Gestational Diabetes Onset Estimation

By Jessica Lewis

Introduction

Gestational diabetes (GDB) is a type of diabetes that occurs during pregnancy. The placenta takes over hormone production for both the fetus and the host; in some pregnancies the placenta mismanages the uptake of blood sugar. This is what causes GDB. Without a prior diagnosis of gestational diabetes or a history of non-gestational diabetes, pregnant people are typically tested for GDB between 24 and 28 weeks of pregnancy.

Problem Statement

Can I reverse a time series of gestational diabetes data to pinpoint the onset in my pregnancy?

SUCCESS CRITERIA

There are two aspects to this project: making predictions on a reversed time series and estimating the onset of GDB in my pregnancy. I consider this project a success if either of these criteria succeed.

1. I make predictions for dates before my data starts
2. My predictions cross a threshold chosen using target blood sugar levels for pregnant people. These values depend on what I end up using as my target variable:
    - Fasting Levels > 95
    - Non-Fasting Levels > 130
    - Daily average of both > 120

Context

I became pregnant in 2021 and at 27 weeks I was diagnosed with GDB. I immediately scheduled an appointment with a specialist and at 29 weeks I changed my diet and began taking blood sugar readings 4 times per day: when I woke up and an hour after each meal. I was also given carb and protein targets for meals/snacks and recorded my carb intake. I took up to 10 records per day, until I went into labor, totalling roughly 11 weeks of data.

Approach

The Data

All of the data had been saved to a mobile app called One Touch Reveal, a popular tracking app for people with diabetes and their healthcare professionals. I was able to log into the website application and load a report with all of my data and although there was an export function it was unhelpful for this project; it compiled a pdf file. There was also an API working to generate the report, but it was not accessible to me.

At first, I scraped the data via a javascript function that I run directly on the page. The output was a JSON string that I then copied and pasted into a file that I could import in a notebook.

Here is the javascript function that I used:

```
function getData() {
    const myTable = $('table.datatable');
    const myRows = myTable.children().children('.bgRow');

    const records = []


    // loop through tr.bgRow
    myRows.each((i,el) => {

        const row = $(el);
        const cols = row.children();

        const first = cols.eq(0);

        var rowType = first.text();
        var detailTableClassSelector = `.${rowType == 'BG' ? 'bg' : 'carb'}detailtable`;


        // loop through cells in current row
        cols.each((i,el) => {

            const col = $(el);
            const value = col.find('.liner').text().trim();


            // if current cell has a value then it has the class 'toolTip'
            if( col.hasClass('toolTip') ) {
                const metaTableCols = col.find(`${detailTableClassSelector} td`);

                // save the data of the current cell
                records.push([
                    first.text(),

                    // grab the first value in the tooltip --> timestamp
                    metaTableCols.eq(0).text(),

                    // grab the second value in the tooltip --> value
                    metaTableCols.eq(1).text(),

                    // grab the third value in the tooltip --> notes
                    metaTableCols.eq(2).text()
                ])
            };

        });

    });

    console.log(JSON.stringify(records));
}
getData();
```

After working with the data for a while I realized that the notes column in the webapp was different from the notes I had saved in the mobile app. I wasn't able to find my notes in the HTML of the auto-generated log so I ended up downloading the app onto my phone again and finding an export button. Unfortunately the export feature only worked from the current date; I couldn't set custom export parameters. I was able to get around this by manually setting the system data on my phone to the last day of my data and then running the "Last 90 days" export option, which was enough to include everything. Once I had the spreadsheet on my phone storage I copied it to google sheets and was then able to grab it on my laptop for this project.

A COUPLE NOTES ON TERMINOLOGY

There are some terms that I use interchangeably throughout this project. Blood glucose is the same thing as blood sugar. Also, readings and records both mean rows in the dataframe.

Data Cleaning and Preprocessing

After loading the data I began by fixing the datatypes and I set the index to the datetime column. I removed rows where the value was 0 and checked for other null values. Next up was feature extraction. Overall the data didn't need a lot of cleaning.

Feature Extraction

In order to work with the data by day, instead of by datetime, I added a column that counted up from 1 for each calendar day. I called this daycount and used it to iterate through each day while setting the next feature: subtypes.
Each record came in with a type: either Blood Sugar Reading or Carbs. But each of those also broke down into more specific subtypes that I wanted to be able to compare:

Blood Sugar Reading

- fasting
- after_breakfast
- after_lunch
- after_dinner

Carbs

- breakfast
- snack_morning
- lunch
- snack_afternoon
- dinner
- snack_night

Once I had subtypes for each record I noticed that there were quite a few repeated subtypes within the same day. In these cases I had, say, eaten two snacks instead of one, or had some more lunch before taking my blood sugar reading. In these cases I wanted to combine the extra readings so that each day only had one of each subtype. I built a function to do this but still ended up having to make quite a few individual decisions.

Once I had all of the subtypes in I created another column to flag values that are outOfRange. These are subtype-specific:

fasting	>	90	breakfast snack_morning snack_afternoon snack_night	>	30
after_breakfast after_lunch after_dinner	>	130	lunch dinner	>	45

FULL DATAFRAME

This is the primary dataset that went through wrangling and cleaning. It consists of one record per row and includes these features:

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 683 entries, 2021-03-11 07:11:00 to 2021-05-23 21:56:00
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   type        683 non-null    object
 1   value       683 non-null    int64 
 2   unit        683 non-null    object
 3   month       683 non-null    int64 
 4   date        683 non-null    object
 5   daycount    683 non-null    int64 
 6   subtype     683 non-null    object
 7   outOfRange  683 non-null    bool  
dtypes: bool(1), int64(3), object(4)
memory usage: 59.5+ KB
```

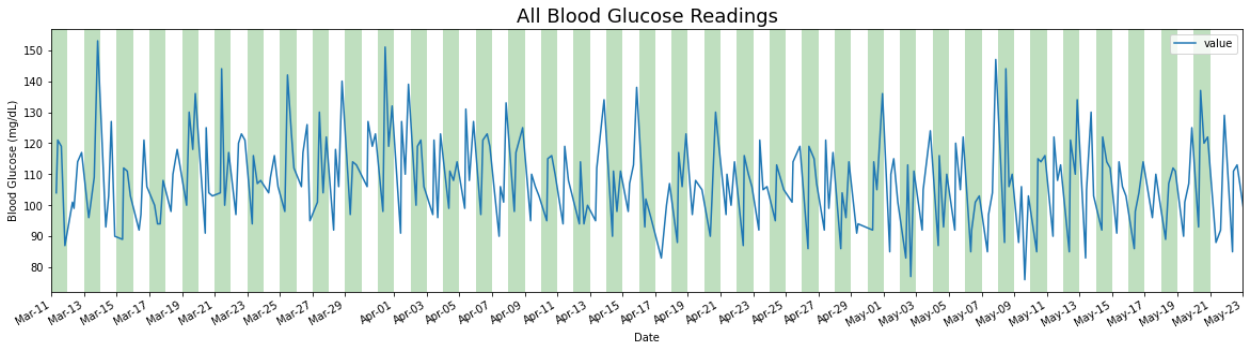
DAILY DATAFRAME

This is a consolidation of the data by day, using aggregation of the values of the full dataframe by type or subtype. It also includes a feature called meds_dose with the dosage of Metformin I took to get bg_fasting under control.

```
<class 'pandas.core.frame.DataFrame'>
Index: 73 entries, 2021-03-11 to 2021-05-23
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   daycount     73 non-null     int64
 1   bg_fasting   73 non-null     float64
 2   bg_avg       73 non-null     int64
 3   carbs_sum    73 non-null     int64
 4   meds_dose    73 non-null     int64
dtypes: float64(1), int64(4)
memory usage: 5.5+ KB
```

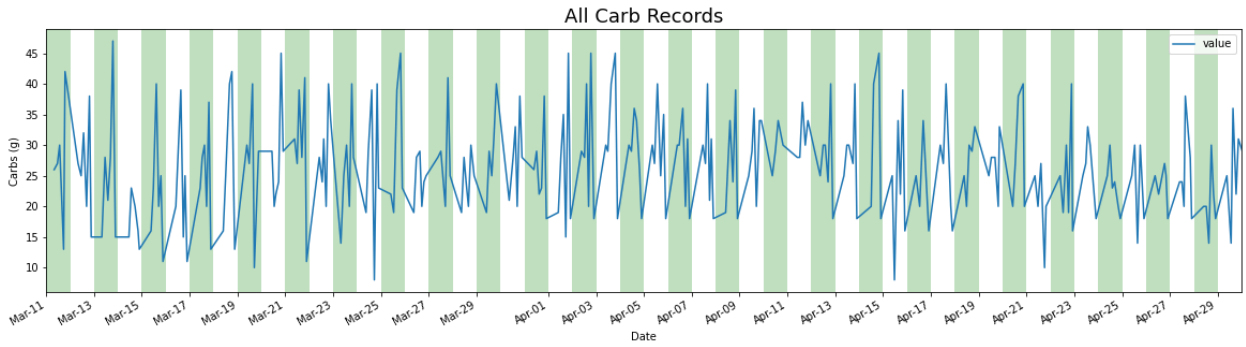
EDA

Full Dataframe



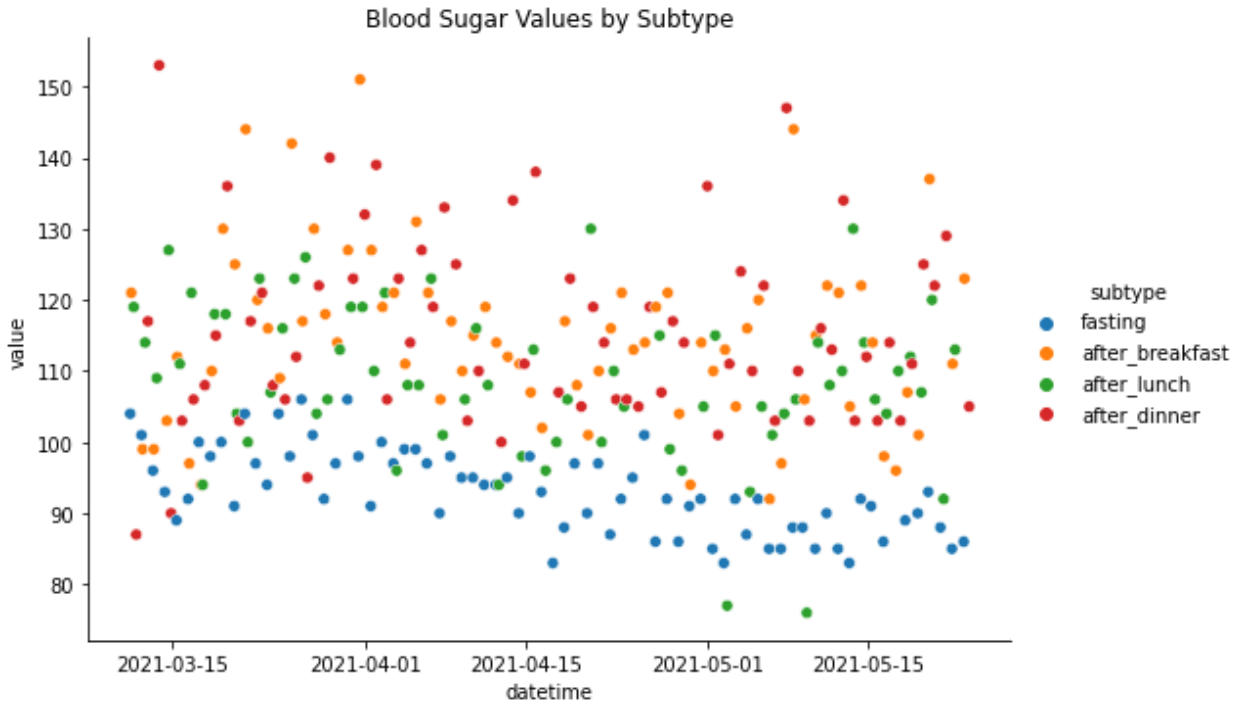
Notes:

- I expected to see a tiny bit more seasonality in the full blood sugar graph, since fasting numbers (the first of every day) should be lower than non-fasting. I suppose it makes sense that there is no visual seasonality since my fasting numbers were always higher than they should have been.
- This chart doesn't really help me much.



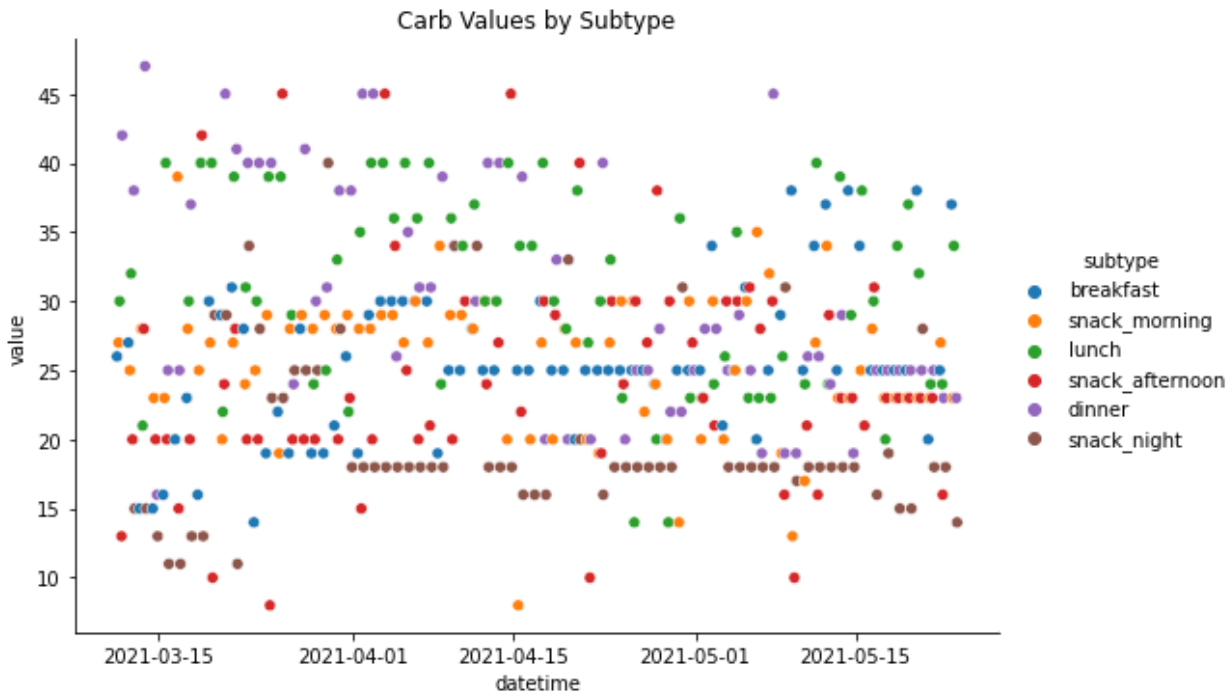
Notes:

- I can see a little more seasonality in this chart, but overall it's also not very helpful.



Notes:

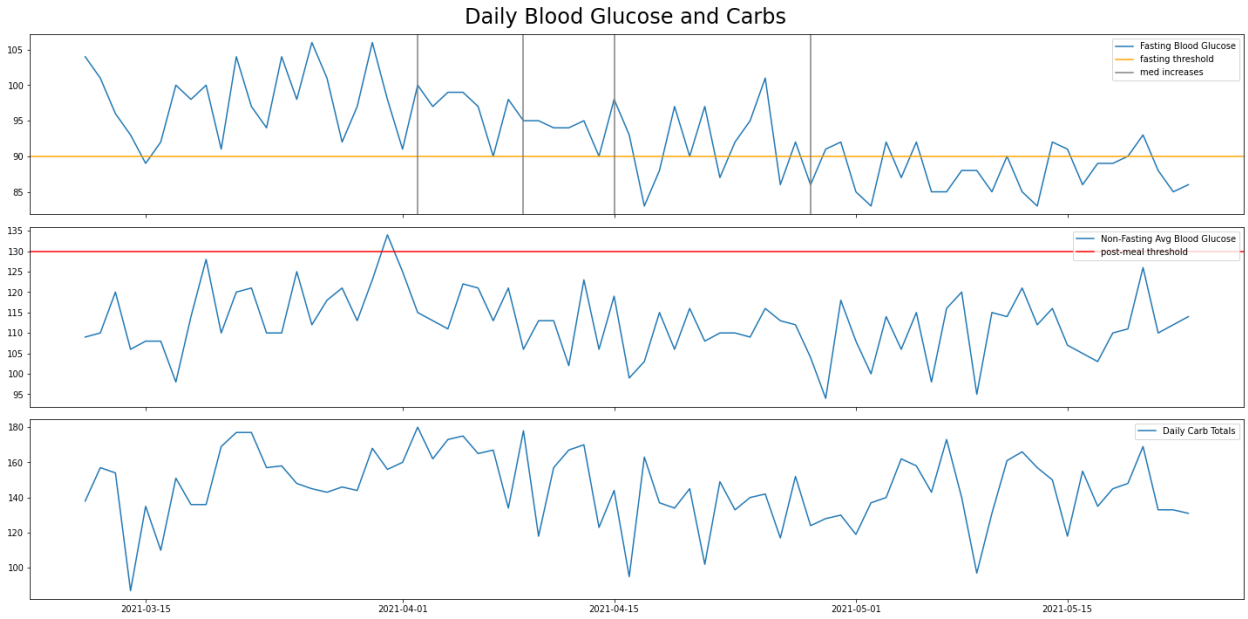
- It's easy to spot the out of range readings, specifically for after_lunch and after_dinner
- The fasting readings, in blue, show a nice downward trend that I'm sure would correlate with meds_dose increases



Notes:

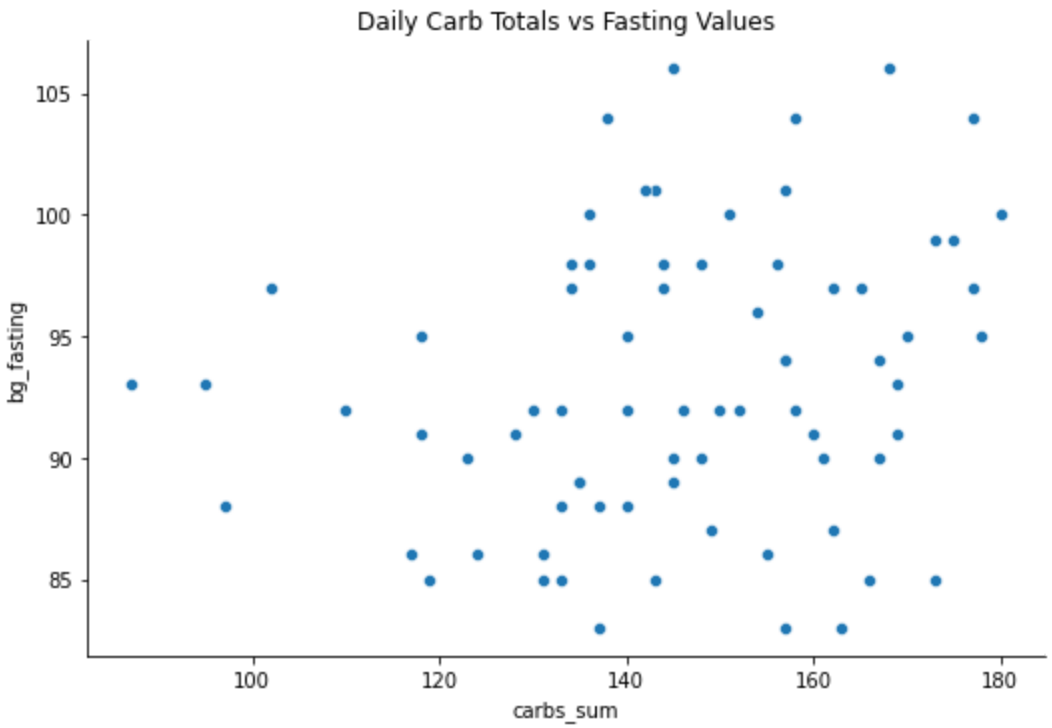
- My main takeaway from this one is that as time progresses you can see more consecutive values where I settled on certain foods that I liked for certain meals/snacks

Daily Dataframe



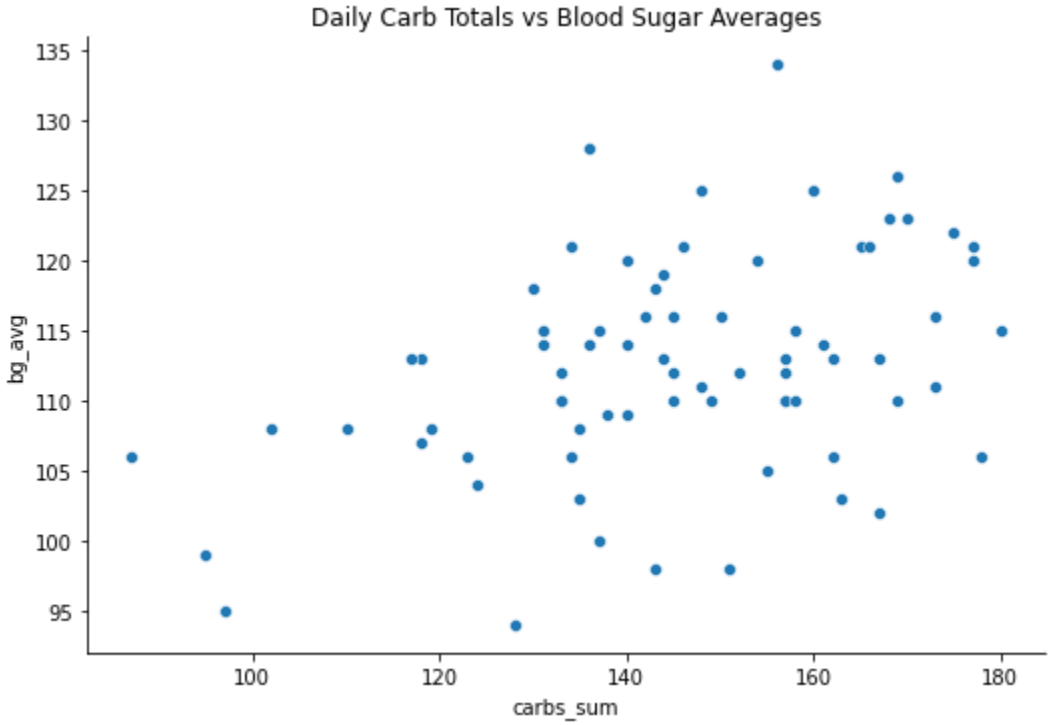
Notes:

- The gray lines in the first graph are the dates I increased my meds_dose
    - It's nice to see how clearly the medication helped lower my fasting blood sugar readings.
- The yellow line in the first graph is the target threshold for fasting values
- The red line in the second graph is the target threshold for non-fasting values
- A lot of the non-fasting avg dips correlate to dips in the carbs total graph, which makes sense
- I might see a little bit of an upward trend in the beginning of the non-fasting average graph. Perhaps that's enough for a decent onset prediction?



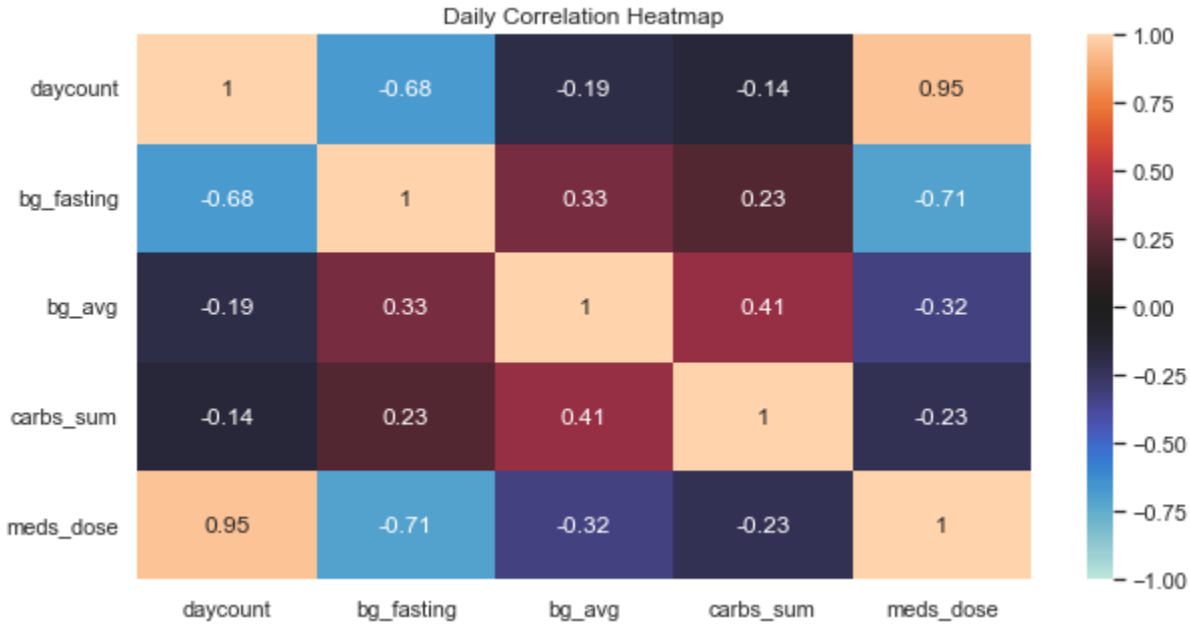
Notes:

- You can see a loosely positive correlation here



Notes:

- As expected, there is a stronger correlation between carbs and non-fasting blood sugar values
- There's still quite a bit of spread, indicating that there are other things influencing blood sugar levels (exercise, hormones, etc)



Notes:

- The heatmap reinforces what I already know about this data
    - The highest correlation is between meds_dose and bg_fasting, which was the point
    - There is a moderate correlation between carbs and bg_avg

Pre-processing

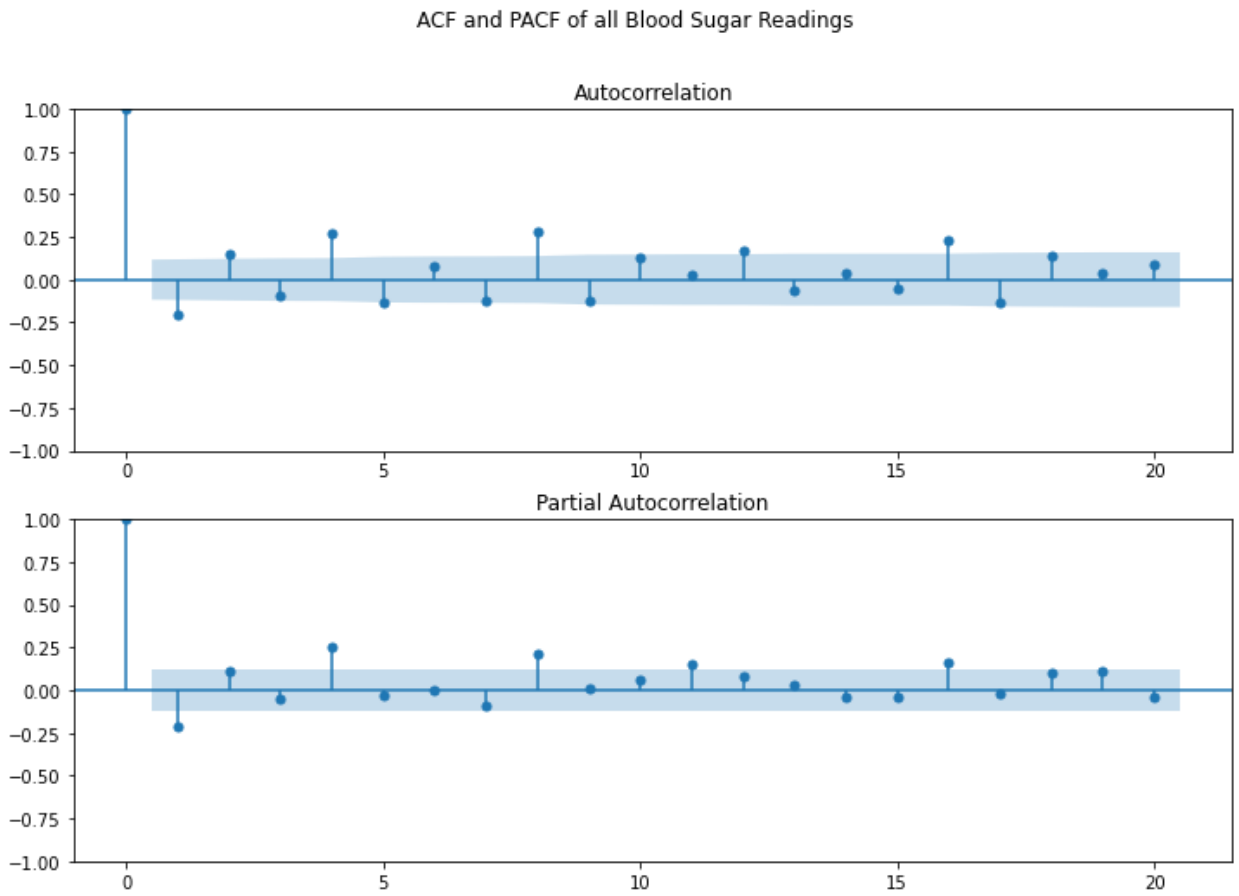
Full Dataframe

TESTING FOR STATIONARITY

Test	p-value	Result
Dickey Fuller	0.129589	non-stationary
KPSS	0.01	non-stationary

Both tests indicate that the time series is non-stationary.

ACF AND PACF



The ACF shows small, erratic, autocorrelations for all visualized lags. According to Duke University's guide on ARIMA models this indicates that the data does not need a higher order of differencing (Rule 2). This contradicts the results of the Dickey Fuller test.

I went ahead and differenced this time series and the ACF/PACF confirmed that one order of differencing is too much.

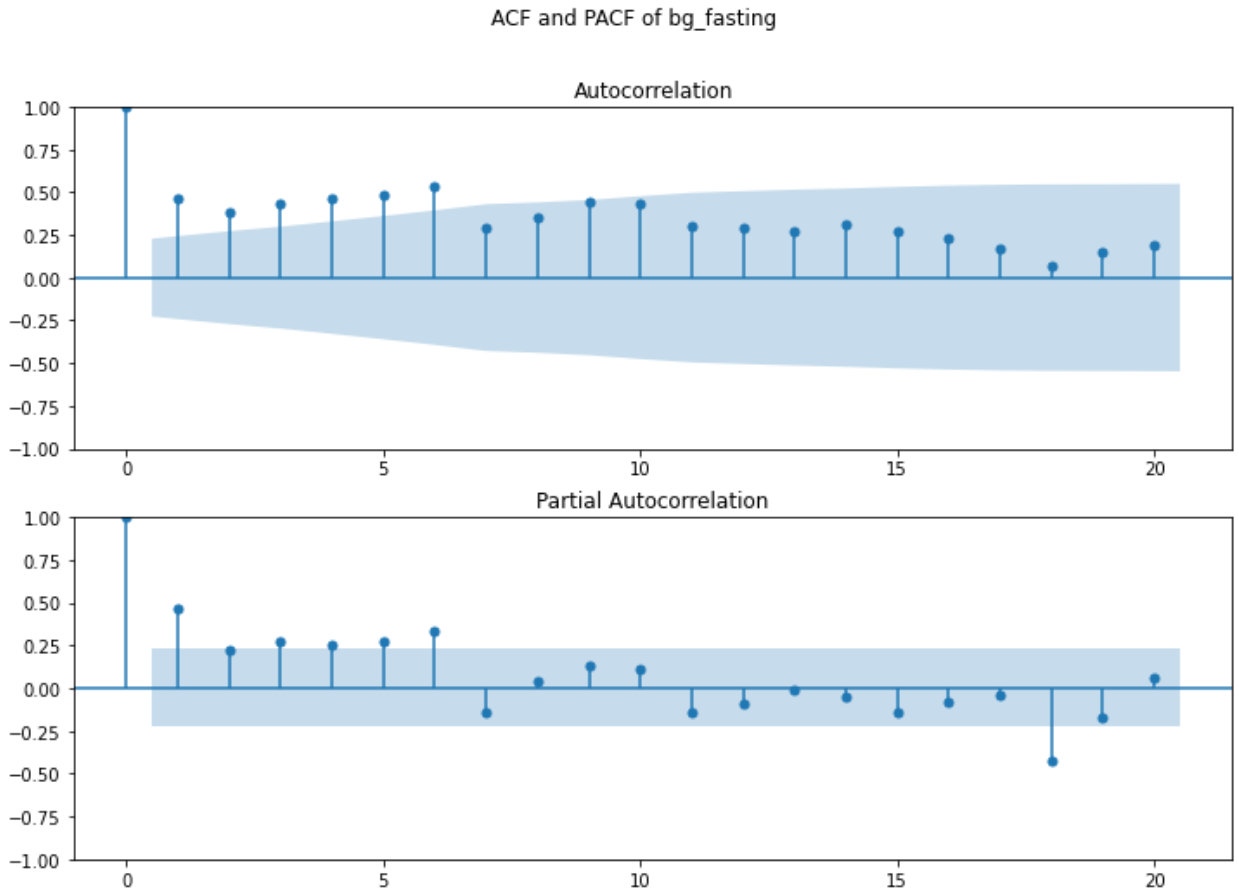
Daily Fasting

TESTING FOR STATIONARITY

Test	p-value	Result
Dickey Fuller	0.81028	non-stationary
KPSS	0.01	non-stationary

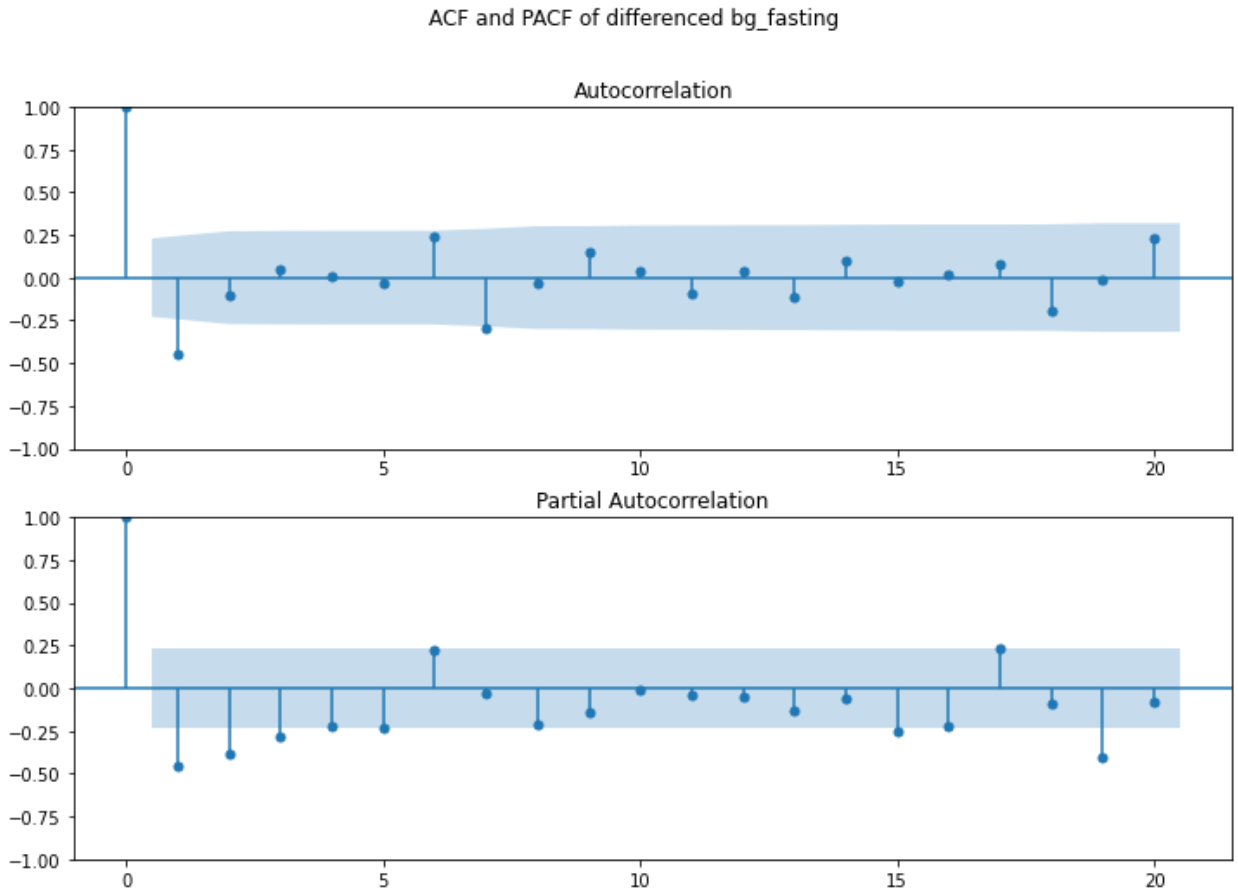
Both tests indicate that the time series is non-stationary.

ACF AND PACF



The ACF shows positive correlations for all visualized lags. According to Duke University's guide on ARIMA models this indicates that the data likely needs a higher order of differencing (Rule 1). This confirms the results of the Dickey Fuller test.

LET'S DIFF!



The differenced ACF and PACF look slightly over-differenced because that first lag is almost -0.5. We're on the right track, but let's make sure we have at least one MA order.

Running the Dickey Fuller test again confirms stationarity of the differenced time series.

Daily Blood Sugar Averages

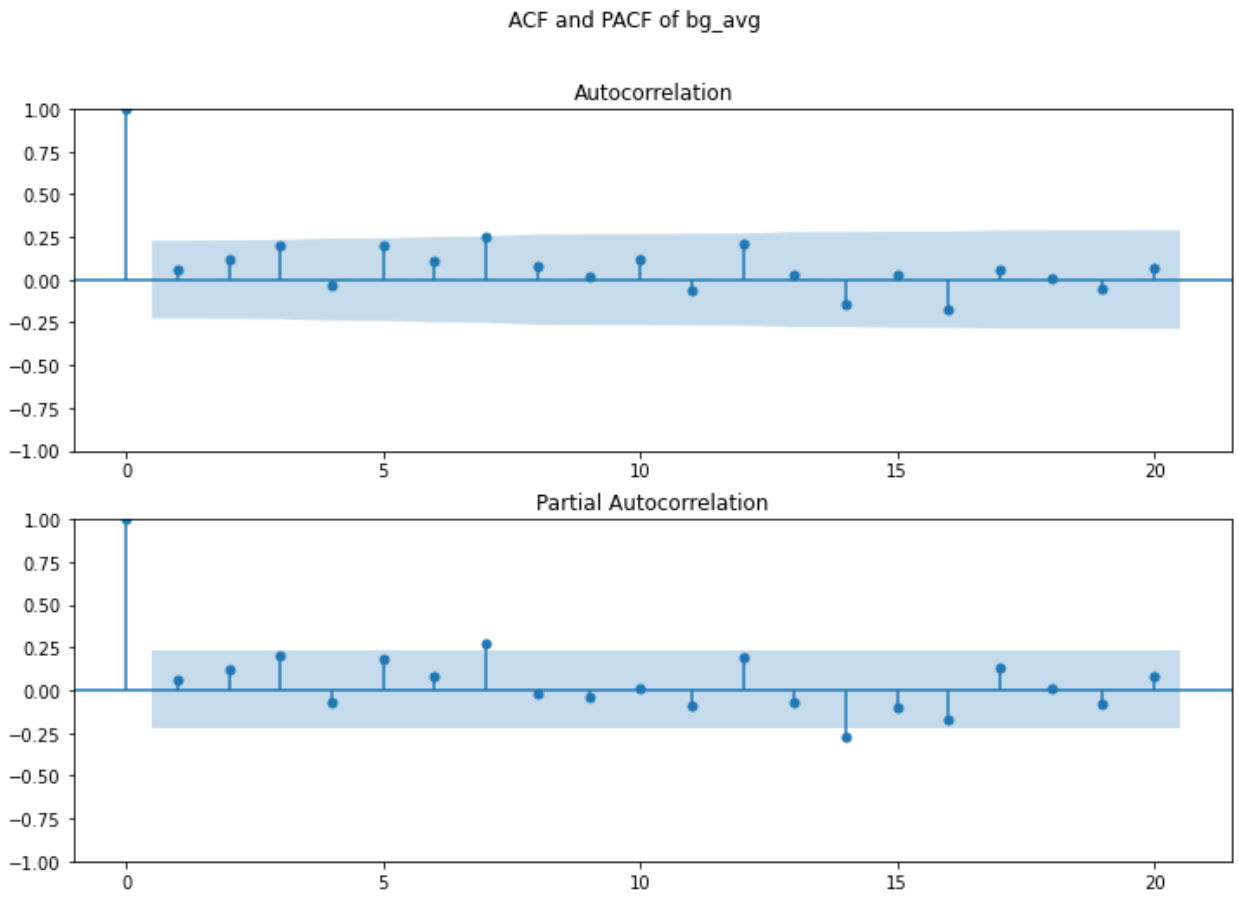
TESTING FOR STATIONARITY

Test	p-value	Result
Dickey Fuller	0.520113	non-stationary

KPSS	0.049775	non-stationary

Both tests indicate that the time series is BARELY non-stationary. This is a case where the p-level threshold can be interpreted differently if we have good reason. Let's see what that ACF and PACF look like.

ACF AND PACF



I'm not convinced that any differencing needs to be done, and because of how close the Dickey Fuller and KPSS tests were I'm going to go forward without differencing.

Models

At first I planned on only modeling my daily fasting records. After doing so and struggling to get any reasonable predictions I tossed the daily averages into my models to compare. Once I was doing that I figured I might as well do the same with my full blood sugar set.

Before exploring a couple models I made sure to split my data into training and test sets, 80% and 20% respectively. This way I'll have some room to train my final model on the full.

I started with your basic ARIMA model. I have an idea where to start looking for good orders, but I also wrote a function to test a range of parameters and return the top three orders organized by AIC.

Daily Fasting Readings

ARIMA

I knew from my stationarity testing and ACF/PACF that I'd need 1 order of differencing and 1 or higher MA, so I ran my ARIMA optimization function with the following parameters:

```
ps = range(0,8)
qs = range(1,8)
d = 1
```

And here are the top three orders:

	Order	AIC
0	(5, 1, 3)	323.945820
1	(5, 1, 4)	325.173819
2	(4, 1, 7)	325.750351

Fitted Model Results:

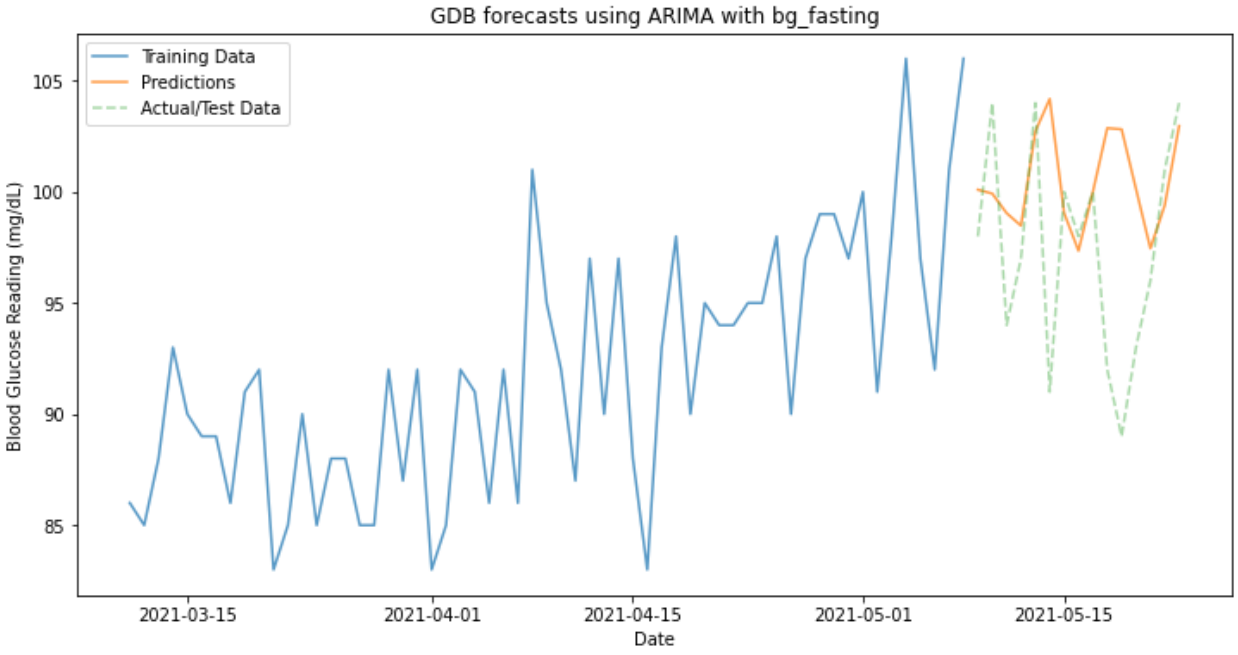
SARIMAX Results

Dep. Variable:	bg_fasting	No. Observations:	59
Model:	ARIMA(5, 1, 3)	Log Likelihood	-152.973
Date:	Wed, 09 Mar 2022	AIC	323.946
Time:	20:40:21	BIC	342.490
Sample:	03-11-2021	HQIC	331.169
	- 05-08-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.9836	0.173	-5.679	0.000	-1.323	-0.644
ar.L2	-1.2687	0.202	-6.289	0.000	-1.664	-0.873
ar.L3	-1.3976	0.150	-9.305	0.000	-1.692	-1.103
ar.L4	-0.9021	0.180	-5.018	0.000	-1.255	-0.550
ar.L5	-0.5989	0.153	-3.921	0.000	-0.898	-0.300
ma.L1	-0.0393	0.234	-0.168	0.866	-0.498	0.419
ma.L2	0.4840	0.380	1.272	0.203	-0.262	1.230
ma.L3	0.6784	0.452	1.502	0.133	-0.207	1.563
sigma2	9.9175	4.829	2.054	0.040	0.454	19.381

Ljung-Box (L1) (Q):	0.06	Jarque-Bera (JB):	0.47
Prob(Q):	0.80	Prob(JB):	0.79
Heteroskedasticity (H):	2.75	Skew:	0.04
Prob(H) (two-sided):	0.03	Kurtosis:	3.43

Predictions vs test set:



AUTO ARIMA

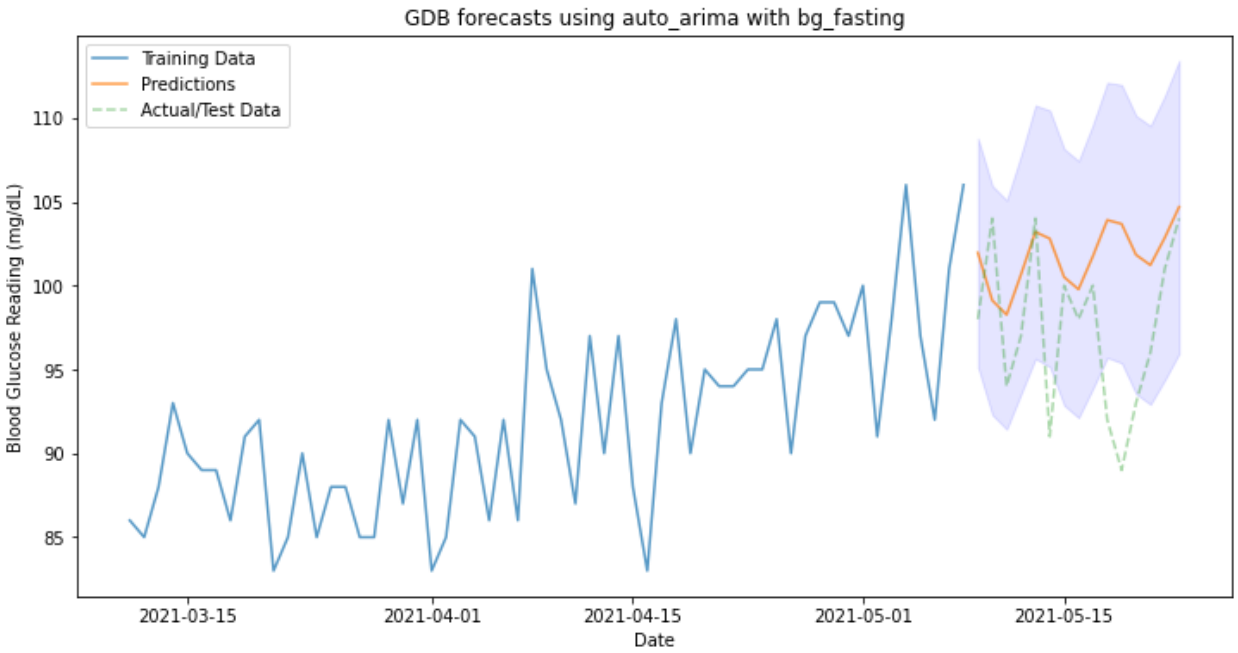
Model Order: (2, 1, 3)

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	59
Model:	SARIMAX(2, 1, 3)	**Log Likelihood**	-156.709
Date:	Wed, 09 Mar 2022	**AIC**	327.418
Time:	21:04:35	**BIC**	341.841
Sample:	0	**HQIC**	333.036
	- 59		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	0.2908	0.118	2.456	0.014	0.059	0.523
ar.L1	0.6048	0.068	8.852	0.000	0.471	0.739
ar.L2	-0.9251	0.084	-11.046	0.000	-1.089	-0.761
ma.L1	-1.6623	0.165	-10.099	0.000	-1.985	-1.340
ma.L2	1.6657	0.412	4.042	0.000	0.858	2.473
ma.L3	-0.7765	0.252	-3.077	0.002	-1.271	-0.282
sigma2	11.9380	3.676	3.248	0.001	4.733	19.143

Ljung-Box (L1) (Q):	0.38	**Jarque-Bera (JB):**	0.45
Prob(Q):	0.54	**Prob(JB):**	0.80
Heteroskedasticity (H):	1.47	**Skew:**	0.10
Prob(H) (two-sided):	0.41	**Kurtosis:**	2.62



METRICS

Model Metrics: Daily Fasting

	RMSE	Standardized
ARIMA	6.4957	1.3815
Auto Arima	6.7511	1.4359

Based on the models so far, it looks like the ARIMA model is performing better than the auto-arima. That said, I'm not sure I'm seeing the downward trend I'd like to see at the very end of the predictions; there might be a slight decrease in the ARIMA model, which would account for the better RMSE.

Let's keep exploring.

Daily Blood Sugar Averages

ARIMA

I knew from my stationarity testing and ACF/PACF that there is no need to difference this time series. I ran my ARIMA optimization function with the following parameters:

ps = range(0,8)
qs = range(0,8)
d = 0

And here are the top three orders:

	Order	AIC
0	(0, 0, 7)	411.538806
1	(3, 0, 3)	412.252132
2	(0, 0, 0)	412.969429

Fitted Model Results:

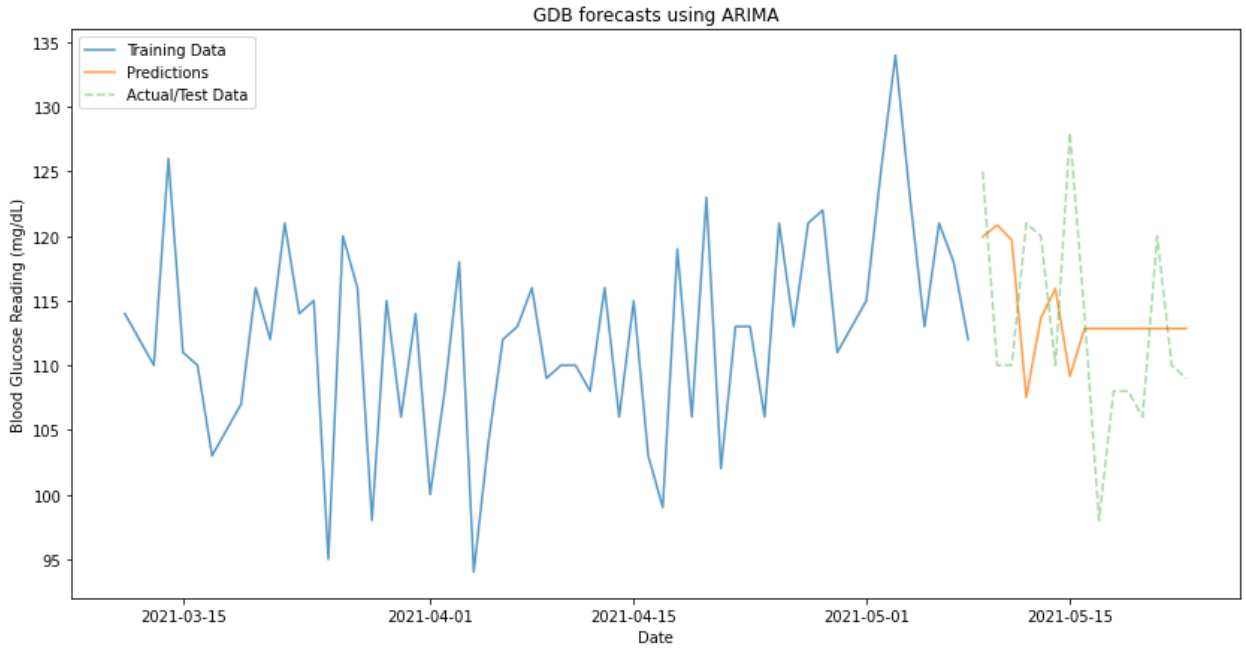
SARIMAX Results

Dep. Variable:	bg_avg	No. Observations:	59
Model:	ARIMA(0, 0, 7)	Log Likelihood	-196.769
Date:	Wed, 09 Mar 2022	AIC	411.539
Time:	11:53:41	BIC	430.237
Sample:	03-11-2021	HQIC	418.838
	- 05-08-2021		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
const	112.8440	2.119	53.248	0.000	108.690	116.998
ma.L1	0.0238	0.380	0.063	0.950	-0.721	0.768
ma.L2	0.1521	0.371	0.410	0.682	-0.575	0.880
ma.L3	0.1044	0.437	0.239	0.811	-0.751	0.960
ma.L4	0.0723	0.393	0.184	0.854	-0.697	0.842
ma.L5	0.2921	0.484	0.604	0.546	-0.656	1.241
ma.L6	-0.1266	0.351	-0.360	0.719	-0.815	0.562
ma.L7	0.6420	0.323	1.989	0.047	0.009	1.275
sigma2	42.4077	16.209	2.616	0.009	10.639	74.176

Ljung-Box (L1) (Q):	0.13	Jarque-Bera (JB):	0.05
Prob(Q):	0.72	Prob(JB):	0.97
Heteroskedasticity (H):	0.75	Skew:	0.01
Prob(H) (two-sided):	0.52	Kurtosis:	3.14

Predictions vs test set:



The order that was used here has quite a bit of smoothing, which is affecting the predictions in a significant way. I don't like that, so let's try the second recommended order.

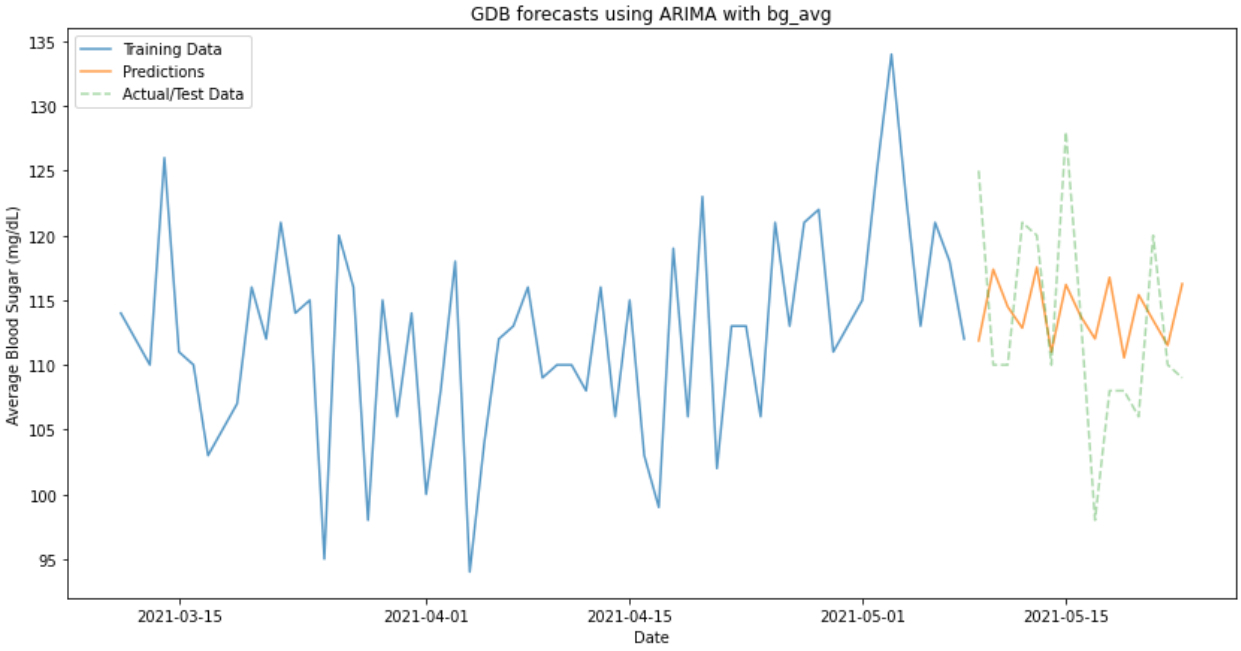
ARIMA 2

Model Order: (3, 0, 3)

Fitted Model Results:

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                 bg_avg   No. Observations:                   59
Model:                ARIMA(3, 0, 3)   Log Likelihood                -198.126
Date:                Wed, 09 Mar 2022   AIC                            412.252
Time:                        11:53:43   BIC                            428.872
Sample:                    03-11-2021   HQIC                           418.740
                         - 05-08-2021
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        112.7030      2.377     47.424      0.000     108.045     117.361
ar.L1         -0.6919      0.256     -2.699      0.007      -1.194      -0.189
ar.L2          0.4514      0.384      1.177      0.239      -0.301       1.203
ar.L3          0.8841      0.251      3.516      0.000       0.391       1.377
ma.L1          0.7871      1.191      0.661      0.509      -1.547       3.121
ma.L2         -0.1877      0.764     -0.246      0.806      -1.686       1.311
ma.L3         -0.7526      1.234     -0.610      0.542      -3.171       1.666
sigma2        45.5484     65.376      0.697      0.486     -82.586     173.683
===================================================================================
Ljung-Box (L1) (Q):                   0.38   Jarque-Bera (JB):                 0.43
Prob(Q):                              0.54   Prob(JB):                         0.81
Heteroskedasticity (H):               0.97   Skew:                            -0.19
Prob(H) (two-sided):                  0.94   Kurtosis:                         2.85
===================================================================================
```

Predictions vs test set:



That's much better. It's a little smoothed, but actually that's not a bad thing considering the final model will be compared with a threshold rather than simply predicting out a certain timeframe.

AUTO ARIMA

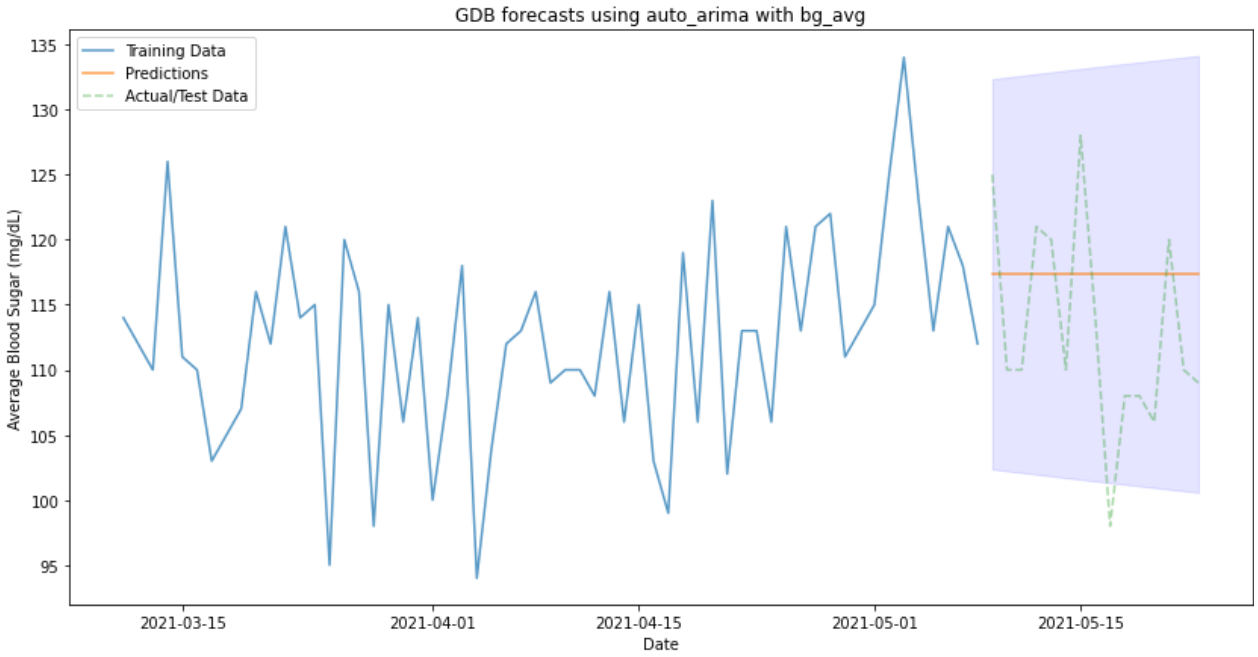
Model Order: (0, 1, 1)

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	59
Model:	SARIMAX(0, 1, 1)	**Log Likelihood**	-200.952
Date:	Wed, 09 Mar 2022	**AIC**	405.904
Time:	11:53:44	**BIC**	410.024
Sample:	0	**HQIC**	407.509
	- 59		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ma.L1	-0.8655	0.082	-10.555	0.000	-1.026	-0.705
sigma2	58.4203	11.681	5.001	0.000	35.525	81.315

Ljung-Box (L1) (Q):	0.96	**Jarque-Bera (JB):**	0.28
Prob(Q):	0.33	**Prob(JB):**	0.87
Heteroskedasticity (H):	1.00	**Skew:**	-0.15
Prob(H) (two-sided):	0.99	**Kurtosis:**	2.85



For some reason auto_arima did not handle the data very well.

METRICS

Model Metrics: Daily BG Average

	RMSE	Standardized
ARIMA (3, 0, 3)	7.8574	1.0101
Auto Arima	8.8481	1.1374
ARIMA (0, 0, 7)	9.0899	1.1685

The RMSE mirrors my opinion that the second ARIMA model I tried performed the best of the three models.

All Blood Sugar Readings

ARIMA

I knew from my stationarity testing and ACF/PACF that I'd need 1 order of differencing and 1 or higher MA, so I ran my ARIMA optimization function with the following parameters:

ps = range(0,8)
qs = range(1,8)
d = 1

And here are the top three orders:

	Order	AIC
0	(5, 1, 3)	323.945820
1	(5, 1, 4)	325.173819
2	(4, 1, 7)	325.750351

Fitted Model Results:

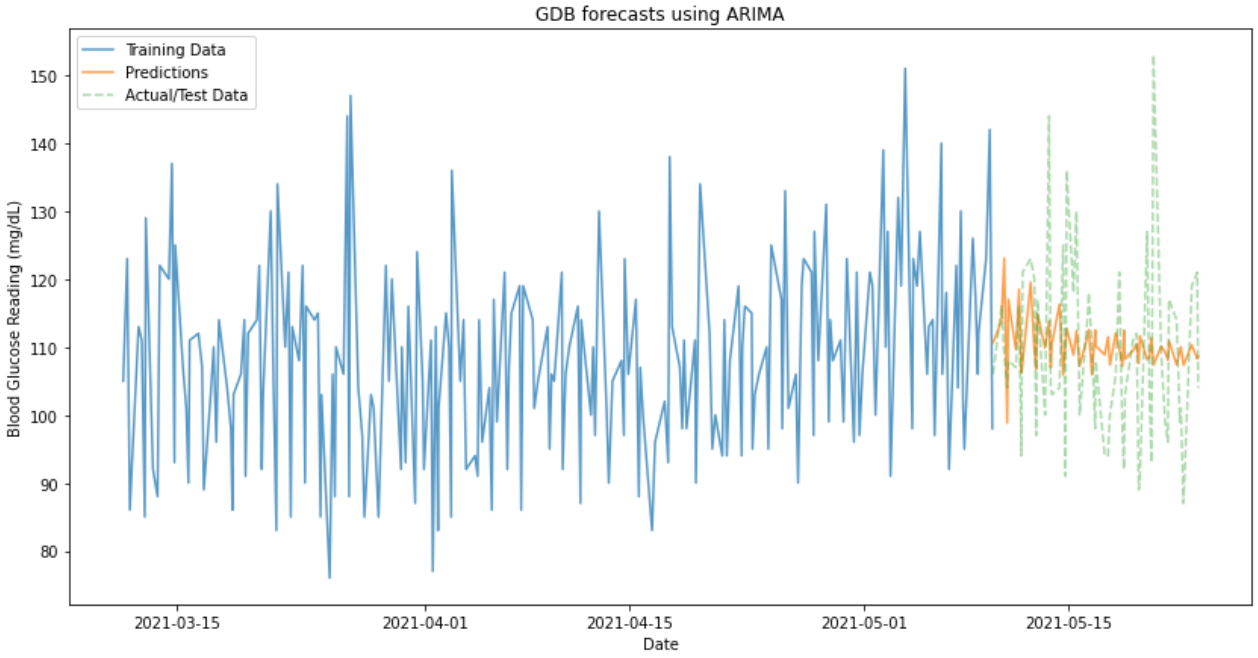
SARIMAX Results

Dep. Variable:	bg_fasting	No. Observations:	59
Model:	ARIMA(5, 1, 3)	Log Likelihood	-152.973
Date:	Wed, 09 Mar 2022	AIC	323.946
Time:	20:40:21	BIC	342.490
Sample:	03-11-2021 - 05-08-2021	HQIC	331.169
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	-0.9836	0.173	-5.679	0.000	-1.323	-0.644
ar.L2	-1.2687	0.202	-6.289	0.000	-1.664	-0.873
ar.L3	-1.3976	0.150	-9.305	0.000	-1.692	-1.103
ar.L4	-0.9021	0.180	-5.018	0.000	-1.255	-0.550
ar.L5	-0.5989	0.153	-3.921	0.000	-0.898	-0.300
ma.L1	-0.0393	0.234	-0.168	0.866	-0.498	0.419
ma.L2	0.4840	0.380	1.272	0.203	-0.262	1.230
ma.L3	0.6784	0.452	1.502	0.133	-0.207	1.563
sigma2	9.9175	4.829	2.054	0.040	0.454	19.381

Ljung-Box (L1) (Q):	0.06	Jarque-Bera (JB):	0.47
Prob(Q):	0.80	Prob(JB):	0.79
Heteroskedasticity (H):	2.75	Skew:	0.04
Prob(H) (two-sided):	0.03	Kurtosis:	3.43

Predictions vs test set:



I like those predictions. It's not unexpected to see that reversion to the mean, but I believe I also see a slight downward trend.

AUTO ARIMA

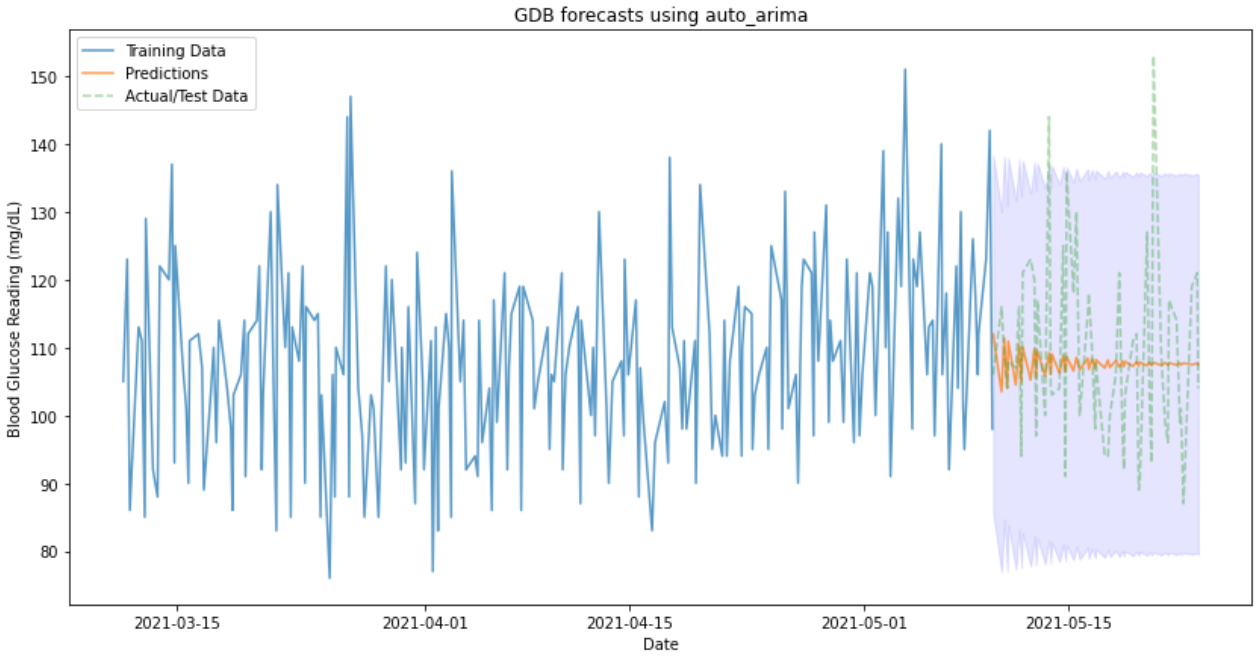
Model Order: (1, 0, 1)

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	224
Model:	SARIMAX(1, 0, 1)	**Log Likelihood**	-900.197
Date:	Wed, 09 Mar 2022	**AIC**	1808.393
Time:	21:43:35	**BIC**	1822.040
Sample:	0	**HQIC**	1813.902
	- 224		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	207.9564	4.874	42.663	0.000	198.403	217.510
ar.L1	-0.9331	0.043	-21.505	0.000	-1.018	-0.848
ma.L1	0.8063	0.075	10.719	0.000	0.659	0.954
sigma2	180.9520	18.370	9.850	0.000	144.947	216.957

Ljung-Box (L1) (Q):	0.03	**Jarque-Bera (JB):**	3.23
Prob(Q):	0.85	**Prob(JB):**	0.20
Heteroskedasticity (H):	0.94	**Skew:**	0.28
Prob(H) (two-sided):	0.78	**Kurtosis:**	2.82



Again, the reversion to the mean is expected, but it happens very fast and looks very flat. I'm expecting the metrics to favor the ARIMA model.

METRICS

Model Metrics: All Blood Sugar Readings

	RMSE	Standardized
Auto Arima	13.3582	0.9837
ARIMA (7, 0, 5)	13.5269	0.9961

Despite the lower RMSE of auto_arima, I think ARIMA technically scored better because the standardized value (which is RMSE / Standard Deviation of y_test) is closer to 1. It just happens to be below 1 instead of over 1 like the other models.

Performance

Predictions vs Test Values

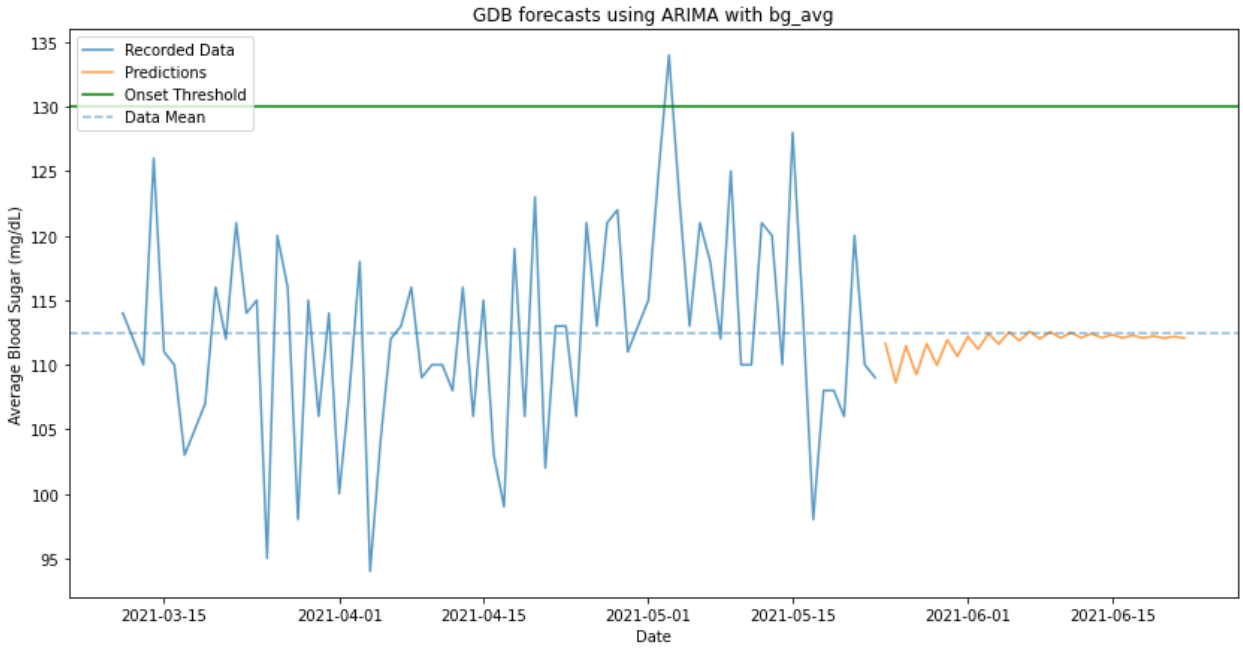
Let’s compare the metrics for all of the models across the different datasets.

Target Variable	Model	RMSE	Standardized	Absolute Value of 1 - Standardized
bg_fasting	**ARIMA**	6.278110	1.335264	0.335264
	Auto Arima	6.751092	1.435861	0.435861
bg_avg	**ARIMA (0, 0, 7)**	9.089890	1.168490	0.16849
	ARIMA (3, 0, 3)	7.857382	1.010054	0.010054
	Auto Arima	8.848146	1.137414	0.137414
all	**ARIMA (7, 0, 5)**	13.526907	0.996113	0.003887
	Auto Arima	13.358158	0.983687	0.016313

I calculated how close each model came to Standardized = 1 and the top two models are the ARIMA model using the full blood sugar dataset and the second ARIMA model using the daily averages. These were the two that I pointed out as potential final models.

Final Model

Target Variable: bg_avg
Model: ARIMA (3, 0, 3)



The predictions after fitting the model on the entire dataset were, expectedly, not as good as I saw in the training/test model. There is a fast reversion to the mean, which is exactly what you would expect to see here. Unfortunately it happens really fast and goes nowhere near the green threshold I was hoping would estimate onset.

This failure is because of the small amount of data; although there is probably some hyperparameter fine-tuning I could try, the threshold line is so far above my data that there's no way this model could reach it.

Ideas for Improvement

There is a lot of room for improvement in this experiment. Here are a few ideas I've recorded while working through this capstone.

- Optimize ARIMA order by RMSE instead of AIC or BIC
    - This could lead to overfitting, but since I have so little data, and only a small trend at the very end I want to focus on, I'd like to see if that actually helps with the prediction.
- Focus on data before meds, either by isolating that data or by giving it extra weight

- ** moving average might weight
  - Simple or exponential moving average
- Facebook prophet model

Conclusion/Final Thoughts

Despite the failed predictions of the final model I still call this experiment a moderate success. The goal was to see if I could reverse a time series to make predictions in the past and I found a way to do that. For most of the workflow I used a time series with a forward-running date index, which was necessary for functions that require monotonicity. Once I had run my final model, however, I tried pulling in my reversed time series and running that through and it actually worked. I got an warning that the next version of the model will throw an exception instead of generating predictions, and the predictions didn't come in with a date index. In the end I just made a date index manually and then graphed it all together: