## Pre-Processing:

1. Checked for null values in the dataset but there were no null values present.
2. I found that the dataset was highly imbalanced and it's a classification problem.
3. Dropped unnecessary features (ID, molecule_name, conformation_name) from the dataset after making copy of it.
4. As there was no information provided regarding features, so I started looking for highly correlated features and removed them from the dataset by making copy of it.
5. I used heatmap for looking the correlation values using heatmap but there were too many features present, so I used 0.7 as threshold for correlation value which turned out to be perfect fit after training model for which I wrote custom code and dropped those features form .
   Features reduced from 169 to 52.
6. I also used recursive feature extraction but it was slow as there were too many features present, I took some insights from there and used correlation values to find suitable features.
7. Divided the training and testing in the ratio of 80 and 20 respectively.

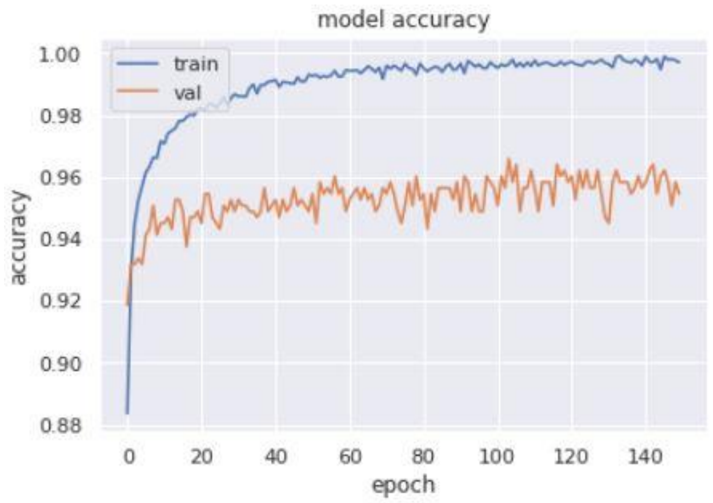
## Deep learning Framework:

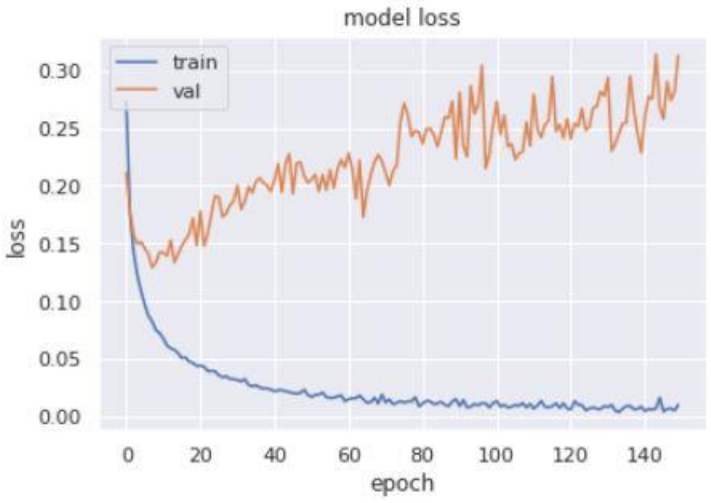
1. I used Keras framework for building neural network.
2. Now after making neural network with different layers I figured out stacking many layers didn’t give much better results.
3. So, I decided to go with not much deep neural network.
4. By hyper tuning the parameter presents in the Keras layer I was able to achieve decent results.
5. I used binary_crossentropy as  a loss function because the target column has only 1s ad 0s.
6. For updating the weights during training iteration I used adam optimizer as it calculates the individual learning rate of every parameter also it is one of the famous approach also gave me decent results.

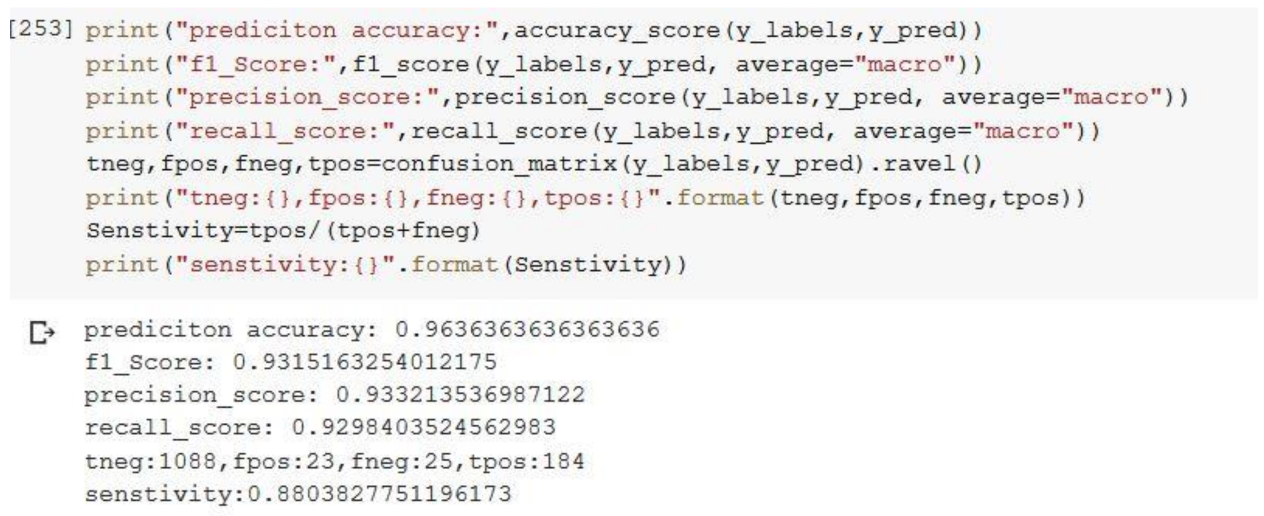
## Post-Processing:

1. The prediction results I achieve were in decimal, so I had to convert them into 0 or 1 as provided in the dataset.
2. By deciding the threshold value which is 0.5 in my case. I converted them into 0’s and 1’s accordingly.
3. Sensitivity, F1, precision and recall score also helped to evaluate the model and threshold value.

## Scores:


model accuracy
train
val
accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0
20
40
60
80
100
120
140
epoch


model loss
train
val
loss
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
20
40
60
80
100
120
140
epoch



```
[253] print("prediciton accuracy:",accuracy_score(y_labels,y_pred))
      print("f1_Score:",f1_score(y_labels,y_pred, average="macro"))
      print("precision_score:",precision_score(y_labels,y_pred, average="macro"))
      print("recall_score:",recall_score(y_labels,y_pred, average="macro"))
      tneg,fpos,fneg,tpos=confusion_matrix(y_labels,y_pred).ravel()
      print("tneg:{},fpos:{},fneg:{},tpos:{}".format(tneg,fpos,fneg,tpos))
      Senstivity=tpos/(tpos+fneg)
      print("senstivity:{}".format(Senstivity))
```

```
prediciton accuracy: 0.9636363636363636
f1_Score: 0.9315163254012175
precision_score: 0.933213536987122
recall_score: 0.9298403524562983
tneg:1088,fpos:23,fneg:25,tpos:184
senstivity:0.8803827751196173
```