NeuroFlow Report

Abhishek Dendukuri

April 2021

1 Data Initialization

Looking at the data we're given, we can see that the `patient_date_created` column does not change value for a repeated instance of `patient_id`. The type column also gives us only one unique value (`gad7`). With this we can remove those two columns, allowing us to focus on `date` and `score`.

```
df = pd.read_csv('phq_all_final.csv',
        parse_dates=['date', 'patient_date_created'])
df = df.sort_values(['patient_id', 'date'])
df = df.drop(['patient_date_created', 'type'], 1)
df['Ord'] = pd.to_datetime(df['date']).map(dt.datetime.toordinal)
```

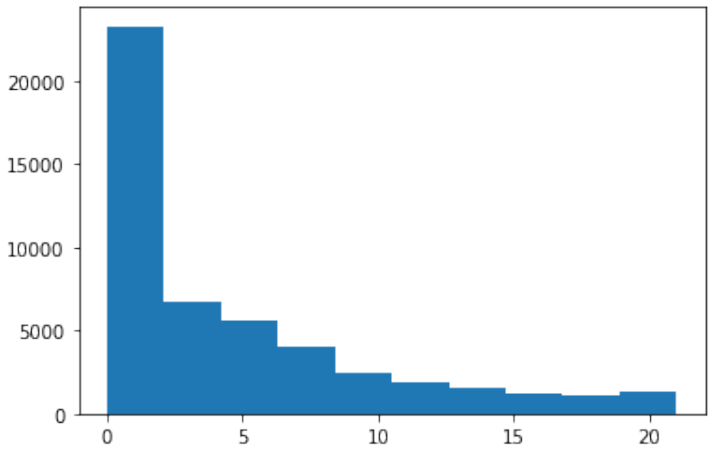
I also created a new column `Ord` to convert the date to an integer. This will be helpful at a later point when I want to use dates for linear regression purposes.

2 Data Preparation

First thing I want to do is filter out any rows where a patient took the test only once. Since we want to be able to support how well a patient is responding to treatment, monitoring multiple test scores per patient is crucial.

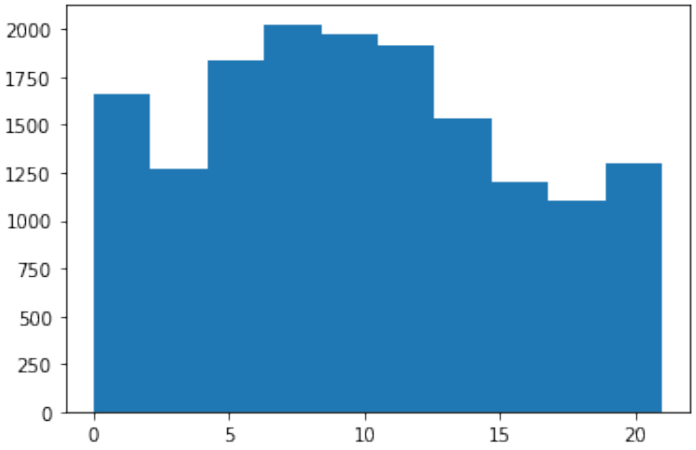
```
df = df[df.patient_id.map(df.patient_id.value_counts()) > 1]
```

Now let's see visualize the distribution of scores:



After plotting the first dataframe scores, we can see that there's an incredibly high concentration of values around `score = 0, 1`. To limit that I now want to filter out `patient_ids` that never eclipsed `score >= 10` in any measurement. Since the recommended threshold for further clinical evaluation is a score of 10 or higher, I want to make the assumption that patients who have reached this threshold are likely to seek therapy in order to better their condition.

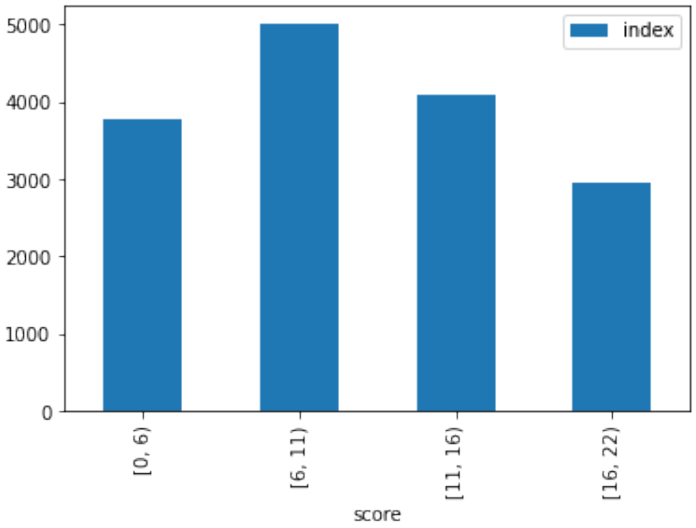
```
df['needsEval'] = 1*(df.score >= 10)
df2 = df[df.patient_id.map(df.groupby('patient_id').needsEval.sum() >= 1)]
```



Redistributing the bins according to the given scoring scale:

```
left = [0, 6, 11, 16]
right = [6, 11, 16, 22]
bins = pd.IntervalIndex.from_arrays(left, right, closed='left')
```

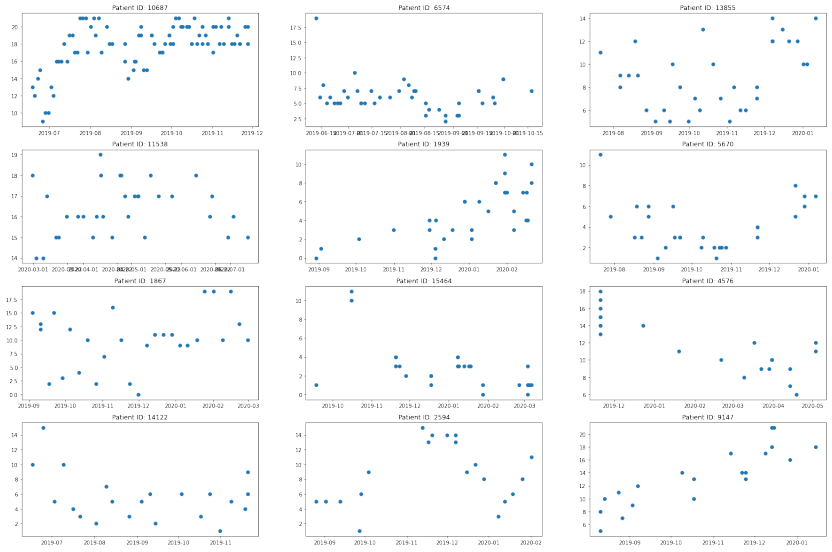
```
df_bin = pd.cut(df2.score, bins).reset_index()
df_bin_gp = df_bin.groupby('score').count()
df_bin_gp.plot(kind='bar')
```



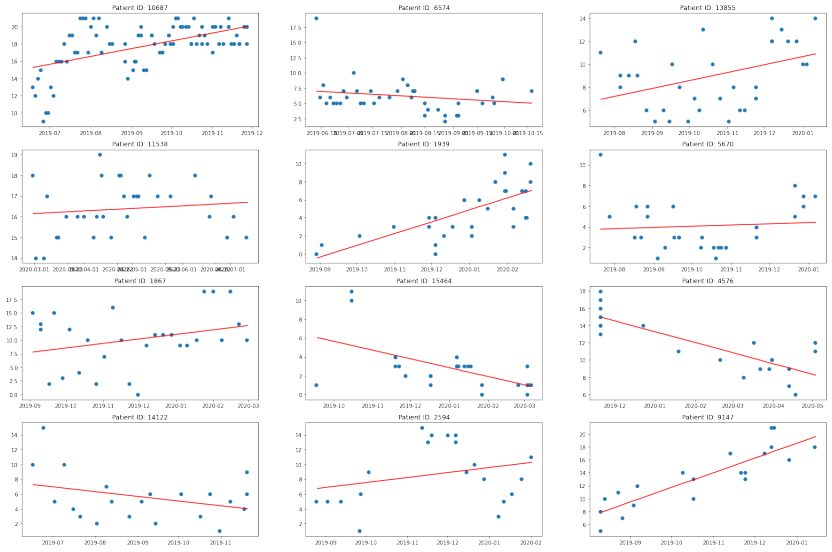
After removing those rows, we now have a histogram that is more uniformly distributed and it reveals that most patients report a score between 6-10.

3 Data Visualization by Patient

Now that we have reduced our dataset and gotten a reasonable distribution of scores, I then visualized the data for the patients that had the most scores recorded. I made the assumption that patients who recorded several scores seeked the most help/treatment throughtout their clinical process. I believe that making this assumption helps us better understand if patients are responding to treatment or not.



With these scatter plots, we can argue that there are some clear trends that emerge for some of the patients. To further extend my analysis, I then fit a linear regression line to each of the plots.



As we can see on the plots, there are some very clear trends for some patients, but for others the results are fairly inconclusive. But this is good information because now we can take the slope coefficient from linear regression and the resulting R^2 score to check how many patients are responding well to treatment and how many are not.

The first step I took was creating a new dataframe to contain all these new values, along with `patient_id`. I created a new binary variable `resp_to_treatment` that returned `True` for negative coefficient values (if the score decreases, the patient feels better) and `False` otherwise.

	patient_id	coefs	r2	resp_to_treatment
0	1	0.166667	1.000000	0
1	2	-0.085059	0.782544	1
2	15	0.042361	0.522690	0
3	32	-0.010667	0.014597	1
4	45	0.043478	1.000000	0
5	47	-0.176471	1.000000	1
6	52	0.054935	0.079511	0
7	56	0.099848	0.336191	0
8	57	-0.007641	0.012066	1
9	59	0.117647	1.000000	0

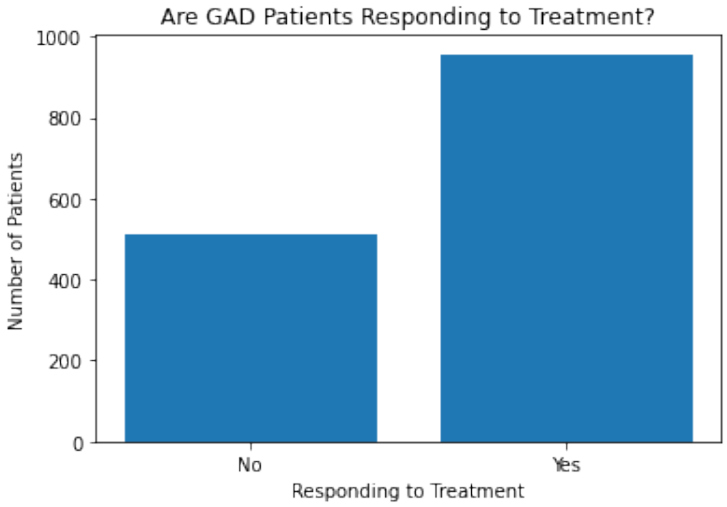
```
coefs = []
r2 = []
for pid in df2.patient_id.unique():
  lr.fit(df[df.patient_id == pid].Ord.values.reshape(-1, 1),
         df[df.patient_id == pid].score)
  coefs.append(lr.coef_)

  y_pred = lr.predict(df[df.patient_id == pid].Ord.values.reshape(-1, 1))
  r2.append(r2_score(df[df.patient_id == pid].score, y_pred))
```

I then created a restriction on the dataset to only show rows where the number of datapoints per `patient_id` were greater than 2, because for those that were equal to 2, the $R^2 = 1$ since it was fitting a line between two points. The next filter I created was to only include rows where $R^2 > 0.25$. This is more of an arbitrary measure and can be changed in any study, but I wanted to make the assumption that a good R^2 value is accompanied by at least a decently fit trendline.

```
df_lr_filt = df_lr[(df_lr.r2 != 1) & (df_lr.r2 > 0.25)]
df_resp_filt = df_lr_filt.groupby('resp_to_treatment').count().coefs
```

With our given parameters, we can now display a plot that reveals how many patients we believe are responding well to treatment.



4 Insights and Improvements

The GAD study is interesting because it is short questionnaire with the same 4 answers for each question. There are opportunities for us to add each question as a feature to the dataset and the patients' resulting responses to each question; with this information, we would be able to breakdown the metrics of the GAD scores even further and reveal some more interesting insights that could help clinical evaluation further. An example would be if a patient randomly fills in answers - we would be able to tell because we can assume that anxiety patterns would pertain over time and wild fluctuations in individual questions would stick out even if the total score remains relatively similar.

Another insight that I would love to collect would be socio-economic information - age, gender, marital status, income, etc - because we would be able to add further dimensionality to the dataset. This in partnership with the GAD question breakdowns as mentioned above can really help us identify which patients are at risk of getting worse, and which are responding better to treatment.