2022

Final Report: Time Series Analysis of Tweets' Sentiment

CSI 678: TIME SERIES ANALYSIS AND FORECASTING

SAMIUL ISLAM

Time Series Analysis of Tweets' Sentiment

1. Introduction:

Since its inception in March 2006, Twitter has become very popular among people from every race as a microblogging and social networking service. Users post and interact with messages known as tweets. As a Twitter account holder, one can post a tweet, like a tweet, retweet, etc. Due to its immense popularity and wide acceptability, marketing agencies, social and political groups, etc., demands different analysis and act on the outcome of those in their favor. Understanding the sentiment and predicting it ahead of time is a hotcake to that community, and statistical and machine learning approaches are acting as a booster to this goal.

Through Twitter, we have access to some of the tweets' data from different states. These tweets' data instances have many attributes, and such voluminous data can be crucial to answering some of the pressing social questions. So, in this project, I used time series analysis to predict the polarity values of the tweets. Rather than going on the individual level, I focused on grouped statistics and used the hourly data to estimate or forecast.

2. Data Description:

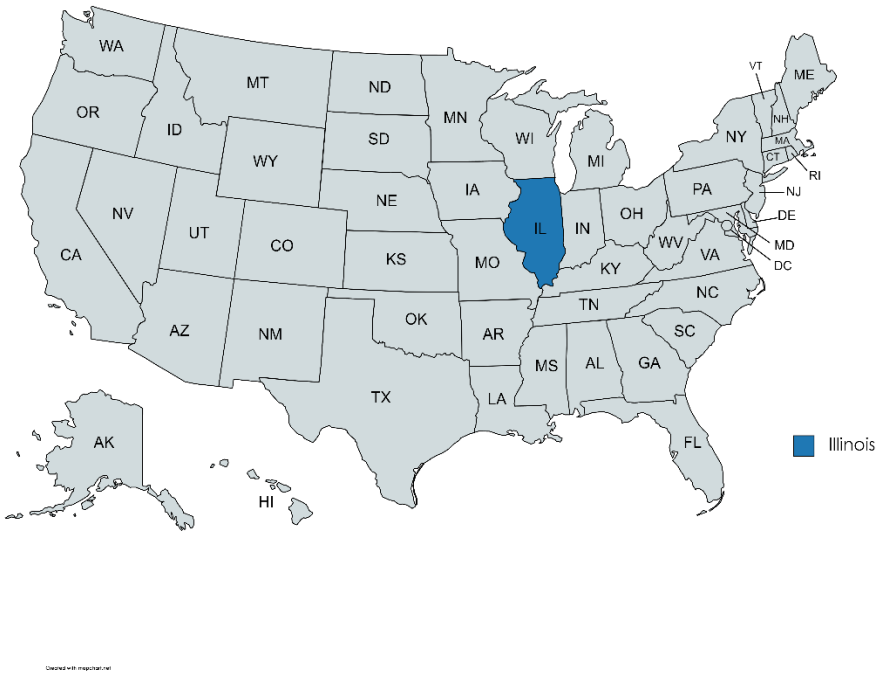

Fig 1: Geographical Region of this Study

To design the research prototype, I will focus on Illinois state (IL; highlighted in figure 1) and use some of the tweets made by the users (Twitter account holders) from May 16, 2014, to the end of that year. We

have 20 million (20,237,711) tweets from IL that I can use for this project. This dataset of tweets is very rich in terms of the attributes that are available for each of the instances (follow figure 2).

	id [PK] bigint	date character varying (100)	loc_time character varying (100)	tid character varying (50)	county character varying (50)	county_census character varying (10)	lat double precision	lng double precision	state [PK] character (2)	zipcode character (5)	uid character varying (50)	name character varying (100)
1	4378479	Sat May 17 03:59:59 +0000 2014	2014-05-16 22:59:59	467514845546676224	Cook	031	41.89168036000001	-87.62788338	IL	60611	240920261	Harley Blevins□
2	4378480	Sat May 17 02:31:49 +0000 2014	2014-05-16 21:31:49	467492655988224002	DuPage	043	41.78006103999999	-88.25960118	IL	60502	352723884	Tom Carlson
3	4378481	Sat May 17 02:31:49 +0000 2014	2014-05-16 21:31:49	467492658227589121	Cook	031	41.875125929999996	-87.62700824	IL	60605	354080439	L o
4	4378482	Sat May 17 02:31:49 +0000 2014	2014-05-16 21:31:49	467492658244354048	DuPage	043	41.78833700999999	-87.97138615	IL	60559	387513432	Qarin Johnnic
5	4378483	Sat May 17 02:31:50 +0000 2014	2014-05-16 21:31:50	467492659779891201	Cook	031	42.108849899999996	-87.84058476	IL	60062	279666652	Austin Weber
6	4378484	Sat May 17 02:31:51 +0000 2014	2014-05-16 21:31:51	467492666700476416	DuPage	043	41.966533999999996	-88.0527412	IL	60157	705931909	007
7	4378485	Sat May 17 02:31:52 +0000 2014	2014-05-16 21:31:52	467492668747304961	Lake	097	42.33383416000001	-88.08577934	IL	60073	49889712	Sean Wells
8	4378486	Sat May 17 02:31:54 +0000 2014	2014-05-16 21:31:54	467492677617860608	Will	197	41.50568960999999	-87.83435488	IL	60423	490804937	em
9	4378487	Sat May 17 02:31:54 +0000 2014	2014-05-16 21:31:54	467492676318015488	Cook	031	41.857926219999996	-87.65648761	IL	60608	174274812	Kelly Northcutt
10	4378488	Sat May 17 02:31:54 +0000 2014	2014-05-16 21:31:54	467492677756653570	Christian	021	39.38435655000001	-89.08327435	IL	62557	777289087	Lauren Stauder
11	4378489	Sat May 17 02:31:55 +0000 2014	2014-05-16 21:31:55	467492679895760897	Cook	031	41.926188060000015	-87.66200569	IL	60614	264976808	Henny Cent
12	4378490	Sat May 17 02:31:55 +0000 2014	2014-05-16 21:31:55	467492680617961 61	Cook	031	41.9657383	-87.7676874	IL	60630	2183240461	Desiree
13	4378491	Sat May 17 02:31:56 +0000 2014	2014-05-16 21:31:56	467492687516430336	Will	197	41.6445999	-88.12244197	IL	60446	371411917	Mop Savage

Fig 2: Snapshot of the Raw Data

	id [PK] bigint	state character (2)	uid character varying (50)	tid character varying (50)	text text
1	1	IL	240920261	467514845546676224	This awful day turned into a good night.. Avatar is on 😍
2	2	IL	352723884	467492655988224002	Avatar is a modern day Pocahontas... I can't see how they make 3 more of these things...
3	3	IL	354080439	467492658227589121	@HasherRoth Henny
4	4	IL	387513432	467492658244354048	Talk about the threepeat shut the state out once again #hellyea #lovethisteam
5	5	IL	279666652	467492659779891201	@madalynnloren @katiemwhelan @jimmerb40 WELCOME!
6	6	IL	705931909	467492666700476416	DVC champs on both levels, couldn't be prouder to be a falcon tonight, great job boys #FalconPride
7	7	IL	49889712	467492668747304961	Jesus came to save us from our sins. Sin perverts purpose & corrupts potential 🔑
8	8	IL	490804937	467492677617860608	"@bigboobprobs: Discretely holding your boobs when going up and down the stairs in public. #BigBoobProbs"
9	9	IL	174274812	467492676318015488	Bacon and brew #yum tpizzle44 @ Honky Tonk Bbq http://t.co/CWY9wtasFg
10	10	IL	777289087	467492677756653570	"I thought an exclusive relationship was when you go out and do things outside the home" 😬

Fig 3: Snapshot of the Raw Tweets

The source data has a lot of attributes for each of those 20 million tweets, such as global and local timestamp, tweet id, county from where the tweet has been made, latitude and longitude, user id, display name, etc., along with the actual tweet. Along with many others, this dataset has the exact text of those 20 million tweets (a snapshot of these is in figure 3), which can be used to design, train, validate and test a neural language-based model to answer the users' emotions possibly, thought processes, etc. However, I am not going that route; instead, I would like to use an unorthodox approach of streamlining the data as time series and forecasting the time series to identify sentiment.

I need the timestamp and associated tweeted texts for this project, meaning I can use all 20 million tweets. However, I decided not to use all of them. The reason is that I am currently working with another research team working on the same data where I use the available features to resolute key-demographics characteristics of the users. In that process, I need to sacrifice some tweets (it may not be a good idea to explain why I need to do that here since the objectives of these two projects are different). I decided to use the same 6 million tweets I am using on that project to keep coherency and make the integration easy when we have the outcomes from both projects.

3. Research Objective:

I want to use the time-series data of polarity values to predict future values. This primary objective is to do so on aggregated data of all the users from Illinois on an hourly basis. However, people's sentiment depends on many other factors related to demography and society. Since I am not taking any of those variables as predictors and grouping users without considering their characteristics and mutual interests,

the general performance of this initial approach is expected to hit a threshold. Although my primary goal is believed to be limited by these factors, I am designing the working prototype so that I can pass a more explainable set of tweets in the future before integrating it with the other project. Due to time limitations, I could not group the tweets based on different topics as it needs topic modeling, a new domain of research for me. As a set of secondary objectives of this research, I plan to go for topic modeling as I think people of similar interests should share the same topic in their tweets more often. For example, people who closely follow sports may tweet regarding soccer after an intense champion's league match; climate activists can be found active during an international meeting of policymakers, etc. Grouping topics based on the related topics and analyzing similar groups (similar demographical and socioeconomic characteristics) should yield high accuracy when using the historical time series data and other associated predictors.

4. Preprocessing:

Most of this prototyping model is involved in getting the data ready for time series analysis. I am using more than 20 million tweets from Illinois, where I performed some simplistic cleaning methods before using their polarity value. I have access to the users' display names from the source data. A person's name can be used to synthesize race and gender; first name can tell whether the person is male or female, and race/ethnicity can be identified from the family name. I am using that route for the other project that I am involved in and keeping up the coherency between the projects; I decided to use the data that stands after identifying race, gender, etc. The display name field is unstructured and often remains noisy. Like the tweet cleaning process, I employed a similar set of simplistic cleaning approaches for the display name fields. The cleaning approaches are as follows:

- Tweet Cleaning
    - Remove Special Characters
    - Remove Hashtags
    - Remove Retweets
    - Remove Links/Web Addresses

- Name Cleaning
    - Remove Special Characters
    - Substitute'_' with a Space
    - Strip Unwanted Spaces
    - Split Based on Spaces (to identify different parts of the names)

	pos	neg
2014-05-16 17:00:00	0.166924	-0.323421
2014-05-16 18:00:00	0.156326	-0.337527
2014-05-16 19:00:00	0.160065	-0.350448
2014-05-16 20:00:00	0.159889	-0.359096
2014-05-16 21:00:00	0.160754	-0.348988
...	...	...
2015-01-01 00:00:00	0.171641	-0.337959
2015-01-01 01:00:00	0.174609	-0.347718
2015-01-01 02:00:00	0.165537	-0.348260
2015-01-01 03:00:00	0.174688	-0.307910
2015-01-01 04:00:00	0.175032	-0.333467

5347 rows × 2 columns

Fig 4: Data Grouped by Polarity Values and Aggregated on an Hourly Basis

After performing data cleansing, I got more than 6 million tweets to work with. Each of these has been passed through a polarity calculating function where I used the 'textblob' package available in Python.

I separated tweets into two groups by positive and negative polarity values. I used timestamps (available from the source data) to group the sentiment on an hourly basis (follow figure 4). Through this process, I ended up with 5347 records of data indexed by their corresponding timestamp.

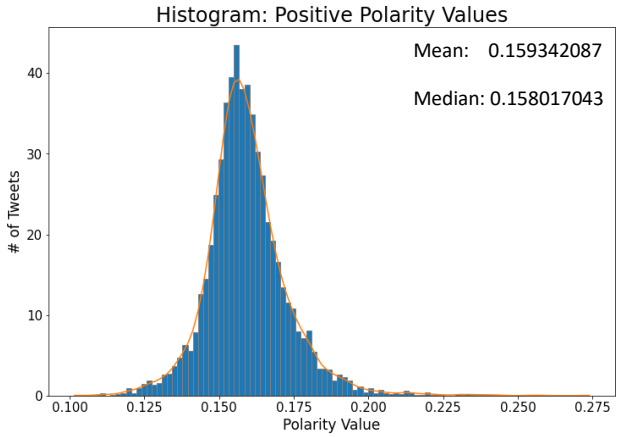

Fig 5(a): Positive Values Following Normal Distribution.

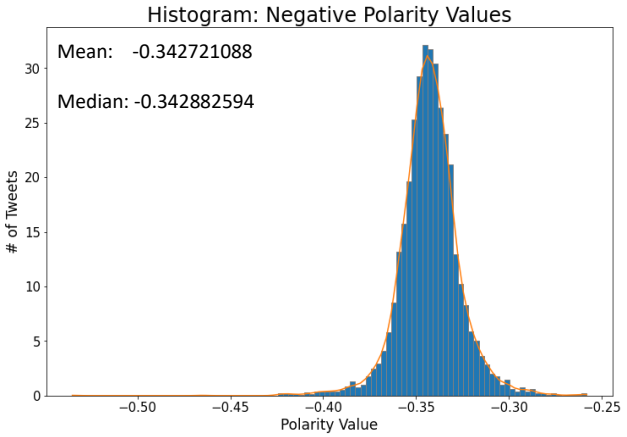

Fig 5(b): Negative Values Following Normal Distribution.

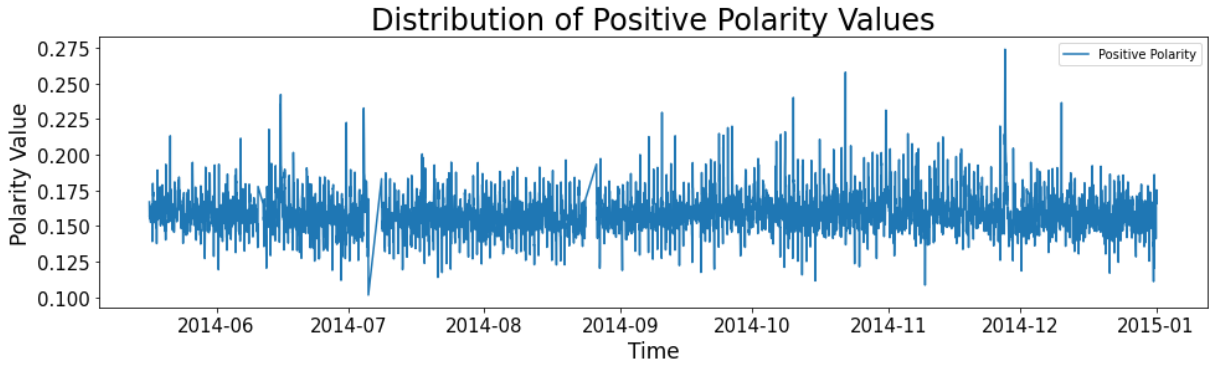

Fig 5(c): Positive Values Mostly Remain Stationary for the Entire Period

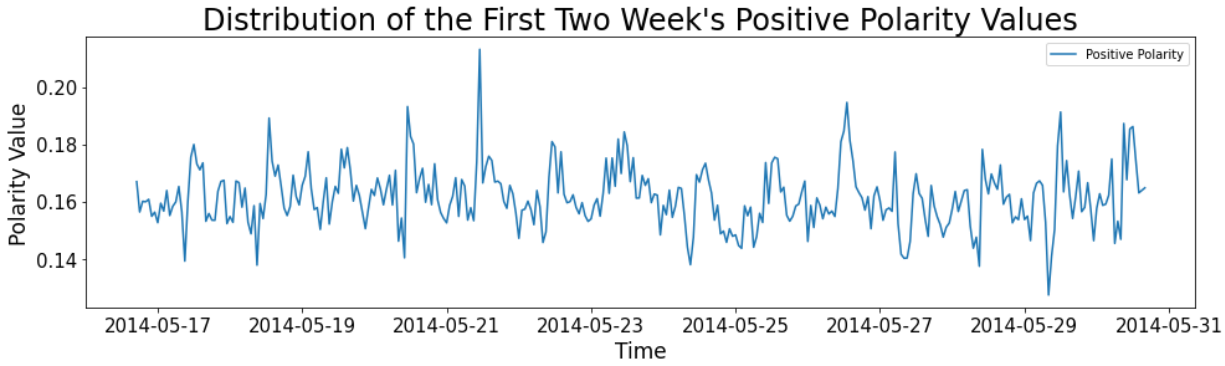

Fig 5(d): From the First Two Weeks of Data, there is no Visible Seasonality among the Positive Values

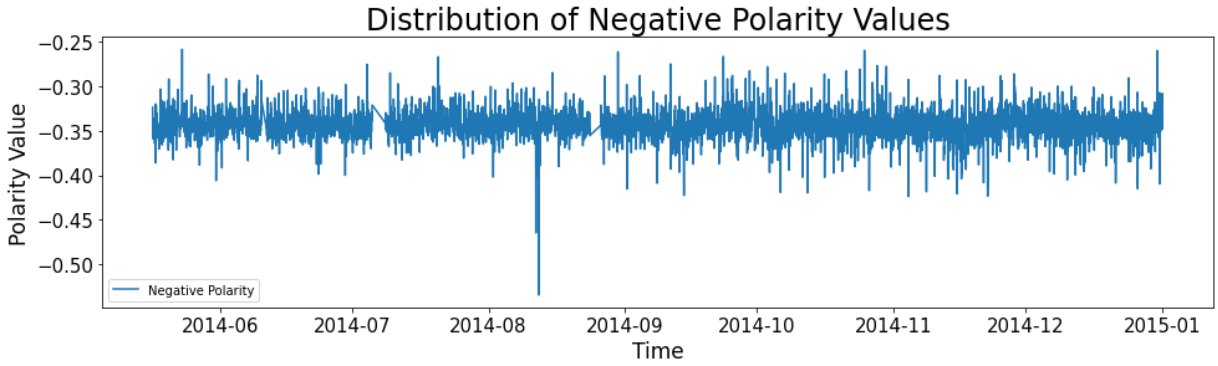

Fig 5(e): Negative Values Mostly Remain Stationary for the Entire Period

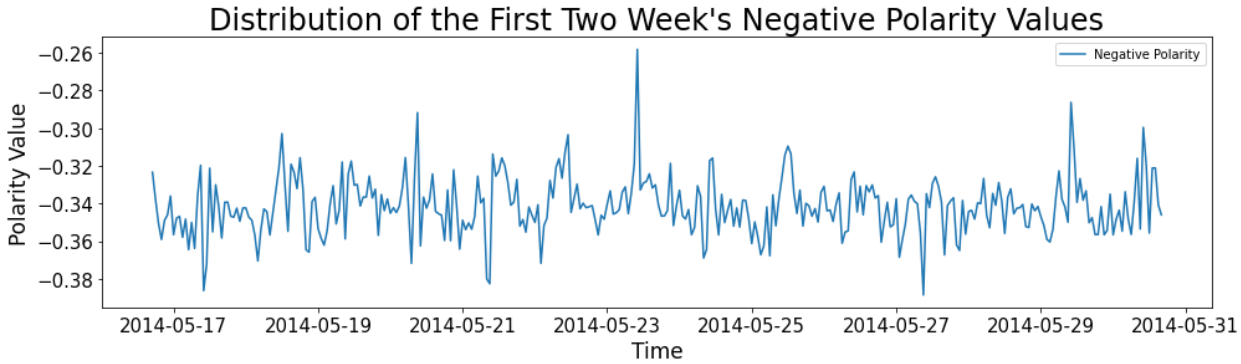

Fig 5(f): From the First Two Weeks of Data, there is no Visible Seasonality among the Negative Values

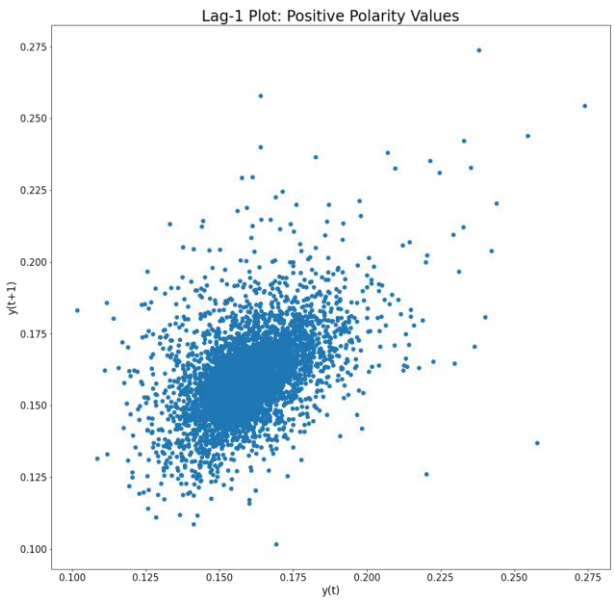

Fig 5(g): No (or weak) Correlation between Consecutive Positive Values

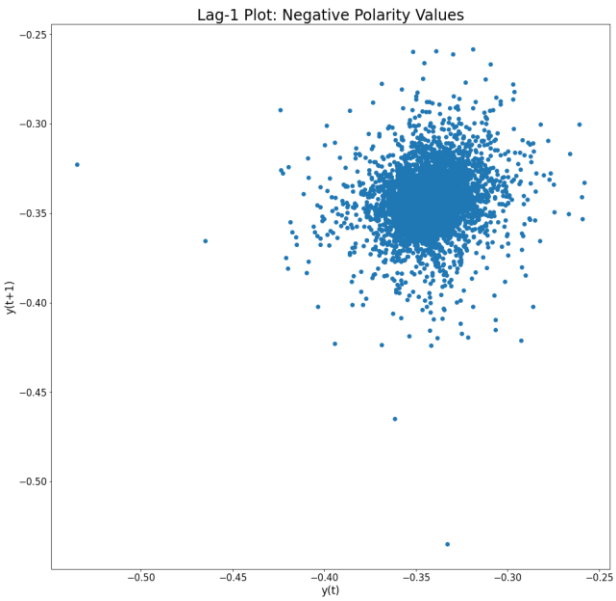

Fig 5(h): No Correlation between Consecutive Negative Values

The above set of plots show the values are following normal distributions (figure 5(a) and 5(b)), show how they are distributed over time, are there any trends or seasonality (figure 5(c), 5(d), 5(e) and 5(f)) and how the consecutive values are correlated (figure 5(g) and 5(h)).

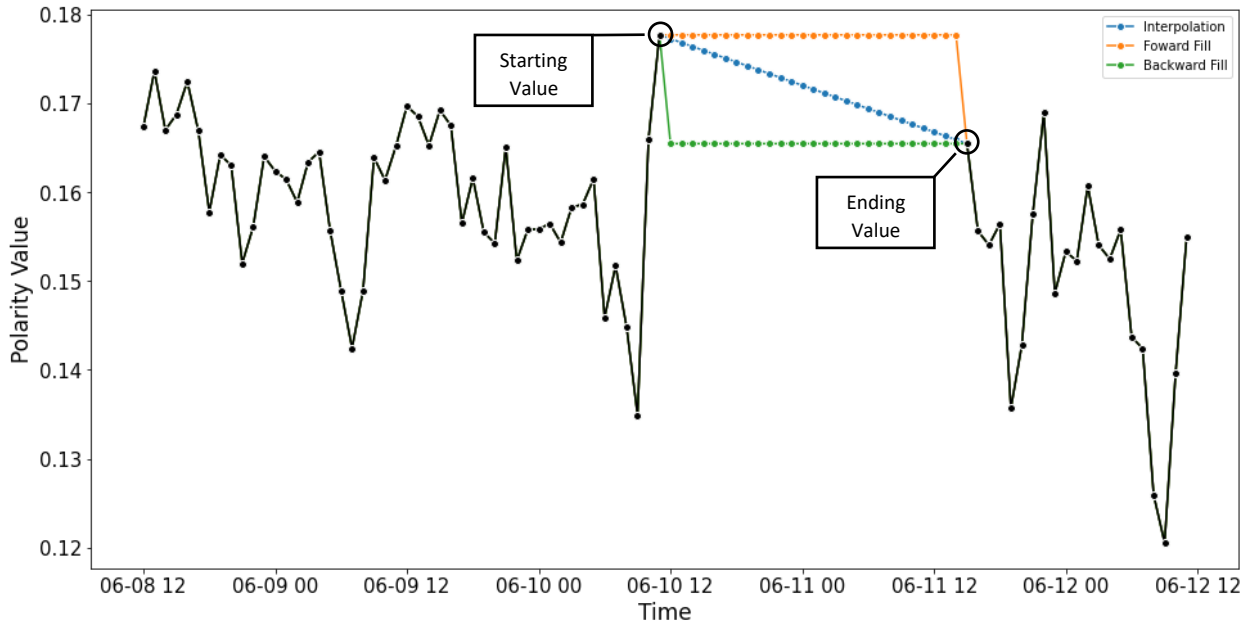

Fig 6: Visual Representation of How Interpolation, Forward, and Backward Fill Works

Upon searching for the missing timestamp after the aggregation process, I identified there are no values for both the positive and negative polarity values for 161 entries, meaning I do not have values for 161

hours. As mentioned above, I got 5,347 hours' worth of data from the source, and if we count the 161 entries where I did not get the data, our actual timeframe expands to 5508 hours. To mend the gaps created due to those missing values, I employed three techniques: Interpolation, Forward Fill and Backward Fill.

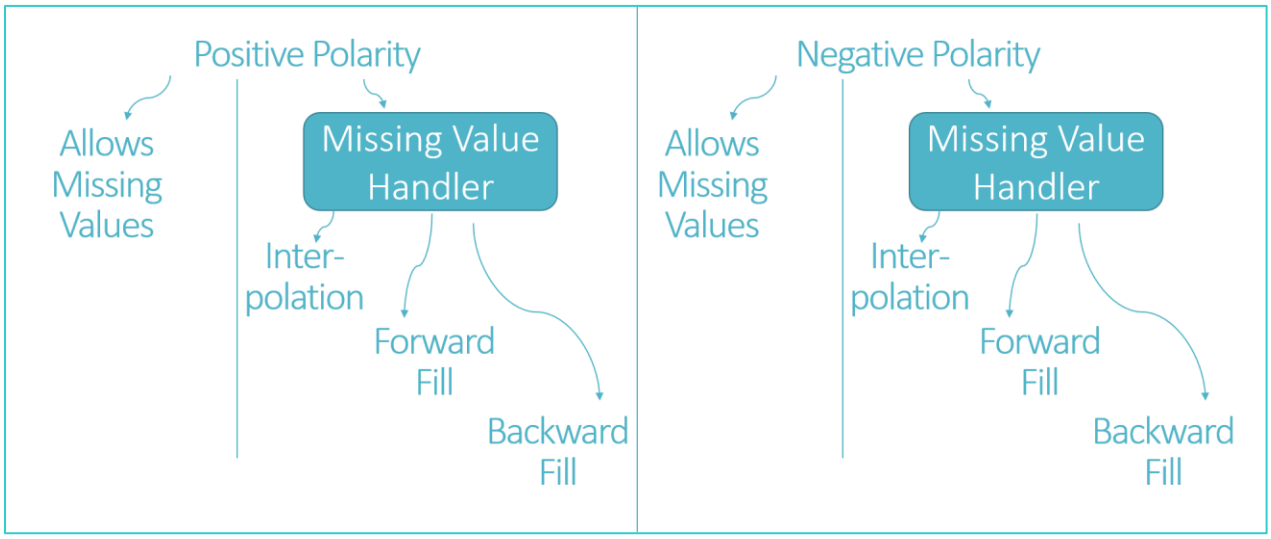

Fig 7: Data Groups based on how I handle Missing Values

Depending on the approach I choose to fill the gap, either starting value (after which the gap starts), ending value (where the gap ends), or both points to help estimate the missing values. I calculate the missing values for interpolation by drawing a line between the starting and ending values. For the other two cases, I use the starting and ending values as estimations for the forward and backward fill, respectively. Figure 6 shows a visual representation of how I handle the missing values. Along with the fill-up versions of the data, I also decided to keep the raw data (allowing missing values) while experimenting with different algorithms, resulting in eight different versions of datasets divided into two groups: positive values and negative values of polarity (shown in figure 7).

5. Experimental Setup:

From the distributions in figure 5, we see that the data does not show any stationarity or trends. However, to strengthen this claim, we passed both positive and negative sets through the Augmented Dickey-Fuller (ADF) test and Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test. We received 2.499448e-18 and 2.806902e-20 for the ADF test for positive and negative data, respectively. Since these values are less than the 0.05 significance level, the data is assumed to be stationary. KPSS also supports this claim.

We chose to execute three different sets of models to proceed with the experiments. We kept aside 20% of the data (trailing 20%) for each model to see how well each algorithm performs and used the beginning 80% of the data to design or train the models. I am listing the groups and models below.

5.1. Simple Models:

5.1.1. Naïve: I used the last observed value as the predicted value. For example, if the last value of the first 80% of the data is 0.175, I assume all the remaining 20% test data values would be the same as 0.175.

5.1.2. Average: Here, I took the average of a specified data window to predict future observations. Like Naïve, all the future value is assumed to be exactly the same here, too; instead of the last observed value of Naïve, I used average of the specified window here. I ran two iterations of the average method with one and four weeks of data as the window.

5.1.3. Seasonal Naïve: In Naïve, we used one value (the last observed) as the future prediction. In seasonal naïve, we used a set of values repeatedly. Again, this works with a specified window, and I used one and four weeks as the window for the two iterations that I executed.

5.1.4. Drift: Like the average and seasonal naïve method, drift requires a window, and I used one and four weeks of the window again. Drift uses the first and last values of the specified window to draw a fitting line that expands into the future; all the future observations are assumed to be the line value for each corresponding timestamp.

5.1.5. Rolling Average: It again ran it with one and four weeks window where the roll happens by one unit into the future as we predict future values one at a time.

5.2. Regression Models: I chose the previous 24 hours of data as the feature set of each observation for the regression models. A more systematic approach could have been applied to identify the number of features and their weights to design better-performing models. Different metrics, such as the Akaike information criterion (AIC), Bayesian information criterion (BIC), etc., could have been used to judge model performance and complexity. While discussing the research objective, I explained why I think the implantation of the different models could be limited, and thus I chose to keep it that way. A more comprehensive model design process would be implemented later with a more refined source data set. To make a fair comparison, I kept the test data the same as simple models (1080 entries; 20% approx.). Here is the list of regression models I used.

- Linear Regression
- Support Vector Machine
- K Neighbors Regression
- Decision Tree Regression
- Random Forest Regression

5.3. ARIMA Models: I used auto ARIMA on raw data, where I got ARIMA (3,0,4) as the optimal selection with a stepwise search and got ARIMA (5,0,0) while executing an exhaustive search. Both models have been executed on all the different data and bench marked to find the best performer later.

As for performance metrics, I used an extensive list so that we can evaluate all the different models thoroughly before concluding which performs better and for what case. I am listing all the metrics below.

- R-squared: R squared is the proportion of the variation in the dependent variable that is predictable from the independent variable
- MAE: Mean Absolute Error
- MAPE: Mean Absolute Percentage Error
- MSE: Mean Squared Error
- RMSE: Root Mean Squared Error
- NRMSE: Normalized Root Mean Squared Error
- ME: Mean Error
- MPE: Mean Percentage Error

Results and Analysis:

We experimented with each group: simple, regression, and ARIMA models to find out each group's best performer(s) and then put the group-wise best performers to identify who performs best overall. We considered the performance matrices to populate an exhaustive table for each group and summarize the results upon scanning that table.

	Algorithm	Data	R-squared	MAE	MAPE	MSE	RMSE	NRMSE	ME	MPE
0	Naive	Positive Data (allows missing values)	-0.022055	0.009715	0.059577	0.000201	0.014169	0.088956	0.002081	0.006045
1	Average of 1 Week	Positive Data (allows missing values)	-0.071682	0.010682	0.067842	0.000211	0.014509	0.091090	-0.003752	-0.030842
2	Average of 4 Week	Positive Data (allows missing values)	-0.052448	0.010492	0.066447	0.000207	0.014378	0.090269	-0.003210	-0.027410
3	Seasonal Naive 1 Week	Positive Data (allows missing values)	-0.462071	0.011612	0.073593	0.000287	0.016947	0.106395	-0.003911	-0.030231
4	Seasonal Naive 4 Week	Positive Data (allows missing values)	-0.809750	0.013033	0.082835	0.000355	0.018854	0.118371	-0.003378	-0.027403
...	...	...	...	...	...	...	...	...	...	...
67	Seasonal Naive 4 Week	Negative Data (missing value handled through b...	-1.059940	0.017541	0.051344	0.000589	0.024278	-0.070684	0.000471	-0.003716
68	Drift 1 Week	Negative Data (missing value handled through b...	-5.025398	0.034577	0.098966	0.001724	0.041522	-0.120888	-0.032516	0.092287
69	Drift 4 Week	Negative Data (missing value handled through b...	-0.376575	0.015545	0.046188	0.000394	0.019847	-0.057782	0.009369	-0.029708
70	Rolling Average 1 Week	Negative Data (missing value handled through b...	-0.197615	0.014083	0.041859	0.000343	0.018512	-0.053895	0.007484	-0.024267
71	Rolling Average 4 Week	Negative Data (missing value handled through b...	-0.026025	0.012620	0.037106	0.000294	0.017134	-0.049885	0.002491	-0.009696

Fig 8: A partial Snapshot of the Simple Models' Performances

```
Positive Data Group
R-squared => Naive => Positive Data (allows missing values) => -0.02205544329408138 => 0
MAE => Naive => Positive Data (allows missing values) => 0.009714643972222223 => 0
MAPE => Naive => Positive Data (allows missing values) => 0.059576890298754404 => 0
MSE => Naive => Positive Data (allows missing values) => 0.00020076185537795508 => 0
RMSE => Naive => Positive Data (allows missing values) => 0.014169045676331031 => 0
NRMSE => Naive => Positive Data (allows missing values) => 0.08895591115536865 => 0
ME => Naive => Positive Data (allows missing values) => 0.0020814274037037075 => 0
MPE => Naive => Positive Data (allows missing values) => 0.00604485290978223 => 0


Negative Data Group
R-squared => Naive => Negative Data (allows missing values) => -0.0013962515967456124 => 36
MAE => Naive => Negative Data (allows missing values) => 0.012403671787037036 => 36
MAPE => Naive => Negative Data (allows missing values) => 0.036148634112235366 => 36
MSE => Naive => Negative Data (allows missing values) => 0.00028653626757989503 => 36
RMSE => Naive => Negative Data (allows missing values) => 0.0169273821833116 => 36
NRMSE => Naive => Negative Data (allows missing values) => -0.0492827625480692 => 36
ME => Seasonal Naive 4 Week => Negative Data (allows missing values) => 0.00047102561111111253 => 40
MPE => Naive => Negative Data (allows missing values) => -0.0005772428986620852 => 36
```

Fig 9: Best performer of the Simple Models

We ran nine different experiments on the eight different datasets I have to populate a table of 72 entries for simple models. It records the name of the simple algorithms, the corresponding dataset used, and the metrics' values (see figure 8). Later, I passed that table to a function that finds out the best performers of each metric and reports the algorithm's name, the dataset used, value of the metric, and the associated index of the performance table. Figure 9 contains the outcome of the simple models. For both the positive and negative data, the naive method, where I used the last observation from the training set to predict or estimate future values, worked best regardless of data cleaning.

	Algorithm	Data	R-squared	MAE	MAPE	MSE	RMSE	NRMSE	ME	MPE
0	Linear Regression	Positive Data (allows missing values)	0.205066	0.008882	0.055549	0.000156	0.012496	0.078452	-0.000036	-0.005737
1	Support Vector Regression	Positive Data (allows missing values)	-2.125377	0.022145	0.144280	0.000614	0.024777	0.155557	-0.020432	-0.136308
2	K Neighbors Regression	Positive Data (allows missing values)	0.206979	0.008996	0.056534	0.000156	0.012481	0.078357	-0.000492	-0.008461
3	Decision Tree Regression	Positive Data (allows missing values)	-0.488800	0.012344	0.077518	0.000292	0.017101	0.107363	-0.000906	-0.011063
4	Random Forest Regression	Positive Data (allows missing values)	0.253046	0.008604	0.054027	0.000147	0.012113	0.076047	-0.000333	-0.007445
5	Linear Regression	Positive Data (missing value handled through i...	0.233192	0.008766	0.054842	0.000151	0.012273	0.077051	-0.000036	-0.005493
6	Support Vector Regression	Positive Data (missing value handled through i...	-2.125377	0.022145	0.144280	0.000614	0.024777	0.155557	-0.020432	-0.136308
7	K Neighbors Regression	Positive Data (missing value handled through i...	0.199829	0.009027	0.056777	0.000157	0.012537	0.078710	-0.000745	-0.010022
8	Decision Tree Regression	Positive Data (missing value handled through i...	-0.368408	0.011803	0.074128	0.000269	0.016395	0.102931	-0.000326	-0.007184
9	Random Forest Regression	Positive Data (missing value handled through i...	0.229014	0.008691	0.054536	0.000151	0.012306	0.077261	-0.000403	-0.007908
10	Linear Regression	Positive Data (missing value handled through f...	0.231673	0.008757	0.054761	0.000151	0.012285	0.077128	0.000060	-0.004809
11	Support Vector Regression	Positive Data (missing value handled through f...	-2.125377	0.022145	0.144280	0.000614	0.024777	0.155557	-0.020432	-0.136308
12	K Neighbors Regression	Positive Data (missing value handled through f...	0.205728	0.008995	0.056588	0.000156	0.012491	0.078419	-0.000696	-0.009720
13	Decision Tree Regression	Positive Data (missing value handled through f...	-0.364609	0.011756	0.073360	0.000268	0.016372	0.102788	-0.000055	-0.005314
14	Random Forest Regression	Positive Data (missing value handled through f...	0.253839	0.008583	0.053962	0.000147	0.012107	0.076007	-0.000540	-0.008734
15	Linear Regression	Positive Data (missing value handled through b...	0.224561	0.008799	0.055081	0.000152	0.012342	0.077484	-0.000153	-0.006267
16	Support Vector Regression	Positive Data (missing value handled through b...	-2.125377	0.022145	0.144280	0.000614	0.024777	0.155557	-0.020432	-0.136308

Fig 10: A partial Snapshot of the Regression Models' Performances

```
Positive Data Group
R-squared => Random Forest Regression => Positive Data (missing value handled through backward fill) => 0.260998412166099 => 19
MAE => Random Forest Regression => Positive Data (missing value handled through forward fill) => 0.00858276079131929 => 14
MAPE => Random Forest Regression => Positive Data (missing value handled through forward fill) => 0.05396173414479203 => 14
MSE => Random Forest Regression => Positive Data (missing value handled through backward fill) => 0.00014516172373449162 => 19
RMSE => Random Forest Regression => Positive Data (missing value handled through backward fill) => 0.012048307919973312 => 19
NRMSE => Random Forest Regression => Positive Data (missing value handled through backward fill) => 0.0756415240224702 => 19
ME => Decision Tree Regression => Positive Data (missing value handled through backward fill) => -1.784915833333338e-05 => 18
MPE => Decision Tree Regression => Positive Data (missing value handled through backward fill) => -0.004590603360155674 => 18


Negative Data Group
R-squared => Linear Regression => Negative Data (allows missing values) => 0.03668087674874554 => 20
MAE => Linear Regression => Negative Data (allows missing values) => 0.012144387023861503 => 20
MAPE => Linear Regression => Negative Data (allows missing values) => 0.03539900730034151 => 20
MSE => Linear Regression => Negative Data (allows missing values) => 0.00027564100187575376 => 20
RMSE => Linear Regression => Negative Data (allows missing values) => 0.01660243963626291 => 20
NRMSE => Linear Regression => Negative Data (allows missing values) => -0.048336717482474006 => 20
ME => Decision Tree Regression => Negative Data (missing value handled through interpolation) => -2.408819381687252e-05 => 28
MPE => Random Forest Regression => Negative Data (missing value handled through interpolation) => -0.00022177452749580885 => 29
```

Fig 11: Best performer of the Regression Models

In the same way, I populated the table for regression (figure 10) and ARIMA (figure 12) models and identified the best performers (figures 11 and 13) of each of those two groups. I used five regression models with all eight data, which resulted in a performance table of 40 entries, and used two ARIMA models to populate the corresponding table for ARIMA models. While experimenting with regression

models on positive data, it seemed that handling missing values using forward fill worked in favor of prediction when I used Random Forest.

Negative data could be used as-is, but we should use interpolation if we want to handle the missing values. As an algorithm, if we filled the data using interpolation, we should use Random Forest; however, raw data combined with linear regression worked the best.

	Algorithm	Data	R-squared	MAE	MAPE	MSE	RMSE	NRMSE	ME	MPE
0	ARIMA(5,0,0)	Positive Data (allows missing values)	0.000384	0.009754	0.060640	0.000196	0.014013	0.087974	-0.000076	-0.007596
1	ARIMA(3,0,4)	Positive Data (allows missing values)	0.013218	0.009705	0.060355	0.000194	0.013922	0.087407	-0.000068	-0.007491
2	ARIMA(5,0,0)	Positive Data (missing value handled through i...	0.000085	0.009768	0.060765	0.000196	0.014015	0.087987	-0.000181	-0.008261
3	ARIMA(3,0,4)	Positive Data (missing value handled through i...	-0.000908	0.009772	0.060788	0.000197	0.014022	0.088031	-0.000164	-0.008154
4	ARIMA(5,0,0)	Positive Data (missing value handled through f...	-0.002150	0.009699	0.060032	0.000197	0.014030	0.088085	0.000607	-0.003281
5	ARIMA(3,0,4)	Positive Data (missing value handled through f...	-0.003177	0.009703	0.060054	0.000197	0.014038	0.088131	0.000622	-0.003187
6	ARIMA(5,0,0)	Positive Data (missing value handled through b...	-0.004463	0.009889	0.061825	0.000197	0.014047	0.088187	-0.000963	-0.013204
7	ARIMA(3,0,4)	Positive Data (missing value handled through b...	-0.005395	0.009893	0.061831	0.000197	0.014053	0.088228	-0.000922	-0.012946
8	ARIMA(5,0,0)	Negative Data (allows missing values)	-0.003200	0.012417	0.036154	0.000287	0.016943	-0.049327	-0.000946	0.000338
9	ARIMA(3,0,4)	Negative Data (allows missing values)	-0.003085	0.012416	0.036151	0.000287	0.016942	-0.049324	-0.000947	0.000342
10	ARIMA(5,0,0)	Negative Data (missing value handled through i...	-0.004648	0.012427	0.036162	0.000287	0.016955	-0.049363	-0.001141	0.000907

Fig 12: A partial Snapshot of the ARIMA Models' Performances

```
Positive Data Group
R-squared => ARIMA(3,0,4) => Positive Data (allows missing values) => 0.013218352308989934 => 1
MAE => ARIMA(5,0,0) => Positive Data (missing value handled through forward fill) => 0.00969859124038049 => 4
MAPE => ARIMA(5,0,0) => Positive Data (missing value handled through forward fill) => 0.06003222843433288 => 4
MSE => ARIMA(3,0,4) => Positive Data (allows missing values) => 0.00019383304080340395 => 1
RMSE => ARIMA(3,0,4) => Positive Data (allows missing values) => 0.013922393501241227 => 1
NRMSE => ARIMA(3,0,4) => Positive Data (allows missing values) => 0.08740738280174641 => 1
ME => ARIMA(3,0,4) => Positive Data (allows missing values) => -6.82529276440361e-05 => 1
MPE => ARIMA(3,0,4) => Positive Data (missing value handled through forward fill) => -0.0031868451083417073 => 5


Negative Data Group
R-squared => ARIMA(3,0,4) => Negative Data (allows missing values) => -0.0030851919844088194 => 9
MAE => ARIMA(3,0,4) => Negative Data (allows missing values) => 0.012415964413373547 => 9
MAPE => ARIMA(3,0,4) => Negative Data (allows missing values) => 0.036151354462914946 => 9
MSE => ARIMA(3,0,4) => Negative Data (allows missing values) => 0.0002870195354911483 => 9
RMSE => ARIMA(3,0,4) => Negative Data (allows missing values) => 0.01694165090807706 => 9
NRMSE => ARIMA(3,0,4) => Negative Data (allows missing values) => -0.0493243048354037 => 9
ME => ARIMA(5,0,0) => Negative Data (allows missing values) => -0.0009455727651168185 => 8
MPE => ARIMA(5,0,0) => Negative Data (allows missing values) => 0.00033759766903675956 => 8
```

Fig 13: Best performer of the ARIMA Models

For positive data, ARIMA (3,0,4) performs better with raw data, whereas ARIMA (5,0,0) benefits when we use forward fill to handle missing values. For negative data, ARIMA (3,0,4) performed well with raw data.

To summarize the group-wise performance comparison, I list the best performers below.

On positive values:

- Simple: Naïve with any of the four data
- Regression: Random Forest with Forward Fill
- ARIMA: ARIMA(3,0,4) with raw data and ARIMA(5,0,0) with Forward Fill

On negative values:

- Simple: Naïve with any of the four data
- Regression: Random Forest with Interpolation
- ARIMA: ARIMA (3,0,4) with raw data

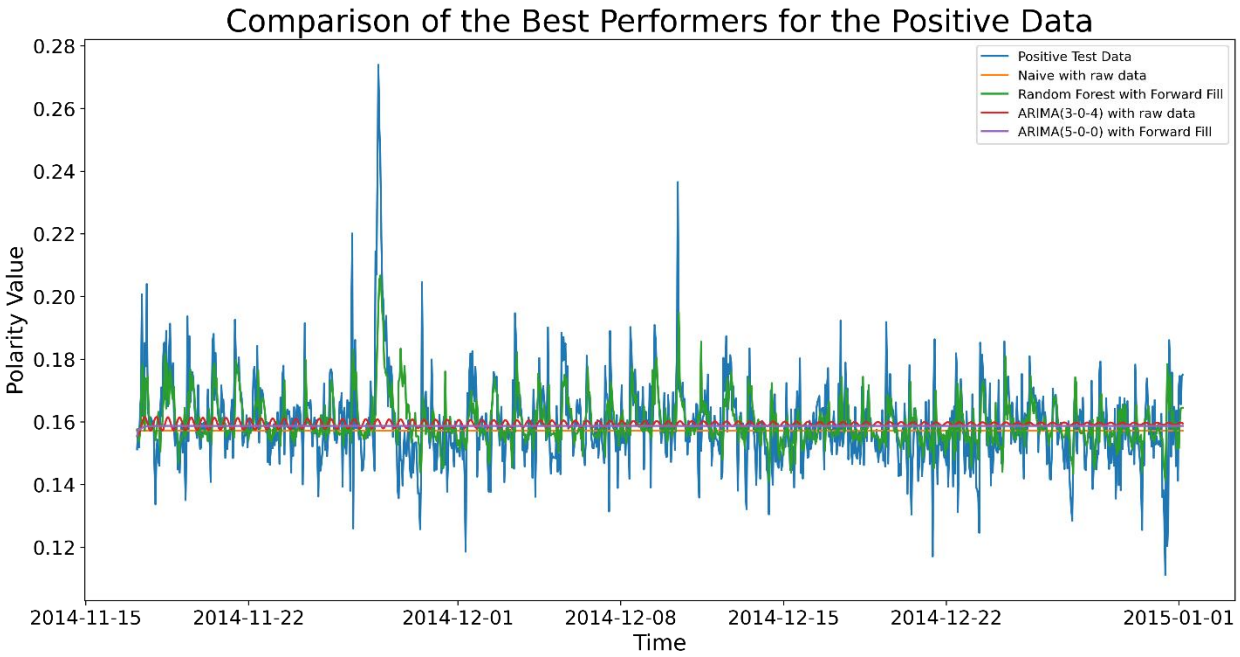

Fig 14: Random Forest Seems to Follow the Test Data Patterns More Closely

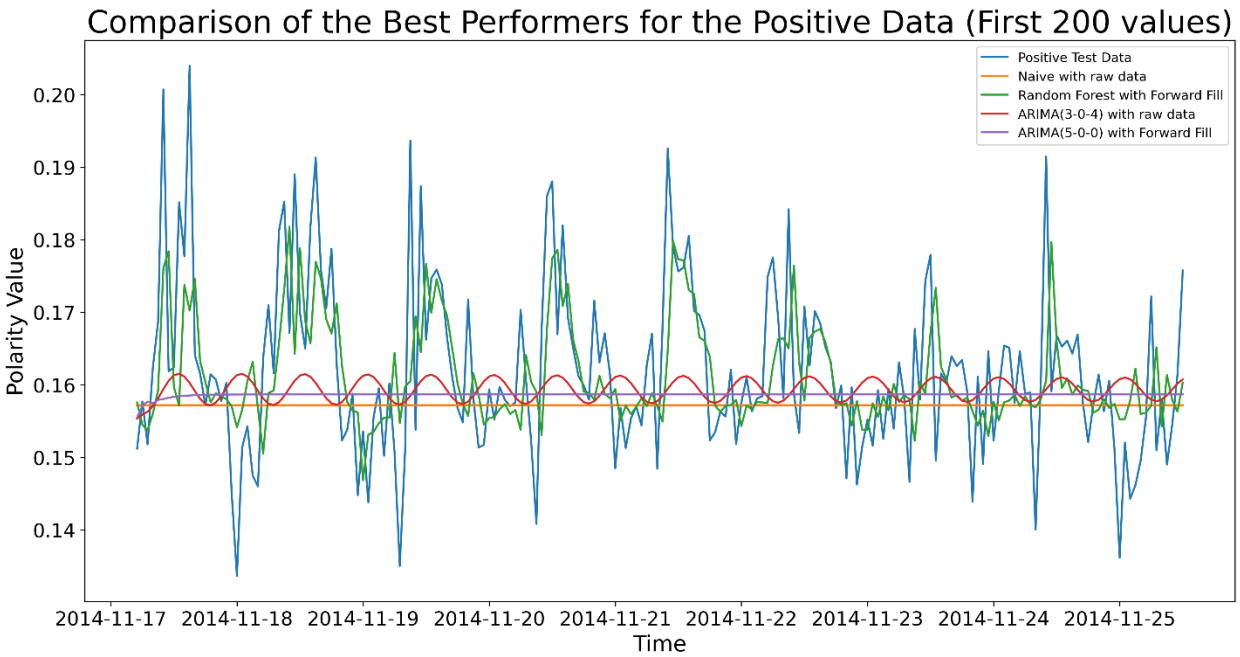

Fig 15: Subsection highlighting First 200 Values Confirms the Relatively Better Performance of Random Forest

Algorithm	R-squared	MAE	MAPE	MSE	RMSE	NRMSE	ME	MPE
Naive	-0.02206	0.009715	0.059577	0.000201	0.014169	0.088956	0.002081427	0.006044853
Random Forest	0.253839	0.008583	0.053962	0.000147	0.012107	0.076007	-0.000540228	-0.008733587
ARIMA(3-0-4)	0.013218	0.009705	0.060355	0.000194	0.013922	0.087407	-6.83E-05	-0.007490592
ARIMA(5-0-0)	-0.00215	0.009699	0.060032	0.000197	0.01403	0.088085	0.000606572	-0.003281259

Table 1: Comparison of the Best Performers of the Positive Data

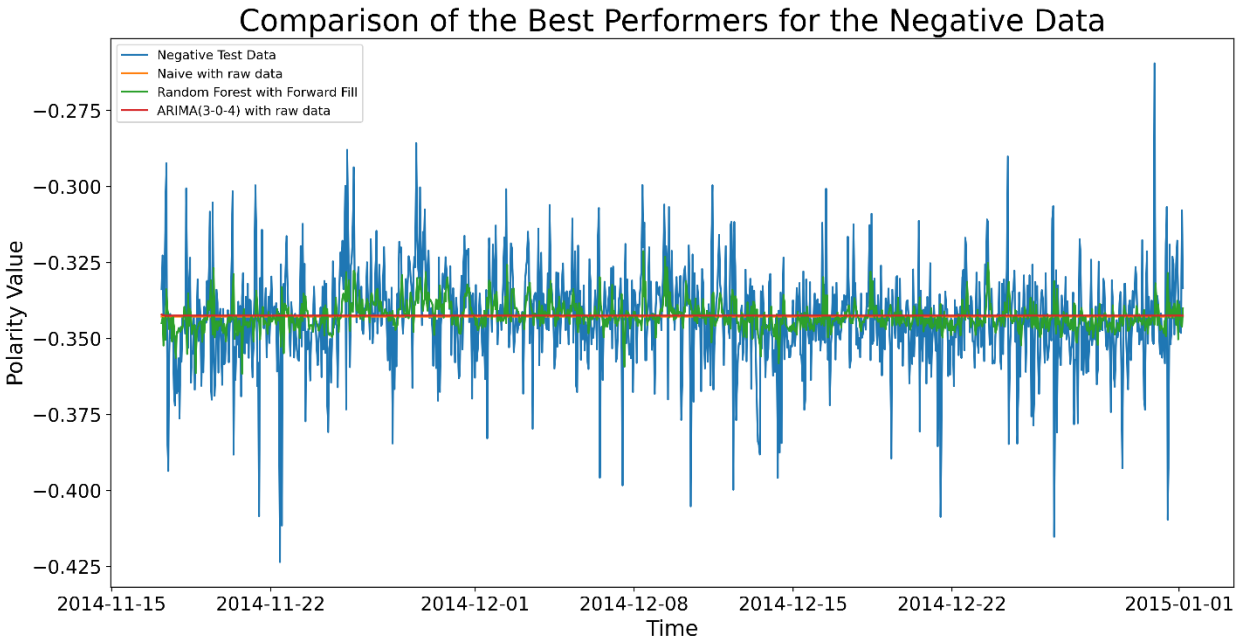

Fig 16: Comparison between the Best Performers for Negative Values

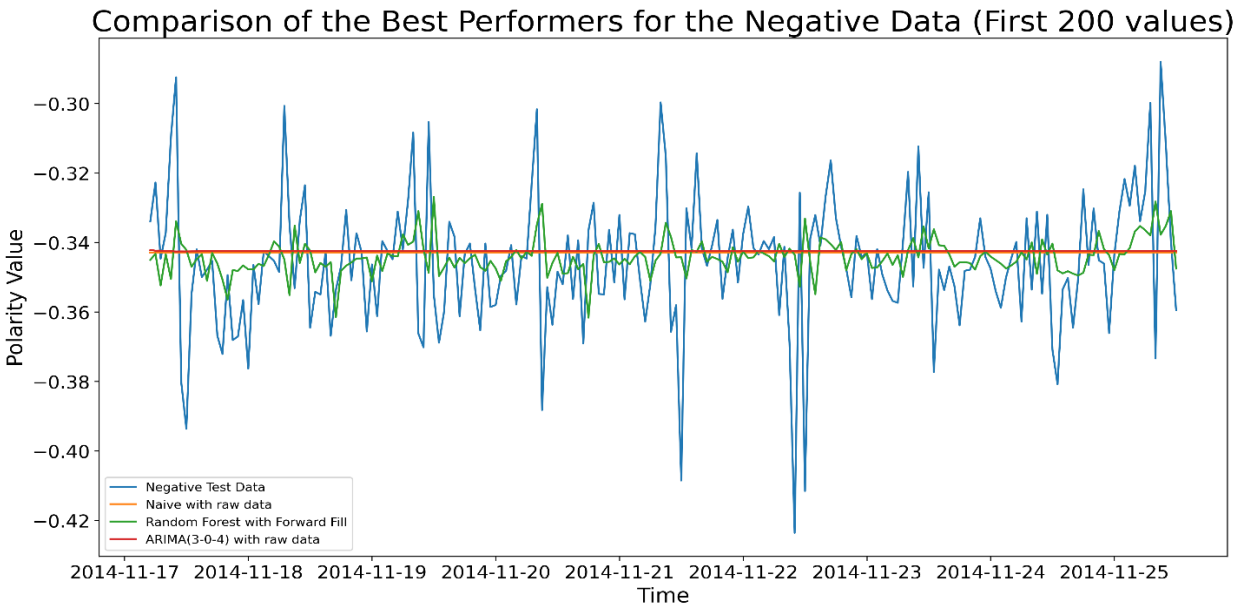

Fig 17: Zoomed-in Version of Comparison to Better See How Well They Perform

Algorithm	R-squared	MAE	MAPE	MSE	RMSE	NRMSE	ME	MPE
Naive	-0.0014	0.012404	0.036149	0.000286536	0.016927	-0.04928	-0.000632075	-0.000577243
Random Forest	-0.00495	0.012292	0.035729	0.000287552	0.016957	-0.04937	-0.000720464	-0.000221775
ARIMA(3-0-4)	-0.00309	0.012416	0.036151	0.00028702	0.016942	-0.04932	-0.000946872	0.000341534

Table 2: Comparison of the Best Performers of the Negative Data

To identify the best performer from each group, I plotted them to visually compare (see figures 14, 15, 16, and 17). I put the performance metrics' values in a tabular format (follow tables 1 and 2), highlighting the overall best performer. Random forest performs better for positive data when I apply forward fill to address the missing values. For negative data, there is a competition between the Naïve model with raw data and the random forest with interpolation, where Naïve seems to outperform the random forest. The naïve model predicts future values as a straight line, so I would conclude that I failed to pick an algorithm that captures the pattern for the negative values.

6. Concluding Remarks and Future Works:

Rather than finding an appropriate set of algorithms to predict future values, this implementation serves as an outline and prototype for my secondary objectives, where tweets grouped by topics and demographic characteristics of the users will be fed into the model. Comparison between different geographic areas, applying sophisticated algorithms such as neural networks, using other predictors for the regression models, and designing and weighing the number of predictors appropriately will be performed in the future, where I see this project's work as a stepping stone. In summary, rather than comparing the results of this project, I would like to take this as an opportunity to expand the research horizon.