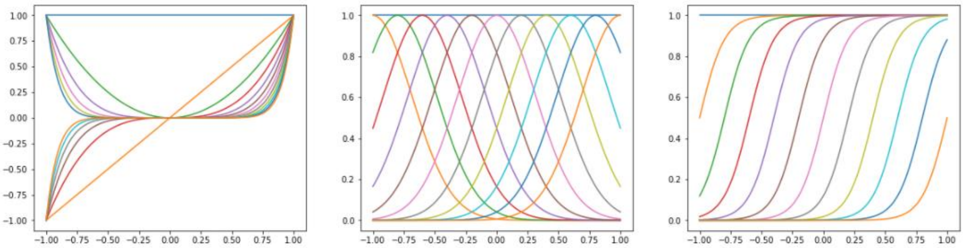
However, rms error is higher than pervious, but we just use one feature can get this performance

this result is proving CGPA is most contributive feature.

2.2.a

2.2.a result

I think polynomial could be best basisi function. I refer this, I think polynomial can fit most data

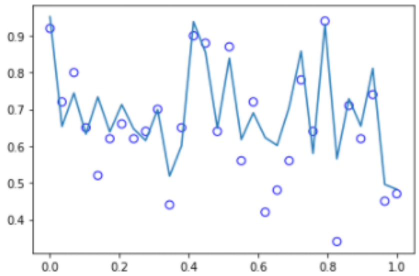


2.2.b

polynomial

```
[56]: plt.scatter(list_x, y_test[:30], facecolor="none", edgecolor="b", s=50, label="training data")
      plt.plot(list_x, y_pred[:30])
```

```
[56]: [<matplotlib.lines.Line2D at 0x1c9caf4f048>]
```

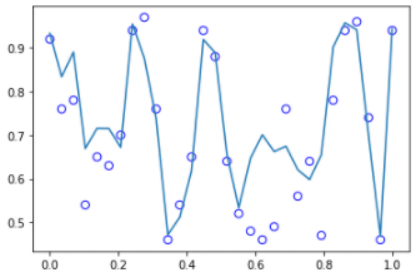


```
[57]: print('rmse result:', rmse(y_test, y_pred))
      rmse result: 0.05570575304404033
```

Gaussian

```
[42]: plt.scatter(list_x, y_test[:30], facecolor="none", edgecolor="b", s=50, label="training data")
      plt.plot(list_x, y_pred[:30])
```

```
[42]: [<matplotlib.lines.Line2D at 0x13a9326c400>]
```

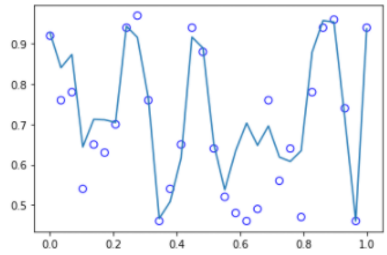


```
[43]: print('Gaussian rms error result:', rmse(y_test, y_pred))
      Gaussian rms error result: 0.06853405908828991
```

Sigmoid

```
[62]: list_x = np.linspace(0, 1, 30)
      plt.scatter(list_x, y_test[:30], facecolor="none", edgecolor="b", s=50, label="training data")
      plt.plot(list_x, y_pred[:30])
```

```
[62]: [<matplotlib.lines.Line2D at 0x13a92f306a0>]
```



```
[63]: print('Sigmoid rmse result:', rmse(y_test, y_pred))
      Sigmoid rmse result: 0.06650853971720312
```

In this experiment we compare three basis function. polynomial can get best performance.

2.3.a

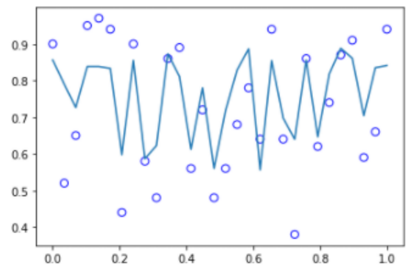
Please see 2.3.a.pdf

2.3.b

MPA result

```
[104]: plt.scatter(list_x, y_test[:30], facecolor="none", edgecolor="b", s=50, label="training data")
       plt.plot(list_x, y_pred[:30])
```

```
[104]: [<matplotlib.lines.Line2D at 0x13a95494b38>]
```



```
[105]: print('Gaussian rms error result:', rmse(y_test, y_pred))
       Gaussian rms error result: 0.07997968609757523
```

At first I thought that the error of MAP would be smaller than the error of MLA because L2 regulation could solve the overfitting problem. But the result was the opposite. My thought was that my model was underfitting, so L2 regulation did not help, but made the error worse. Come bigger