Data is most important to us. We do our best to find data from reliable and reputable sources.

We use APIs and JSON files to create around 80% of all our data. There was manual intervention needed at some points. The following is how we captured the data and the data captured.

***Quantitative Company Data Wrangling***

We acquired the qualitative data from a single API found at polygon.io. We captured the data by running Get\_Stock\_Price.ipynp. There were two years of data we were able to capture and write to a single comma-delimited file. After this we did add a column by computing the percent change from opening and closing prices. Finally, we imported all data into the ticker\_daily\_stat.

***Fields Captured***

* + Open
  + Close
  + High
  + Low
  + Volume
  + Volume Weight
  + Number of Transactions
  + Percent Change

***Qualitative Company Data Wrangling***

The first thing we did was to capture a listing of the top 100 Nasdaq companies. This data was readily available in several places out on the web. We did compare sources and make sure we had the most current source of the top 100 companies. After that, it was our belief we needed more qualitative data. We were able to capture company facts from <https://companies-datas.p.rapidapi.com> and port the data to a JSON file. From this source, we used company size and employee number to add to our qualitative data. We also used Wikipedia to capture sector information. All this information was organized and imported into the company\_info table.

***Fields Captured***

* + Ticker
  + Company URL
  + Revenue
  + Employee Count
  + Sector

***Stock’s Data***

We were able to capture longitude and latitude company headquarters by using an API at nominatim.openstreetmap.org via coordinate\_acqusition\_FINAL.ipynb. This data was imported into the table ticker\_location along with other fields we manually looked up.

***Fields Captured***

* + Ticker
  + City
  + State
  + Country
  + Longitude
  + Latitude

***Map’s Regional Data***

Thankfully we were able to find regional longitude, latitude GeoJSON file from <https://eric.clst.org/tech/usgeojson/>. This site offered several regional files to download. options. We chose the smallest file present for US States. We had to do some adjusting by adding the REGION field to the file. To be able to use them was a little tricky. We did have to create a repository for all mapping data we created. We downloaded gz\_2010\_us\_040\_00\_500k.json and added a column in the properties section named REGION. After that we were able to utilize it to create the maps.

***Daily and Monthly Stock Changes***

Both the daily and monthly stock JSON was created using the company\_all\_star (joined) table. The processes needing to run were json\_time\_v2 and random\_forest\_30\_day\_json\_v1. The output files were monthly\_json\_new3.json and daily\_stock\_map3.json. Don’t forget to mention we published our results for the map’s data at <https://github.com/kjkubik/ProjectJSONStockInfo>

Using a combination of the following tools and some manual searching we were able to collect information to populate our datasets:

Polygon.io - where all stock prices were retrieved for two years

RapidAPI.com - The API used to get all companies data was - "companies-datas"

finance.yahoo.com - Some sectors found

Geopy - python library used to get Long. and Lat., as well as locations

we created to join all the tables together. We used the company\_all\_star table

We got the regional map from: https://eric.clst.org/tech/usgeojson/ . We downloaded gz\_2010\_us\_040\_00\_500k.json and added a column in the properties section named REGION. When we fixed it, we named it UnitedStatesRegion.json (edited)

son\_time\_v2 => ../resources/monthly\_json\_new3.json

[10:17](https://utmccvirtdata-9gs4974.slack.com/archives/C035P6G6868/p1648955822879779)

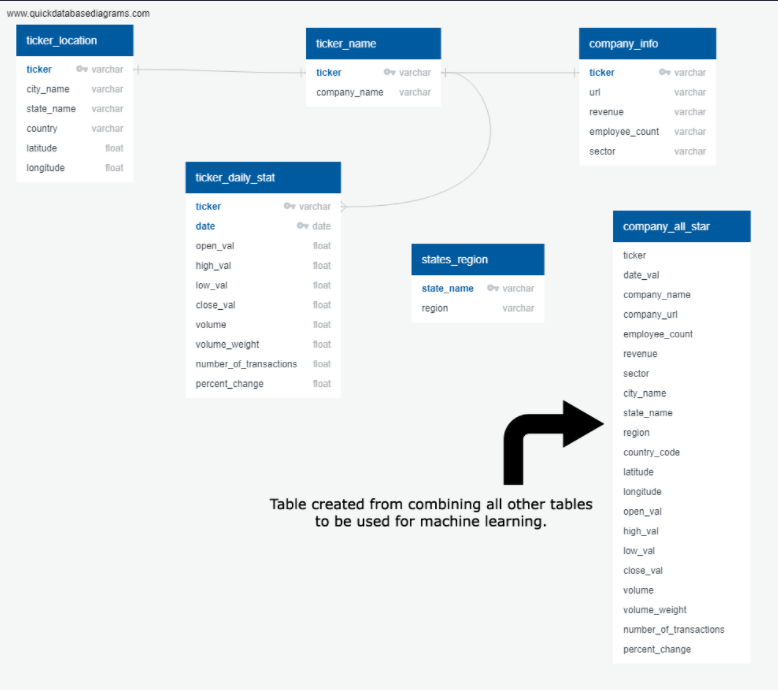
random\_forest\_30\_day\_json\_v1 => ../resources/daily\_stock\_map3.json

[10:19](https://utmccvirtdata-9gs4974.slack.com/archives/C035P6G6868/p1648955974846459)

We got the regional map from: <https://eric.clst.org/tech/usgeojson/> . We downloaded gz\_2010\_us\_040\_00\_500k.json and added a column in the properties section named REGION. When we fixed it, we named it UnitedStatesRegion.json (edited)

[10:19](https://utmccvirtdata-9gs4974.slack.com/archives/C035P6G6868/p1648955997216929)

This is where our data came from for our maps.



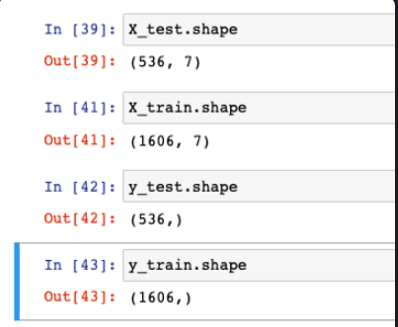
We were able to utilize nominatim.openstreetmap.org to gain the longitude and latitude data we need for each company. The processing we used to wrangle the data can be found in the getting\_courdinates folder.

Preprocessing of Model Data

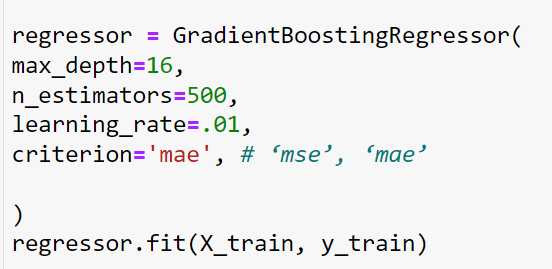
We initially had 2 years of quantitative data consisting of open, close, high, low, volume, volume weight and percent change to begin with for the top 100 Nasdaq stocks. The qualitative data consisted of the revenue, employee count and sector. The locational data consisted of city, state, regional and the company’s home office longitudes and latitudes. We used a query to join the data into one table before exporting the data into a comma delimited file.

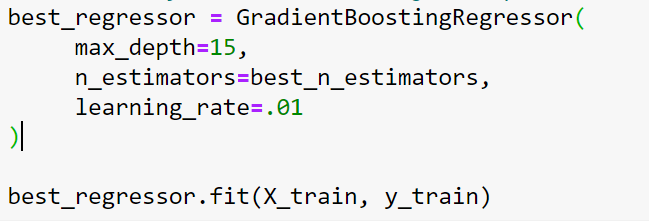
In the preprocessing we moved the comma delimited file into a dataframe. From there we used interval controls by capturing the first and last day of an interval, computing the percent change in the volume weight and volume weight and merging the first and last day into a signal dataframe. We then created the categorical data and dropped the columns that were of no use for modeling.

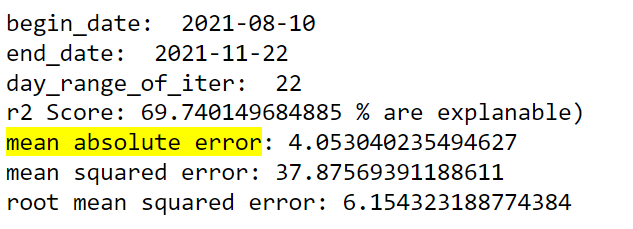
After that we did our preprocessing, we started training our data. The target for our model is the percent change in volume weight. The reason we chose this as our target is because when buying and selling stock, it is more important than the actual price of the stock. The shape of our training and test data is as follows:



To train the Gradient Boosting Decision Tree, I used the following settings.





The results of the Gradient Boosting Decision Tree are as follows:   


The **r2 score** is expressed as a percentage. The score of 69.74 means that the percentage change of the volume weight prediction is at most 69.74% accurate. However, just because the results are explainable doesn’t mean 69.74% accurate. We only have 69.74% of the picture.

One of the things that we did notice is that the r2 score was high if we picked dates where it was obvious the stock market in an upward or downward trend. The results below show us that the market is in an upward trend between those dates. It is very possible to get r2 scores as low as 32%.



Because of these inconsistencies, we believe that using more stock metrics would up the scores. We would have love to add these to our data and if we did have time, we would have.

tkr\_current\_annual\_earnings

tkr\_price\_to\_sales\_ratio

tkr\_price\_to\_earnings\_ratio

tkr\_price\_to\_book\_ratio

tkr\_debt\_to\_equity\_ratio

tkr\_FCF

tkr\_PEG\_ratio

tkr\_operating\_expenses

tkr\_capital\_expences

tkr\_have\_dividends

tkr\_have\_share\_buybacks

tkr\_earnings\_per\_share

tkr\_value

tkr\_payout\_ratio

tkr\_beta

tkr\_return\_on\_equity

tkr\_compound\_annual\_growth\_rate

We also started a Long Short Term Memory Network model but ran out of time.

As for the **mean absolute error** of 4.05. From what we could find, this may mean that on average, the forecast's distance from the true value.

The **mean squared error** and **root mean squared error** are related. That is, the square root of 37.87 is 6.15. It is the arithmetic average of the absolute errors |ei| = |yi – xi|, where yi is the prediction and xi the true value. It kind of tells us how are we are off; but it not exactly meaningful presently.