USING THE BLOOMBERG TERMINAL TO EVALUATE STOCK MARKET TRENDS: AN ANALYSIS OF HISTORICAL EPS FORECAST ACCURACY FOR ALL FIRMS IN THE 2019 S&P 500 INDEX

BY
MARC ANGELO ACEBEDO

A Thesis

Submitted to the Division of Natural Sciences
New College of Florida
in partial fulfillment of the requirements for the degree
Bachelor of Arts in Computer Science
Under the sponsorship of Dr. Matthew Lepinski

Sarasota, Florida
May 2020

Acknowledgments

Throughout my four and a half years as a New College student, I cannot count on my fingers the number of inspiring people I was lucky to have met, networked with, and made friends with for life. After becoming involved in the fields of Natural Sciences, Chinese Language & Culture, Islamic Studies, and Spanish Language & Culture, as well as having spent a semester abroad in the United Arab Emirates, I am proud to say that I formed connections with people from various walks of life, all with very diverse interests and perspectives.

First and foremost, I would like to thank my thesis sponsor, academic advisor, and thesis committee member **Dr. Matthew Lepinski**. He has been a supportive professor since my first day as an undergraduate, when I was a newbie to the industry of computer science.

I took classes with my other thesis committee members, **Dr. Tania Roy,** and **Dr. Melissa Crow.** Thank you, Dr. Roy, for encouraging us to do the hard work needed to see stellar results. I also thank Dr. Crow for being a great and helpful professor, and she was a source of invaluable help in handling the statistics portions of this thesis.

Dr. Gary Kalmanovich is pretty much who I consider the "dad" professor. When I first walked into his office, I immediately knew that I would like this guy. He was always encouraging but also was realistic and stayed grounded in all his reasoning. Transparency is a trait I admire, and Dr. Kalmanovich always made sure that I knew my shortcomings so I could grow from them. I would also like to thank **Dr. John Doucette** for being my advisor for most of my time at NCF.

Once upon a time, I was pursuing a double-major in Computer Science and Physics. I took beginner physics courses with **Dr. Mariana Sendova**, and I thank her for our cooperation over our code-based physics projects. I took an ISP and a tutorial with her where I had to transcribe physics concepts, equations, and ideas into code. We also bonded over conversations regarding cultural backgrounds and traditions.

The friends I made at NCF have been invaluable. I have met many people throughout the past four years, but the following people have left a noticeable impact on my life. I would like to thank **Marty Chenyao** for being his unapologetic, assertive queer self. We have countless nostalgic memories of our time as students *and* afterwards, regarding our reckless excursions. I enjoy all our conversations and "accountabilibuddy" meetings.

One of my other friends, with whom I have spent a lot of time studying and working, is **Serena Bonci**. Thank you for the late-night excursions to Steak n' Shake along with our existential conversations. I see her as both the "technically savvy" and "idea generator" friend. We would go to some cafe late at night and work until 4:00am, and I can attribute most of my productivity hours to her company. I thank **Franklin** for joining my beta reader circle and for being your opinionated weird self. There is never a conversation we have that does not get boring. Our topics of conversation can ruffle some feathers in public, but that is a big reason why I enjoy your company—it never becomes dull. I also thank **Gustavo** for all our fun outings, rants, and dealing with some *interesting* people. Whenever

I feel like shitting on American imperialism with somebody, I know you are the person to turn to.

I thank **Diana** for those memories of going to hackathons, cooking maqluba and layali lebnaan, and dealing with my less-than-stellar driving. I also thank **YoungKeun** for his exemplary practicality—his problem-solving skills are exceptional, and I enjoy all our conversations, whatever they may be about. Even after graduation, we remain good friends. I also thank **Rozana** for being an amazing friend. We had many days where we "played" soccer, which is pretty much just me aimlessly kicking a ball around. Together with her, we founded the ***Arabic Language Club*** and she was a great co-instructor to work with.

Jumping from the heat of Florida to the ice-capped mountains of Colorado, I stayed good friends with a few people from high school. I thank **Sheila, Oasis, Jasmine,** and **Hannah** for our monthly (sometimes yearly) group calls, and for putting equal effort into keeping in touch after graduating high school. Though we harmlessly jab at each other from time to time, I can call you all friends for life. I also thank **Alessia** for some of the most unforgettable high school years—of going to concerts, running, and doing literally *everything* together—we pretty much lived at each other's houses. Eight years later, our friendship remains strong.

From the USA all the way to the UAE, I have all my professors at the ***American University of Ras al-Khaimah (AURAK)*** to thank. **Dr. Khouloud Salameh** was a pleasant, results-oriented professor to work with and always made sure I did my best in her classes. Thank you, **Dr. Steven Zani,** for always being a reliable point of contact and for your endless empathy for your students. You helped me adjust to a completely different culture in a short amount of time. Thank you, **Dr. Bryn Holmes**, for the lifelong connections you helped me make, our conversations about different cultures, and for introducing me to all your adorable cats and one special dog.

Ms. Ghada Mohammad Al-Kadri was instrumental in kick-starting my career in technology. As the Senior Officer for the International Office of ***AURAK***, she did more than just connect me to the right people to find my first internship: she did her best to make me feel welcome during my first week, and was always a reliable person to turn to. In addition, she helped arrange trips to Oman, Dubai, and Abu Dhabi, and her badassery and exemplary work ethic are very respectable traits about her. Additionally, I sincerely thank her for helping me establish my first-ever presence in the tech industry—my first internship being halfway across the world from my hometown.

I thank **Antonino Chu Benitez,** for being my partner in crime and always being one of my main sources of inspiration. Seriously, I cannot even begin to list all the reasons how you have impacted me—my life would be much different without you. Likewise, I thank his brother **Paolo Benitez,** his grandfather **Joselito Chu**, and his parents, **Mon and Lalaine Benitez**, for exposing me to the world of entrepreneurship and accepting me as part of the family while I lived in the UAE (I basically lived in their house for my last two weeks). They founded a magazine called ***Illustrado***, a successful Gulf-based magazine with the mission of enhancing the image of ***OFWs*** (Overseas Filipino Workers) in the Middle East.

As strong visionaries, they demonstrated that anybody can execute big ideas as long as they are fearless and take constant action.

My friends in the UAE, who come from various countries, are also people I know I will remain friends for life. **Asma** for being the first person to show me around the unfamiliar city of ***Ras al-Khaimah***, along with your inspirational outlook over seeing technology as a tool to positively impact people. Your stubbornness, genuine care for all people, and your enthusiasm for life are unmatched.

Thank you, **Ammar**, for being your authentic and progressive self. Our trips to various joints in Ras al-Khaimah have been some of the most unforgettable memories. Thank you **Wajeeha** and **Bahja** for your outspokenness and opinionated outlooks on life—the minute we met at Global Day, we instantly clicked and never ceased to have the most interesting conversations since then.

Thank you, **Mona, Faye, Saker, and Zainab** for being my first friends in the UAE. I cannot count the amount of memories we have together—of frequenting the malls, the Louvre in Abu Dhabi, downtown Dubai, the Dubai Mall, water-skiing, among many others. Thank you **Esmaeel** for being the *chico loco*. Through Saker, I met **Tanvi** with whom I instantly clicked the moment we met, and we share many similar viewpoints and outlooks on life. I enjoy all my conversations with Tanvi, whatever they may be about. I also thank **Omar** for your open-mindedness and being my language-exchange buddy. Your enthusiasm for learning another language is inspiring.

I end with my family. I thank my brother **Manolito (Toto) Acebedo** and my sister-in-law **Andrea Beltran**. Additionally, I thank my sisters **Angelika** and the "ettes" **Linette, Yvette, and Joyette Acebedo.** I thank all of them for being there my entire life, regardless of distance and troubles, and for flying out to visit us occasionally, when you can.

Salamat sa lahat ng pamilya. Sa mga magulang ko **May** at **Manuel Acebedo.** My parents' endeavors of forming their own successful band, along with my mom's online clothing business, have shown me that effort and relentlessness are all that counts. Their actions proved to me that having no expectations for success is paradoxically what leads to success.

Table of Contents

USING THE BLOOMBERG TERMINAL TO EVALUATE STOCK MARKET TRENDS: AN ANALYSIS OF HISTORICAL EPS FORECAST ACCURACY FOR ALL FIRMS IN THE 2019 S&P 500 INDEX

Marc Angelo Acebedo

New College of Florida, 2020

ABSTRACT

This thesis is divided into three sections documenting the process of cleaning, exploring, and interpreting the EPS and EOD data I gathered.

The first section is ***data wrangling***, where I explain how I gathered, assessed, and cleaned my data for quality and tidiness issues. During the assessment stage, I check for missing data (null values), duplicate data, and incongruous data types. Once satisfied with my clean CSVs, I stored them under the ***./data/clean*** directory.

The second section focuses on ***data exploration.*** This section is further separated into 3 stages: the univariate, bivariate, and multivariate exploration stages. I re-import the cleaned data, generate visualizations, make observations, and answer my 5 main research questions.

Lastly, I focus on ***data explanation***, where I narrow down my conclusions in a Jupyter Notebook Slides presentation. For each conclusion, I include all relevant visuals and graphs. This is the "storytelling" process of data analysis.

All of the code for this project can be found on my Github repository at https://github.com/nihlan97/Evaluate-Historical-Stock-Market-Forecasts. The project is also hosted on my technical publication site, https://webbyboy.com/.

To view the Jupyter Slides presentation of all my findings, download the reveal.js HTML file in my Github repository: **data_explanatory.slides.html.** Afterwards, render the file in the browser of your choice.

Dr. Matthew Lepinski

Division of Natural Sciences

Introduction

The stock market is a dynamic structure which directly mirrors the country's current economic state. Whether the current economy is experiencing a crisis or expansion, the stock market's performance depends largely on the participation of professional traders and investors, who make financial decisions as a response to sudden shifts in the current economic climate. On a wider scale, the stock market directly reflects the plethora of decisions that traders make, to adapt to while taking advantage of the current financial climate. Economic literacy, as vague a term as it sounds, is a term that has been contested without a concrete definition, and basic financial literacy is crucial for understanding market dynamics.

In my definition, economic literacy means one's level of competency in using basic economic concepts to make decisions covering a variety of fiscal matters: personal finance, budgeting, investing, and to understand trends and patterns—the causes and effects to why the current economy is in the state it's in. Despite being a country founded on capitalistic foundations, the US holds an adult financial literacy rate of 57%.[1] Described as an "epic fail," the financial literacy rate in the US falls noticeably short.

After looking at the above statistic, I decided to investigate the gaps in my own financial knowledge. I then thought of the stock market, in which I had never been an investor. And since I had little familiarity with the inner workings of the stock market, I decided to dedicate this thesis project to a field of business, finance, and economics that I was not familiar with. The book *A Beginner's Guide to the Stock Market* by Matthew R. Krater has

[1]Iacurci, Greg. "Financial Literacy: An Epic Fail in America." Investment News, March 2, 2019.

greatly piqued my interest in stock market trading—particularly the investment of dividend stocks. I quickly downloaded Robinhood and started a trading portfolio. I decided that to further educate and familiarize myself with market dynamics, creating this thesis project as an end to my senior year at NCF would be a perfect opportunity to explore this new interest in the stock market. Not only would I be familiarizing myself with stock market terminology, I would also be working directly with stock market data, which would help me build a deeper intuition in any future endeavors investigating stock market issues and finance-related projects.

I decided that there was no better way to build intuition with stock market patterns and terminology than to directly analyze stock market data. To narrow my focus, though, I picked historical forecast data as my primary variable of interest. More specifically, I wanted to investigate the historical accuracy of price forecasts for the 2019 S&P 500 firms in the past 20 years, from 1999 - 2019.

The reason why I picked the S&P 500 versus other market indices like the Dow Jones and the Nasdaq is because the S&P 500 comprises of the top 500 largest publicly traded companies.[2] This higher number of companies will make it easier to draw connections between these firms' trends and more general stock market trends. I hope that my conclusions covering the accuracy of Bloomberg forecasters in the past 20 years will help people develop a healthy critical eye, as well as intuition, when evaluating historical data, current trends, and future forecasts.

[2] Jackson, Anna-Louise. "S&P 500 versus the Dow Jones Industrial Average: What's the Difference." Acorns, Dec 18, 2019.

This thesis describes in detail how I used the Bloomberg Terminals to extract the EPS and EOD data I needed, then how I cleaned all the noisy data to generate clean visuals that ultimately tell a story. In all data analysis projects, it is the end product that provides the most value: a clear and coherent story extracted from initial sources of distortion, ambiguity, and noise.

Literature Review

My main motivation for this project is to analyze the levels of consistency between stock market predictions and actual price returns. However, I also wanted to see how investor sentiment could impact and even significantly drive the stock market into a certain direction. Does investor sentiment act as an indicator of public reaction to stock tips?

I hypothesize that relying on historical trends to predict future stock returns is largely unreliable and useless, considering that the stock market is a largely dynamic entity. How much evidence will I find to support that statement? Lastly, how wide is the difference in predictability between short-term and long-term price returns? Which machine learning models and approaches are the most effective?

These questions were interesting to broach and explore. Firstly, RK, Dase and DD, Pawar recite a literature review detailing the various uses of artificial neural networks for stock market predictions.[3] They found that artificial neural networks were particularly useful for predicting world stock markets, which I felt made this article more universal due to its approach of analyzing global stock markets than just ones focused in the US. The principal

[3] Dase RK and Pawar DD, "Application of Artificial Neural Network for Stock Market Predictions: A Review of Literature," *International Journal of Machine Intelligence* 2, no. 2 (2010): 14-17.

strength of a neural network is its ability to find patterns, irregularities, and detect multi-dimensional non-linear connections in data,[4] which means it is extremely useful for modeling dynamic systems. Since the stock market is inherently a dynamic entity, it makes perfect sense to use an artificial neural network. This supports my initial reasoning that predicting stock market prices based on historical trends is largely unreliable; they respond only to current factors that drive company stocks up or down.

Among the literature that Dase RK and Pawar DD listed and summarized, the one that piqued my interest the most was the approach given by Heping Pan, Chandima Tilakaratne, John Yearwood (15) in 2005. They created a basic neural network with limited optimality to investigate the Australian stock market index AORD and achieved correctness in a directional prediction of 80%. This goes to show that even a neural network with bare minimum optimality still returned a high accuracy rate in prediction. This raises another question: how would this compare to other neural networks with limited optimality - among the other sources is there a general trend of favoring artificial neural networks? As a matter of fact, the simplest answer would be yes: there is a large favorable image for using artificial neural networks. Qing Cao, Karyl B Loggio, and Marc J. Schniederjans (26) concluded in 2005 that their NNs outperformed all their linear models, with statistically significant results across a few sample firms. Jinyuan Shen, Huaiyu Fan, and Shengjiang Chang (18) used a tapped delay neural network (TDNN) in 2007 (tapped delay neural network), and results showed their optimized neural network model not only greatly reduces calculating complexity, but also improves prediction precision.

[4] Dase RK,

All this literature paints a largely favorable image of artificial NNs. This paper concludes that artificial neural networks can predict the stock index as well as whether it is best to buy, hold, or sell shares of stock at any given situation.[5]

Baker, Malcolm and Wurgler, Jeffrey focus on investor sentiment in the stock market, which ties directly to my prior motivation.[6] They concluded that waves of sentiment have clearly discernible, important, and regular effects on individual firms and the stock market as a whole.[7] This conclusion supports my conjecture that sentiment not only influences the stock market, but holds a novel foothold in its functions and patterns.

I acknowledge that this is a large scope, as trying to find data on mass reactions would be largely scattered. For example, of course there would be opinion articles and reactions, social media comments, and discussions, but there also exist personal sentiments and sudden, impulsive reactions by investors/traders (e.g. somebody frantically shorting all their shares in response to news of a company's predicted bankruptcy). In these cases, *how* would one define a "reaction:" qualitatively through social media and the Internet, and/or quantitatively through impulsive financial decisions? Multiple resources for mass reactions to a single piece of stock news would have to be carefully gathered, making this aim a resource-intensive one. In that case, my prior motivation of attempting to measure the impact of not only investor sentiment, but investor *reaction*, is highly unlikely to be feasible for automation. Additionally, various complex methods would be needed to quantify emotion from vastly different sources.

---

[5] Ibid.
[6] Malcolm Baker and Jeffrey Wurgler, "Investor Sentiment in the Stock Market," *Journal of Economic Perspectives* 21, no. 2 (2007): 129-151.
[7] Baker, "Investor Sentiment,"

Atsakalis, George and Valavanis, Kimon implement an Adaptive Neuro Fuzzy Inference System (ANFIS) to predict short-term trends.[8] This methodology relies on historical data to create predictions of stock market trends. Their ANFIS method utilizes stock market process model inputs, which then are used to determine the stock market process model inputs that give the best stock prediction for the next day. These are calculated in terms of the minimum Root Mean Square Error (RMSE), and after using this system on the New York Stock Exchange (NYSE) and American Stock Exchange (ASE), discovered that this approach is far superior to the "buy and hold" method.

When using historical data, Atsakalis and Valavanis found that accurate predictions of stock market trends are achievable.[9] This is an immensely helpful source because it serves as counterevidence to the hypothesis that relying on historical trends is undependable. This finding supports the conjecture that drawing conclusions from historical trends is ***not always*** reliable for price forecasting. In all cases, historical trends do not ***necessarily*** exist as an infallible method for forecasting methodologies. However, considering how highly dynamic the stock market is, depending on historical trends is still an approach I would not be entirely comfortable pursuing. Yes, short-term historical trends may be useful for predicting the stock price index of a certain company that is obviously spiraling toward bankruptcy or reaching new 200-day moving average highs, but I'm coming from the lens of where *we do not know the current track of the company in question.* Therefore, despite

[8] Atsakalis, George S. and Valavanis, Kimon, "Forecasting Stock Market Short-Term Trends Using a Neuro-Fuzzy Based Methodology," *Expert Systems with Applications* 36 (2009), 10696 - 10707.
[9] Atsakalis and Valavanis, "Forecasting Stock Market,"

the promising results shown in this source, I would still not use this approach in the ***average case.***

Sankaraguruswamy, Srinivasan examines whether or not market-wide investor sentiment influences stock price sensitivity to firm-specific earnings news.[10] The main parameter of interest used in their evaluations is the Earnings Response Coefficients (ERC), and they test the hypothesis that the ERC of good (bad) earnings news is higher (lower) when sentiment is high instead of low, leading to a positive (negative) association between the ERC of good (bad) news and investor sentiment.[11] As seen, stock price sensitivity to bad earnings is higher during low sentiment periods than high sentiment periods, and vice versa. This means that the more pessimistic investors tend to be, the more likely the stock price will be sensitive to bad earnings. On the flip side, the more optimistic investors are, the more likely the stock price will be sensitive to increased earnings, and the effect of sentiment on stock price sensitivity is significantly worse for bad news than good news. This data portrays a heavy implication that investor sentiment directly influences stock market sensitivity. Sentiment influence is especially pronounced for these: small stocks, young stocks, high volatility stocks, non-dividend-paying stocks, and stocks with extremely high/low market-to-book ratios.

Sankaraguruswamy concludes that stock price reactions to earnings news contain both rational and irrational components.[12] For example, relying solely on short-term market

[10] Sankaraguruswamy, Srinivasan, "Investor Sentiment and Stock Market Response to Earnings News," *The Accounting Review* 87, no. 4 (2012): 1357 - 1384.
[11] Sankaraguruswamy, "Investor Sentiment,"
[12] Ibid.

reaction for the valuation impact of accounting information can be insufficient.[13] This partly proves my conjecture right: investor sentiment and reaction to earnings news cannot always be reliable.

To integrate more complex methodologies, Picasso, Andrea, et al. use 2 approaches for a technical analysis along with sentiment embeddings for predicting market trends.[14] Their first approach is technical, where their model is based on mathematical indicators constructed on stock prices. Their second technique is the fundamental approach, where they exploit information from news, profitability, and macroeconomic factors. The combination of these two approaches falls exactly in line with my motivation; in both cases, a novel combination of both social media and economic models is utilized.

Machine learning techniques were used in this model for time series prediction and sentiment analysis. A predictive model was constructed to predict the trend of a portfolio forecasted by the top 20 companies listed in the NASDAQ100 index, where a high frequency trading simulation was run.[15] More than 80% of annualized return was reached. The final evaluation was performed in two steps: firstly by evaluating the statistical behavior of the classifier, then testing the effectiveness of the model's predictions.[16] The only difference is that the researchers are testing the classifier's ***fields and state*** vs its ***behavior*** that could show faults in statistical analysis and the results that come afterward.

[13] Ibid.
[14] Picasso, Andrea, et al., "Technical Analysis and Sentiment Embeddings for Market Trend Prediction," *Expert Systems With Applications* 135 (2019): 60 - 70.
[15] Ibid.
[16] Ibid.

The outcome of the model could predict both positive and negative trends in the portfolio of stocks under examination, which implied that it could recognize the most meaningful changes in market trends *and* achieve positive returns during trading simulations. The main news sources used to extract sentiment embeddings were the Loughran and McDonald dictionary and AffectiveSpace 2.[17] It was discovered that using Loughran and McDonald led to higher values of annualized returns while AffectiveSpace 2 resulted in achieving more high accuracy values. Therefore, Loughran and McDonald is better for making decisions on which stocks to invest in, while AffectiveSpace 2 is better suited in objectively analyzing stock patterns for accuracy.

Ultimately, the comparison between the feature sets of sentiment embeddings, price, and technical indicators far out-performs the use of the Price set only. This over-performance shows that the sole use of news is not outstanding.[18] This is a novel result. My motivation is proven correctly; reaction to the news is more important than the news itself. Along with price technical indicators, relying on sentiment embeddings in stock market news makes for a strong model.

This work establishes a ***solid archive*** for future collaborations between technical and fundamental approaches to market prediction. The summary by Picasso, Andrea, et al. poses an interesting aim I had not initially considered in my prior motivation. I did not consider the fact that sentiment analysis and machine learning techniques were opposing approaches, and I was unknowingly taking part in this divide by focusing *only* on sentiment analysis of stock market news. Instead of just relying on sentiment analysis taken from

[17] Ibid.
[18] Ibid.

stock market news, it is important moving forward to see how statistical tests and machine learning infuse together with the sentiment analysis approach itself.

Sun, Yuan, et al. examined 22,504 tweets from Sina Weibo, a microblogging site, to see how they influence the stock market.[19] Two types of microblog users' influence on the stock market were analyzed. The results show that an inverse U-shaped curve relationship existed between stock return and news/media/investor attention, implying that news media attention has a positive moderating effect between investor attention and stock return.[20] In extension, the positive effect of social interaction on news media and investor's sentiments on stock return was realized.

These discoveries help narrow my scope from "reaction to news" to higher specificity: through tweets, news media, and investor attention. They give a more concise approach to stock return and general reaction to news by categorizing which audience's reactions they are on the lookout for; the reaction to the news on the part of the *investors* themselves is really what matters in the end.

Sun, Yuan, et al. found that both investors and the news media have a great impact on stock returns.[21] This conclusion helps answer an initial question I had: when referring to "reaction to the news" is it preferable to treat investors and news media differently? How about collectively? What about other audiences such as bloggers who write opinion articles, and what about impulsive investor/trader reactions? How do they feed into the general classification of "reaction/reactors to the news?" Ultimately, this paper helped me

[19] Sun, Yuan, et al., "How Mood Affects the Stock Market: Empirical Evidence from Microblogs," *Information and Management,* July 26, 2019,
[20] Yuan, "How Mood Affects the Stock Market,"
[21] Sun, Yuan, et al.

understand that it is more beneficial to divide reactions into those categories of audiences, as that helps to paint a more layered picture.

The researchers in this paper state that "investor attention" is traditionally defined through price limit, trading volume, advertising expenditure, and media reports.[22] They reference Da, Engelberg, and Gao, who proposed a new direct measure of investor attention through aggregate search frequency in Google. However, this approach is not rigorous enough, as people could be searching for stocks for various reasons (e.g. to write a report, mere curiosity, and many searchers aren't even investors). This paper comes to remedy that faulty approach: Sun, Yuan, et al. propose ***classifying potential investor groups*** from a large number of Weibo users.[23] The underlying belief for this motivation is that comments or investment strategies published by investors on Weibo are direct evidence of investor attention. Therefore, investor attention is measured by the ***number of microblogs that investors posted.***

Social media significantly facilitates the amount, speed, and range of information transmission.[24] This is because social media reveals the opinions and perspectives of a wider range of audiences, unlike before its advent where it would be more difficult to navigate around investor sentiment as that would largely require gathering primary sources in print or in person. Lastly, they have found that microblogs became the ***center*** for collecting public opinions, which I imagine would be due to the ease in its accessibility to

[22] Sun, Yuan, et al.
[23] Sun, Yuan, et al.
[24] Ibid.

the average user. Therefore, relying on blogs is an effective measure of measuring investor reaction to stock market news.

Chapter 1: Preliminary Data Wrangling

A) Overview

All datasets collected consist of the 505 firms found in the 2019 S&P Index, with EPS and EOD data encompassing 20 years: from January 1999 until December 2019.

The broad question I pose in my research process is: *how do price forecasts for each firm in the S&P 2019 index compare to their corresponding actual prices?* In order to address this question, I narrowed down my focus to these four factors: Forecasted Earnings-per-Share (EPS), Actual EPS, Actual End-of-Day (EOD) Price, and Forecasted EPS 3 months prior to the current fiscal term.

In order to properly approach this question, my approach was to analyze the **difference in means** (also see: prediction error) between actual and forecast EPS for each firm. That is, the raw prediction errors (actual - forecasted EPS) among all 505 firms I gathered, as well as separating their average prediction errors by year, quarter, and twenty-year averages.

After gathering, cleaning, assessing, and storing the EPS and EOD data, I generated these two final CSVs. Here is a breakdown of the column features for my final clean CSVs: **features.csv**, **avgs.csv,** and **firms.csv.**

features.csv

- ***firm_id***

- ***feature***
- ***date***
- ***term***
- ***value***

avgs.csv

- ***firm_id***
- ***average***
- ***average_type***
- ***time_period***

firms.csv

- ***firm_id***
- ***firm***

The ***firm_id*** values refer to their corresponding foreign keys in **firms.csv**. A breakdown of each final CSV's features will be summarized in the Data Exploration stage.

B) Defining Our Variables

Before delving into our data, let's define End-of-Day Price and Earnings-per-Share more closely.

EPS stands for ***Earnings per Share.*** The formal definition of EPS given by Investopedia is this:

> *Earnings per share is the portion of a company's profit that is allocated to each outstanding share of a common stock, serving as an indicator of the company's financial health.*

EPS represents a portion of the company's **net income** after all of their dividends have been compensated for. Dividends are profits paid out to shareholders of the company, and this makes EPS one of the most valuable financial measurements because it ***determines a stock's worth.***

In other words, EPS is a value that tangibly summarizes a company's performance and success. The higher the stock, the more the company is able to pay dividends out to its shareholders, and thus the more net profit they are determined to generate.

$$EPS = \frac{Net\,Income - Preferred\,Dividends}{Weighted\,Average\,Common\,Shares\,Outstanding}$$

On the other hand, EOD stands for the ***End of Day*** price. On any given day, the EOD marks the ***price at which the stock was last valued***: at the end of the day's trading period.

C) Approach

I analyzed quarterly price returns within the past 20 years for the firms present in the S&P 500 2019 index—a twenty-year period extending from 1999 until 2019. At first, my intent was to analyze the forecasted and actual EPS of the S&P in its entirety for the past 20 years.

However, after considering that firms continuously enter and leave stock indices every year, my approach would show varying levels of inconsistencies and marginal errors when comparing annual S&P returns alone. In order to combat this problem, I have isolated these two approaches:

- Analyze the historical earnings of *only* the firms present in the S&P 2019 Index
- Keep track of all firms that were present in the S&P for the past 20 years. Keep track of the number of times each firm appeared in the Index and for those with the least count, analyze them individually on how they differ from the firms that stayed longer in the Index.

In the end, I have chosen to narrow my strategy to the first approach: analyze the historical earnings of *only* the firms present in the S&P 2019 Index at the time I gathered the EPS data. In other words, I cannot simply refer to the S&P 2019 Index as a singular entity to analyze trends from 1999 - 2019. Instead, I isolated the **505 firms** present in the S&P 2019 **at the time I gathered them**, and then analyzed specifically those 505 firms over a 20-year period.

D) Bloomberg Excel Functions

In order to gather the data I need, I needed to use the Bloomberg Terminals hosted at the University of South Florida Sarasota-Manatee (USF). The USF campus contains a lab where the Bloomberg Terminals are hosted, and only through Bloomberg was I able to gather the specific financial data for this project. Through using in-built Excel functions that query Bloomberg data, I gathered the following data of interest. I gathered historical

data for both earnings-per-share (EPS) and end-of-day (EOD) stock price for each firm in the 2019 S&P Index for the past 20 years, from January 1999 to December 2019.

For the EPS data, I gathered the following three datasets:

- Historical ***forecasted*** EPS by quarterly fiscal period ***dependent*** on a forecast made 3 months prior to the current term
- Historical ***forecasted*** EPS by quarterly fiscal period ***not dependent*** on a previous date
- Historical ***actual*** EPS for each firm by quarterly fiscal period.
- Historical ***actual*** EOD Price for each firm

For the bulk majority of the data, I focused on the fiscal quarters of each firm. Each quarter is defined as Q1, Q2, Q3, and Q4, which are not uniform since these EPS forecasts and actual values rely on a ***fiscal calendar*** instead of a calendar one. However, the only variables where I utilized the calendar period instead of fiscal period was with EOD prices and forecasted EPS made 3 months prior to the current fiscal period.

For those features relying on calendar period instead (EOD and 3-month-prior EPS), the calendar quarters are defined as follows:

- **Q1:** January - March
- **Q2:** April - June
- **Q3:** July - September
- **Q4:** October - December

It is not enough to just get data from one of the quarterly periods without specifying a date during which the forecasted EPS was made. This is why for the first dataset, I chose the projected earnings estimated three months before each quarterly term. For example, Q3 for 2005 starts in July. To get the EPS for Q3 of 2005, I would rely on the forecast made three months before, on April 1st.

I chose this 3-month time window because it would be intriguing to see the fluctuations of forecasted EPS between the period of the current quarter and the beginning of the last quarter; that is, to what degree does the time gap between EPS predictions affect those very predictions? Do estimated price-earnings fluctuate in the time windows between the beginning of each fiscal period? These are questions to explore later.

For EOD data, I only gathered historical actual EOD Price for each firm over the past 20 years in March, June, and September.

Chapter 2: Data Wrangling

A) Gathering the Data

End-of-Day Price (Actual)

Though EOD Prices do not contribute directly to my main research purposes, I decided it was still an interesting variable to keep for the sake of visualization and enriched analysis. To gather historical end-of-price data for each firm, I used this Bloomberg Excel function:

```
= BDH (equity, last_price_field, start_date, end_date)
```

For example, if I wanted to see the end of day price for a single unit of Apple's (AAPL) stock on 31 March 2005, I would construct the formula in a single Excel cell accordingly:

```
= BDH ("AAPL UW Equity", "PX_LAST", "03/31/2005", "03/31/2005")
```

The dates I use are all consistent to the end of each calendar period:

- 31 March
- 30 June
- 31 December

Now I just repeat the above formula for all remaining 505 firms in the 2019 S&P Index. In addition to querying the Bloomberg Help Desk to gather my data, I also learned to use the Bloomberg Excel BDH function[25] to isolate the members of the 2019 S&P Index. Afterwards, I retrieved all historical data for *only* those firms.

After isolating these 505 firms using the previous function, it was fairly easy and straightforward to gather the rest of the historical data.

Earnings-per-Share (3 months prior)

To gather historical forecasted EPS by quarterly fiscal period dependent on a forecast made 3 months prior, I used the following formula:

[25] "Tutorial 1: Downloading End of Day Price Data for S&P 500 Stocks." *MTSM Bloomberg Lab Wordpress Blog*, June 8, 2017.

```
= BDH("AAPL US Equity", "BEST_EPS", "7/1/2007",
"BEST_FPERIOD_OVERRIDE=08Q1", "days=a", "fill=p")
```

Earnings-per-Share (Forecasted)

This is a central value to measure for the question of comparing historical forecasting accuracy across various firms. EPS Forecasts are made based on a firm's ***fiscal period.***

```
= BDP("AAPL UW Equity","BEST_EPS","BEST_FPERIOD_OVERRIDE = 00Q1")
```

Earnings-per-Share (Actual)

Actual EPS is also measured based on ***fiscal period***, which means there is timely consistency between our forecasted and actual EPS data.

```
= BDP("AAPL UW Equity","IS_EPS", "FUND_PER=Q1",
"EQY_FUND_YEAR=2000")
```

Each CSV was structured as follows: I assigned all 505 firm tickers as column headers, and each row value directly corresponded to a date/time period.

I stored each CSV under my /data/ directory as follows. I also include their number of rows and columns:

Feature to Investigate	File name	Number of columns	Number of rows
EOD Price	sp-eod-act.csv	506	84
Forecasted EPS	sp-eps-fc-csv	506	84
Actual EPS	sp-eps-act.csv	507	84
Forecasted EPS (3 months prior)	sp-eps-fc-terms.csv	507	80

Summary of all original EPS and EOD datasets before cleaning

B) Assessing the Data

The main technologies used in the data assessment stage consist of these main Python libraries: Pandas and NumPy. The reason I used Pandas is due to the library's main feature of turning features into DataFrames—that is, Pandas gave me a way to directly manipulate the original CSV's data without altering any of the original contents. Any entire CSV could be condensed into a singular 2D DataFrame table object, which could then be manipulated and even combined with other DataFrames.

However, Pandas functionality was still not enough; NumPy provided further functionality, particularly for performing mathematical and logical operations on arrays and all other similar abstract data types. The NumPy library comes equipped with its own version of an array—the NumPy array—whose benefits include ***smaller memory consumption*** and ***better runtime behavior***.[26] Since I was dealing with hundreds of thousands of rows of data across four different CSVs, NumPy was the perfect library for efficiently manipulating

[26] "Python Lists vs. Numpy Arrays - What is the Difference?" University of Central Florida.

data across all DataFrames in a way that did not eat away at memory, and in a way that was quick and efficient.

All four CSVs contain data for each firm across various dates. The raw data depicts each firm's EOD price or EPS at any given point in time: a year, a quarter, and/or a date. I figured this would be a great opportunity to not only take advantage of the raw data, but to also generate a new CSV accounting for all firm ***averages*** on a yearly, twenty-year, and quarterly basis.

To begin my data assessment stage before data cleaning, I imported all 4 CSVs and assigned them to DataFrame variables like below:

CSV file	**Pandas DataFrame variable**	**Price Type**
sp-eps-fc.csv	*eps_fc*	Forecasted EPS
sp-eps-act.csv	*eps_act*	Actual EPS
sp-eod-act.csv	*eod_act*	Actual EOD Price
sp-eps-fc-terms.csv	*eps_fc_terms*	Forecasted EPS (3 months prior)

Check for Missing Data

In order to verify whether the number of non-null data entries would weaken my later analysis, my main motivation was to find the number of NaN (not-a-number) entries across all DataFrames.

But firstly, just to establish a clearer overhead of all my data, I looked for the number of *rows* and *columns* that each CSV contained.

For *eps_fc*, there are 505 firms encompassing 84 quarterly fiscal periods since 1999. In total there are 506 columns: 1 column being **Term Forecast**, with the rest being firm tickers.

For *eps_act*, there are 505 firms encompassing 84 quarterly fiscal periods since 1999. There are 507 columns: 2 columns being **Quarter** and **Year**, the rest being firm tickers. I plan to merge the "Quarter" and "Year" fields into a singular Pandas Period object with a YYYYQQ format. (e.g. 2012Q4 represents the 4th fiscal period in 2012).

For *eod_act*, there are 505 firms encompassing 84 quarterly calendar periods since 1999. There are 506 columns over: 1 column being **Date**, and the rest firm tickers. I plan to turn the values under the Date column into DateTime objects, then convert them to Period objects under a separate **Term** field. This way, there will be consistent storage of time periods with *eps_fc* and *eps_act*.

For *eps_fc_terms*, there are 505 firms encompassing 80 quarterly calendar periods since 1999. Overall there are 507 columns: 2 columns being **Forecast Made** and **Term Forecasted**, the rest being firm tickers. This dataset has a couple of discrepancies: one, there are only 80 quarterly calendar periods instead of 84 like the previous datasets, which implies that an entire year is missing. Also, **Forecast Made** refers to the date when the forecast was made for the current fiscal period, and **Term Forecasted** refers to the current fiscal period with the recorded forecast value made 3 months prior.

Though there may be discrepancies of missing data among these DataFrames, the most important thing to note is that the number of firms across all DataFrames is consistent. That

is, data referring to all 505 firms in the S&P 2019 Index have been gathered, and there are *no* missing firms among the collected data.

The next step for examining missing data was to check which year was missing from *eps_fc_terms*. I used an anonymous Python function to isolate the list of years that *eps_fc_terms* contained, and I received an Array of years from 2000 - 2019. As it turned out, the year 1999 is missing from *eps_fc_terms,* which makes sense because the start of the forecasting period would be in the last quarter of 1999, which is October.

Check for Null Values

It is not enough to check for missing values; it is also crucially important to isolate any entries containing null or NaN values, to check for any gaps in the data that may weaken my future analysis.

After storing all DataFrames in a dictionary and iterating through them, it turned out that ***all four DataSets contain null values.*** Below is a summary of the number of rows that contain at least 1 null value:

DataFrame	**# NaN/Null Values**
eps_fc	7055
eps_act	5021
eod_act	6921
eps_fc_terms	7080

In order to combat this problem of null values, my next strategy was to look at both the **number of rows** and **number of columns** with null values, separately. This way, I can isolate which firms and/or time periods contain complete or incomplete data.

After checking rows (time periods) for null data, it turned out that most time periods contained at least one null value—that is, contained incomplete information of at least one firm.

- *eps_fc* has 82 time periods containing missing data, out of 84 total rows.
- *eps_act* has 84 time periods containing missing data, out of 84 total rows.
- *eod_act* has 83 time periods containing missing data, out of 84 total rows.
- *eps_fc_terms* has 80 time periods containing missing data, out of 80 total rows.

After checking columns (firm tickers) for null data, it turned out that there was a higher degree of variance in the number of null data compared to rows. Note that all time period columns contain *no* null data, since I already manually populated those dates before the data gathering stage.

- *eps_fc* has 252 firms containing missing data, out of 505 total firms.
- *eps_act* has 432 firms containing missing data, out of 505 total firms.
- *eod_act* has 505 firms containing missing data, out of 505 total firms.
- *eps_fc_terms* has 340 firms containing missing data, out of 505 total firms.

From the above results, it turned out that the only two datasets with incomplete data for all time periods was **actual EPS** and **forecasted EPS 3 months prior.** To address this problem, I figured it would be helpful to isolate the period ranges for the datasets containing incomplete data by row: **forecasted EPS** and **actual EOD Price.**

Additionally, for all datasets, all firms contain incomplete data across *all* time periods. This was very much expected, as analyzing financial history spanning over 20 years across 505 firms would naturally be rife with missing and inaccurate data. But the advantage is that *eps_fc*, *eps_act*, and *eps_fc_terms* are the most complete DataFrames, while *eod_act* contains the most amount of missing data.

In order to forward with the data assessment stage, I needed to make sure that the above two inconsistencies would not clash with my analysis. My approach to address those inconsistencies was that instead of looking at rows and columns ***with*** missing data, it'd be better to look at rows and columns that are ***all missing data***.

I reasoned that if there was some missing data scattered throughout the matrices of the DataFrames, then that should not skew my analysis so much. However, if there was a significant number of rows/columns that were entirely empty, then I should prepare to ***drop some dates and firms from our overall data.***

Again, I iterated over the DataFrames to look for the *number of entirely empty rows*:

```
#check for empty rows, return False if row contains at least one non-null value, True if all are null
for key, df in dict_dfs.items():
    cols_check = df.columns
    num_empty_rows = (df[cols_check].isnull().apply(lambda x: all(x), axis = 1)).value_counts()
    print(key, '\n', num_empty_rows, '\n----')
```

```
eps_fc
 False    84
dtype: int64
----
eps_actual
 False    84
dtype: int64
----
eod_actual
 False    84
dtype: int64
----
eps_fc_terms
 False    80
dtype: int64
----
```

And as it turned out, *all datasets contain no empty rows.* That is, there is no single period that is completely empty of data. This works greatly to my advantage since I can rely on the firms' values/averages per row instead of having to drop or limit time periods.

Similarly, this code checks for the number of columns with completely empty data:

```
#check for empty columns, return False if column contains at least one non-null value, True if all are null
for key, df in dict_dfs.items():
    cols_check = df.columns
    num_empty_cols = df[cols_check].isnull().apply(lambda x: all(x), axis = 0).value_counts()
    print(key, '\n', num_empty_cols)
```

```
eps_fc
 False    506
dtype: int64
eps_actual
 False    506
True       1
dtype: int64
eod_actual
 False    506
dtype: int64
eps_fc_terms
 False    506
True       1
dtype: int64
```

As it turned out, *eps_act* and *eps_fc_terms* were the only datasets containing one empty column: *one firm with entirely empty data*. I thought this was numerically concerning, so I isolated the singular empty firm for both DataFrames.

```
#helper function to return an array of column names containing empty data
def comb_cols(df):
    empty_cols = []
    for column in df:
        if df[column].isnull().all():
            empty_cols.append(column)

    return empty_cols
```

```
#comb datasets for empty column names
print('In eps_act, the firm {} has no data.'.format(comb_cols(eps_act)))
print('In eps_fc_terms, the firm {} has no data.'.format(comb_cols(eps_fc_terms)))
```

```
In eps_act, the firm ['AMCR UN Equity'] has no data.
In eps_fc_terms, the firm ['AMCR UN Equity'] has no data.
```

The firm AMCR UN Equity is completely empty of data in both *eps_act* and *eps_fc_terms*.

Though this is a quite bothersome discrepancy to deal with, it's still to my advantage that both datasets ***share one firm*** in common regarding missing data. This way, I won't have to worry about dropping two entire firms from the S&P 2019 in my analysis.

Check for Duplicate Data

The next step in my assessment stage was to check for duplicate data: duplicate entries, duplicate time periods, and duplicate firms. I executed the below code to check for duplicate data across all four DataFrames:

```
for key, df in dict_dfs.items():
    print(key, df.duplicated().sum())
```

```
eps_fc 0
eps_actual 0
eod_actual 0
eps_fc_terms 0
```

And as it turns out, there was ***no duplicate data.*** This is extremely good news, but I had to be more thorough. My next step was to check for duplicated firm names. I reasoned that although the presence of duplicated firm names inherently implies the presence of duplicate

data, sometimes data can act irrationally, choosing to disperse in unexpected ways, especially when dealing with large datasets.

```
#check for duplicated firm names
for key, df in dict_dfs.items():
    print(key, df.columns.duplicated().sum())
```

```
eps_fc 0
eps_actual 0
eod_actual 0
eps_fc_terms 0
```

For all datasets, there existed ***no duplicate firm names.*** This is also extremely good news; now I knew that there would be no need to dedupe my data during the cleaning stage.

Check Data Types

The final step in my assessment stage was to check for data types: that is, make sure all numerical data types are consistent. For example, EPS and EOD values must all be *float* values, while Dates should all be *objects* (for now). Here, *float* values refer to a variable type that utilizes fractional values, instead of digits.

- *eps_fc* contains ***1 date field***, so I should expect ***1 object type.***
- *eps_act* contains ***2 date fields,*** so I should expect ***2 object types.***
- *eod_act* contains ***1 date field,*** so I should expect ***1 object type.***
- *eps_fc_terms* contains ***2 date fields,*** so I should expect ***2 object types.***

And other fields should be **float** types.

After examining the number of float and object types across all four DataFrames, it turned out that only *eps_fc*, *eod_act*, and *eps_fc_terms* were all consistent with the expected

number of object types. On the other hand, ***eps_act contained 7 object columns***—5 more object columns than expected. This was odd; there were 5 firms that did not contain numerical data? Earlier, I already established that there was only 1 firm with missing data. To examine further, decided to isolate all columns of 'Object' type under *eps_act:*

Isolate 'Object' Columns under `eps_act`

```
eps_act.select_dtypes(include = 'object').head()
```

	Quarter	BRK/B UN Equity	FOX UW Equity	GOOG UW Equity	HCP UN Equity	SYMC UW Equity	UA UN Equity
0	Q1	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable
1	Q2	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable
2	Q3	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable
3	Q4	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable
4	Q1	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable	#N/A Field Not Applicable

The above firms are all 'Object' types because there is no recorded data under them; there had been no numerical values assigned to these firms during the gathering stage, which meant that Pandas mistook the uniform presence of these "#N/A Field Not Applicable Errors" as objects. **This is all actually missing data. From *eps_act*, there are 6 firms that are empty of data.**

This was a tricky situation to spot, of missing data disguised as "Object" types. There was actually more missing data than anticipated.

Moving forward, I summarized all the issues around the data that I needed to immediately correct. I separated all issues into two separate categories: ***Quality*** and ***Tidiness.***

C) Summary of Data Issues

Missing Data

Despite the discoveries around the vast amounts of missing data, in the end, I decided ***not*** to drop any rows or columns with missing data. For the sake of data preservation, I kept all missing data recorded as NaN, or null, values. Below is a summary of the ***3 discoveries made around missing data:***

- *eps_fc_terms* is missing the year 1999
- *eps_act* and *eps_fc_act* have one firm with empty data: 'AMCR UN Equity'
- *eps_act* contains 7 empty firms: BRK/B UN Equity, FOX UW Equity, GOOG UW Equity, HCP UN Equity, SYMC UW Equity, and UA UN Equity

Data Quality

According to Rouse, Margaret, data quality is a "measure of the condition of data based on factors such as accuracy, completeness, consistency, reliability, and whether it's up to

date."[27] In other words, data quality issues pertain to the degree of *integrity* and *reliability* of the data. Data, though organized, may be outdated, unorganized, and/or incomplete.

From the previous data assessment stage, I have isolated ***11 data quality issues:***

1. Unnormalized date formats among all DataFrames
2. Firm names across all DataFrames are capitalized and contain white space
3. Erroneous data type for *eps_act* Object columns
4. Erroneous data type for **date** under *eod_act*
5. Erroneous data type for **forecast_made** under *eps_fc_terms*
6. Erroneous data types for **term** under *eps_act*, *eps_fc*, and *eps_fc_terms* to DateTime quarter index
7. No recorded quarterly data for *eod_act*
8. Firm names are not referenced by **firm_id**
9. No recorded 20-year average for each dataset
10. No recorded yearly averages for each dataset
11. No recorded quarterly averages for each dataset

Feature Engineering

[27] Rouse, Margaret, et al. "data quality." TechTarget, November 19, 2019.

Data quality issues 8 to 11 have revealed two approaches I could take: I could *feature-engineer* a column containing the twenty-year, yearly, and quarterly averages of EOD and EPS for each firm.

Data Tidiness

Data tidiness causally relates to the *organization* of the data. The principal question for checking tidiness is: *does each entry in the dataset represent values consistent with their column headers?* That is, are there any unrelated values under any column header? For example, a **Letter Grade** column containing a percentage (93.7%) instead of A, B, C, D, or F.

I have located ***3 data tidiness issues***:

1. Unnormalized data among *eps_act, eps_fc, eod_act,* and *eps_fc_terms*
2. Unnormalized data among *df_twenty_year_avgs, df_yearly_avgs,* and *df_quarter_avgs*
3. 20-year, yearly, and quarterly averages contained in different DataFrames

Tidiness issues 2 and 3 refer to three new DataFrames that I generated during the cleaning stage: *df_twenty_year_avgs, df_yearly_avgs,* and *df_quarter_avgs*. Once I created these DataFrames later in the cleaning stage, I added these issues back to this list of tidiness problems for the new DataFrames.

Chapter 3: Data Cleaning

A) Preparation

I created separate copies of each DataFrame: *eps_fc_clean, eps_act_clean, eod_act_clean,* and *eps_fc_terms_clean,* all of which are duplicates of their corresponding DataFrame. It is a wise decision to make clean copies of all original datasets before performing heavy operations on them.

Additionally, I separate my code into a **Code** and **Test** section. The former contains the actual manipulation code, whereas the latter contains code verifying that my changes took effect.

After isolating all the above issues around the data, I proposed a solution for each one accordingly, divided by quality and tidiness. I reorder each issue numerically, and then define the actions I needed to take to resolve them individually.

B) Quality

The code documenting each solution to each data quality and tidiness issue can be found under the *data_cleaning.ipynb* file.

Issue 1: Unnormalized date formats among all DataFrames.

Solution:

- Conjoin *eps_act_clean* dates from 2 columns into 1 to match *eps_fc_clean* format. The format we want is YYYYQQ. For example, Quarter 1 in 2005 will be represented as 2005Q1.
- Rename both fiscal periods under *eps_act_clean* and *eps_fc_clean* to **"term".**
- Under *eps_fc_terms_clean*, format **term** to YYYYQQ.

Issue 2: Firm names across all DataFrames are capitalized and contain white space.

Solution:

- Iterate across all clean DataFrames. Get rid of all characters from the first whitespace character onward. Lowercase column names.

Issue 3: Erroneous data type for *eps_act* Object columns

Solution:

- Convert "#N/A Field Not Applicable" strings into NaN data type
- Convert the 6 *eps_act_clean* Object columns into NaN type.

Issue 4: Erroneous data type for **date** under *eod_act*

Solution:

- Convert column **date** to DateTime object.

Issue 5: Erroneous data type for column **forecast_made** under *eps_fc_terms.*

Solution:

- Convert column **forecast_made** to DateTime object.

Issue 6: Erroneous data types for **term** under *eps_act*, *eps_fc*, and *eps_fc_terms* to DateTime quarter index

Solution:

- Convert YYYYQQ formats into DateTime quarter index type

Issue 7: No recorded quarterly data for *eod_act*

Solution:

- Add a new column **term**, which extracts the calendar *year* and calendar *quarter* from the **date** column

It is important to note that all dates have been normalized to YYYY-MM-DD format, and all quarterly records to YYY-QQ format. This is to enable for more efficient handling, cleaning, and classification of data later on.

Issue 8: Firm names not referenced by **firm_id**.

Solution:

- Assign a **firm_id** to each firm for future normalization.
- Generate a new CSV named *firms.csv.*

Issue 9: No recorded twenty-year averages for each dataset.

Solution:

- Isolate 20-year averages for each firm into its own DataFrame.
- Create new DataFrame *twenty_avgs* depicting all 20-year averages for each firm

Issue 10: No recorded yearly averages for each dataset.

Solution:

- Create 4 separate DataFrames for all attributes

- Rename columns to “Feature_Year” (e.g. *eps_fc_1999, eod_act_2000,* etc.)
- Outer merge all DataFrames to create a new DataFrame: *df_yearly_avgs,* on **firms**

Issue 11: No recorded quarterly averages for each dataset.

Solution:

- Parse *eod_act_clean* dates by **calendar quarter average** with quarterly data in a new temporary DataFrame.
- Create separate DataFrames containing quarterly averages for *eps_fc_clean, eps_act_clean,* and *eps_fc_terms_clean*
- Rename columns to “Quarter_Year_Feature” (e.g. *q1_eps_fc,* etc.)
- Outer merge all DataFrames into a new df *df_quarterly_avgs*

C) Tidiness

Issue 1: Unnormalized data among *df_twenty_year_avgs, df_yearly_avgs,* and *df_quarter_avgs*

Solution:

- Normalize features across all 3 DataFrames.
- For *df_twenty_year_avgs,* melt **feature column names** (eps_fc, eps_act, eod_act, and eps_fc_terms) into a single column called **feature**

Here is the example desired output combining each DataFrame:

firm_id	average	average_type	time_period	feature
501	0.21875	yearly	2002	eps_fc_terms
378	0.059000	quarterly	q4	eps_act
147	NaN	yearly	2016	eod_act
33	0.005067	quarterly	q4	eps_fc

*Projected table structure for **avgs.csv***

My goal is to combine all three twenty-year, quarterly, and yearly DataFrames to achieve the above output. I will concatenate all 3 DataFrames to create a new DataFrame called *all_avgs.* Basically, I want to **condense twenty-year, quarterly, and yearly averages into a single dataset.**

Note: for *df_twenty_year_avgs*, it would be redundant to assign *twenty_year* as the value under both the **average_type** and **time_period** columns. To curb this problem, I will set all **time_period** values as NaN under *df_twenty_year_avgs.*

Issue 2: Twenty-year, yearly, and quarterly averages contained in different DataFrames.

Solution:

- Concatenate the normalized *twenty_year_clean*, *yearly_clean*, and *quarterly_clean* DataFrames into a new ***avgs.csv*** file.

Issue 3: Unnormalized data among *eps_act*, *eps_fc*, *eod_act*, and *eps_fc_terms.*

Solution:

- Achieve desired *features.csv* output below:

firm_id	feature	date	term	value
485	eps_fc	NaT	2004A1	1.40900
104	eps_act	NaT	2001Q4	-0.113333
337	eod_act	2019-03-29	2019Q1	87.490
276	eps_fc_terms	2009-04-01	2009Q3	0.009

*Projected table structure for **features.csv***

I intend to use **features.csv** as the main CSV containing the "raw data" of all our firms during the data exploration stage. As deduced from above, the **date** column contains the **date column** in *eod_act* and **forecast_made** in *eps_fc_terms.* They are assigned to the same column because both columns are consistent: they both contain the same DateTime object, represented in the same form. However, the **date** feature will be NaN for *eps_fc* and *eps_act* features because they were recorded by YYYYQQ, not calendar dates.

For both *eps_fc* and *eps_act* feature types under this new anticipated **features.csv** file, we don't have to worry about handling the **date** column. This will be resolved later when merging all DataFrames with *eod_act* and *eps_fc_terms.*

D) Storing the Data

I successfully created ***avgs.csv*** and ***features.csv*** that directly mirrors the 2 table outputs depicted above. Now that I have created my new, clean DataFrames that have resolved all data quality and tidiness issues, it is time to convert them into CSVs.

I decided to put the following CSVs under the ***./data/clean/averages/components/*** directory because they form *part* of the main CSVs under the ***./data/clean/averages/*** directory:

- *quarter-avgs-eod-act.csv*
- *quarter-avgs-eps-act.csv*
- *quarter-avgs-eps-fc.csv*
- *quarter-avgs-eps-fc-terms.csv*

All the above are components of the ***quarter-avgs.csv*** file under ***/averages/***

- *yearly-avgs-eod-act.csv*
- *yearly-avgs-eps-act.csv*
- *yearly-avgs-eps-fc.csv*
- *yearly-avgs-eps-fc-terms.csv*

All the above are components of the ***yearly-avgs.csv*** file under ***/averages/***

Additionally, I created ***firms.csv*** to display a correspondence between each *firm ticker* and their primary *firm_id* key.

Note: For the twenty-year, yearly, and quarterly datasets, I stored their old *df_* versions instead of their new clean versions. This is because their old versions are stored more efficiently as ***standalone*** CSVs and were only modified to fit cohesively into the larger *all_avgs* DataFrame.

Before finally exploring my clean data, I decided it was crucial to keep these few details in mind:

- When dealing with CSVs depicting averages, we need to keep in mind **missing data** in our interpretations.
- I decided to put the years as columns under the yearly average DataFrame generation because the amount of columns generated here would be less than the amount of firms there actually is: 505.
- Keep in mind that *eod_act* and *eps_fc_terms* are based on **calendar years**, not fiscal years.

After finalizing the data cleaning stage, I converted the entire IPython Notebook into HTML, to allow for easier access and web browser display:

```
#convert notebook to HTML
from subprocess import call
call(['python', '-m', 'nbconvert', 'data_cleaning.ipynb'])
```

Chapter 4: Data Exploration

A) Summary of Clean Data

After cleaning the original datasets as documented in my previous data wrangling process, I generated these two clean CSVs, along with a breakdown of their features and columns.

For the sake of reference, here is a breakdown of what each abbreviation for each stock price type denotes:

Forecasted EPS	*eps_fc*
Actual EPS	*eps_act*
EOD Price	*eod_act*
Forecasted EPS (3 Months Prior)	*eps_fc_terms*

Code abbreviations for each stock price type

Below is a summary of each clean CSV's column names:

features.csv

- **firm_id**: foreign key referring to primary keys in **firms.csv**
- **feature:** type of feature that the **value** field denotes *eps_fc*, *eps_act*, *eod_act*, or *eps_fc_terms*
- **date:** DateTime object in YYYY-MM-DD format. This field has different meanings depending on the previous *feature* field.
    - For ***eod_act***, this is the date at which the EOD Price was recorded

    - For ***eps_fc_terms***, this is the date at which the EPS term forecast was made

- **term:** Pandas Period object in YYYYQQ format (the time period when the value was recorded)
    - ***Fiscal*** terms for *eps_fc* and *eps_act*
    - ***Calendar*** terms for *eod_act* and *eps_fc_terms*
- **value:** the EPS or EOD value as specified in the ***feature*** *column*

avgs.csv

- **firm_id**: foreign key referring to all primary keys in **firms.csv**
- **average:** recorded average value as specified in the ***average_type*** column
- **average_type**: type of average that the **average** field denotes twenty-year, quarterly, or yearly
- **time_period:** the time period in correspondence to the previous ***average_type***
    - For twenty-year averages, default time_period is NaN
    - For yearly averages, it displays the year (YYYY)
    - For quarterly averages, it displays the quarter (QQ)
- **Feature:** type of feature that the **average** field denotes *eps_fc, eps_act, eod_act*, or *eps_fc_terms*

firms.csv

- ***firm_id***: a primary key assigned to the firm ticker under the ***firm*** column
- ***firm:*** firm ticker of a company in the 2019 S&P Index

Our **features.csv** dataset contains 167,660 entries with 5 column names. ***firm_id, feature, date,*** and ***term*** are all categorical variables while the ***value*** field is a numeric and continuous variable. Even though the ***date*** and ***term*** fields are recorded as DateTime and Period objects respectively, I still classify them as discrete, categorical data. This is because there is a limit to the year that can be recorded (1999 - 2020) and there cannot be more than 4 quarters (Q1 - Q5). Therefore, while categorical, these variables are still discrete; they are countable.

Our **avgs.csv** dataset contains 52,015 entries with 5 columns. ***firm_id, average_type, time_period,*** and ***feature*** are all categorical variables while the ***average*** column is numeric and continuous.

Therefore, there is only *one* numerical variable among all of our data.

Features of Interest

In relation to the main research focus, my principal goal is to analyze the historical correlation of forecasted vs. actual EPS across all firms in the 2019 S&P Index.

For the above reason, *eps_fc* and *eps_act* are the main variables of interest. They are both consistent in measuring both the *forecasted* and *actual* EPS for all 505 firms, per fiscal period, over a span of 20 years.

The variable *eod_act* is recorded based on calendar period instead of fiscal period, which is a flaw in the quality of the data. However, this variable could be used for drawing further conclusions on stock market trends.

The variable *eps_fc_terms* is also based on calendar period instead of fiscal period, which is yet another flaw in the quality. Although *eps_fc_terms* depicts forecasted EPS made three months prior to the current fiscal term, this is still not a main variable of interest. *eps_fc_terms* simply acts as a way to *extend* my main research question, but not fully answer it.

B) Research Questions

These are the questions I pose in my exploratory stage:

Question 1: Does average EPS prediction error depict any differences in trends whether by a yearly, quarterly, or full-term basis?

Question 2: I generate naive EPS forecasts by calculating the rolling mean of the 2 actual EPS values from the past 2 quarters. How do my EPS forecasts compare to Bloomberg's EPS forecasts?

Question 3: What differences and similarities emerge when analyzing the prediction error and percentage error of EPS forecasts?

Question 4: How do my naive RMSE values compare to Bloomberg RMSE when accounting for EPS forecasts? For what percentage of firms does my

naive RMSE beat Bloomberg's RMSE?

Question 5: How do EOD Prices trend across all firms from 1999 - 2019?

For the broadest overview, I predict that overtime, EPS forecasts will continually become optimistic for those firms that consistently have high actual EPS values. Vice versa: overtime, EPS forecasts will continually become pessimistic for those firms that consistently have lower-than-expected EPS values.

As for the other features in the dataset that will support my investigation, I expect yearly values to show more consistency in pattern than quarterly values. This is because it is more intuitive and accurate to depict a time series by year than quarter. Economic situations greatly diversify over the period of 20 years, no matter the period, and years paint a better picture of time-based trends. Meanwhile, quarterly data depicted over a long period of time would, at best, show only the quarterly averages among those years.

I separated the data exploration stage into three sections: univariate exploration, bivariate exploration, and multivariate exploration. The univariate approach seeks to find relationships among singular variables (e.g. distribution), bivariate between two variables (e.g. scatterplots), and multivariate among three or more variables (e.g. plot matrices).

The CSVs **features.csv** and **avgs.csv** are the main datasets of focus.

All univariate, bivariate, and multivariate visuals can be found as PNG files under the directory ***./data/visuals/***.

C) Univariate Exploration

To get a better overview of the data before going on to generate visualizations and interpretations, I decided to focus on missing values first.

Missing Values (features.csv)

Here are the distributions of missing values between both CSVs, starting with **features.csv:**

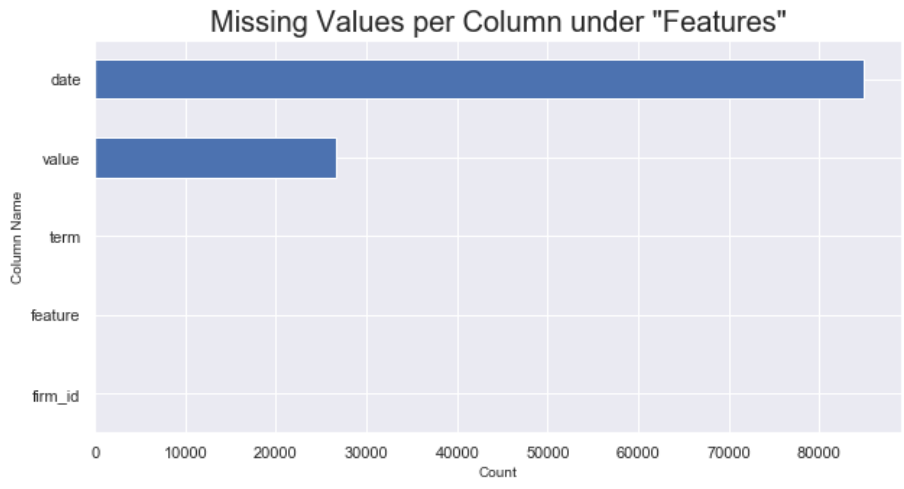

Figure 1

From this visual, I gathered that **date** is the column with the highest amount of missing data, with around 85,000 missing entries. I also observed that the **value** column contains around 26,000 missing entries. Also, the gap in number of missing values between **value** and **date** is noticeably large: this makes sense because **date** is automatically set to 'NaT' for *eps_fc* and *eps_act*. 'NaT', which stands for "Not a Time," which is recognized by Pandas as a null value.

If we dropped all the entries where **date** is not NaT, then we'd isolate only *eod_act* and *eps_fc_terms* data. If we dropped all NaT entries, then our data analysis would be more approach. Therefore, I proposed this question:

Question 1: for all entries where **date** is not NaT, how large is the gap in missing values between *values* without the dropped dates and *values* with the dropped dates?

After dropping all NaT **date** entries under ***features.csv***, I generated this graph:

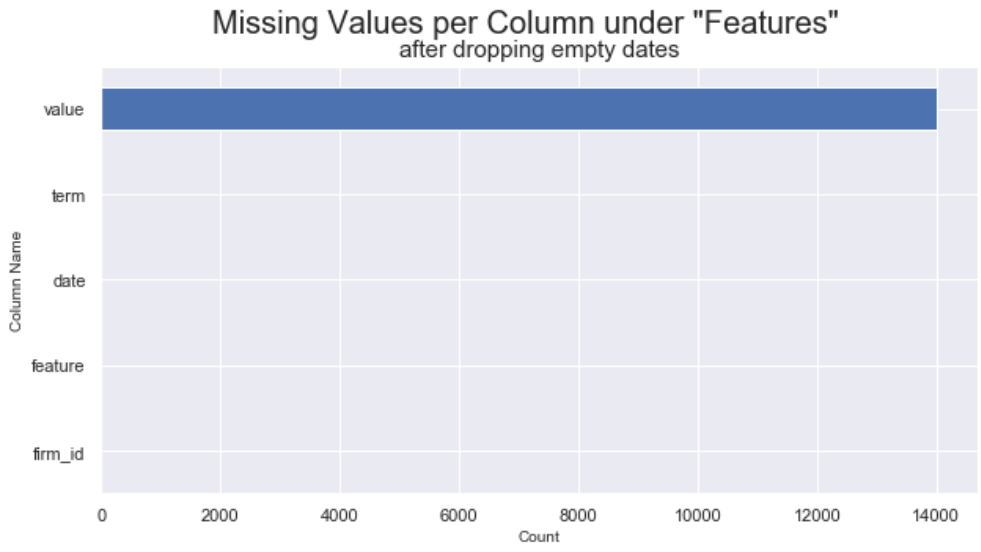

Figure 2

Before dropping NaT dates, I noticed that the amount of overall missing values is around **26,000.** After dropping all the NaT dates (dropping *eod_act* and *eps_fc_terms*), the amount of missing values dropped to around **14,000.**

Drawing from the above observation, this means that *eps_fc* and *eps_act* contain around **12,000** missing values in total. Effectively, the undropped columns, *eod_act* and *eps_fc_terms* would have around **14,000** missing values in total. Now that I isolated the number of missing values of both groups, now I wanted to examine them individually.

Question 2: How many missing values does each feature contain, individually?

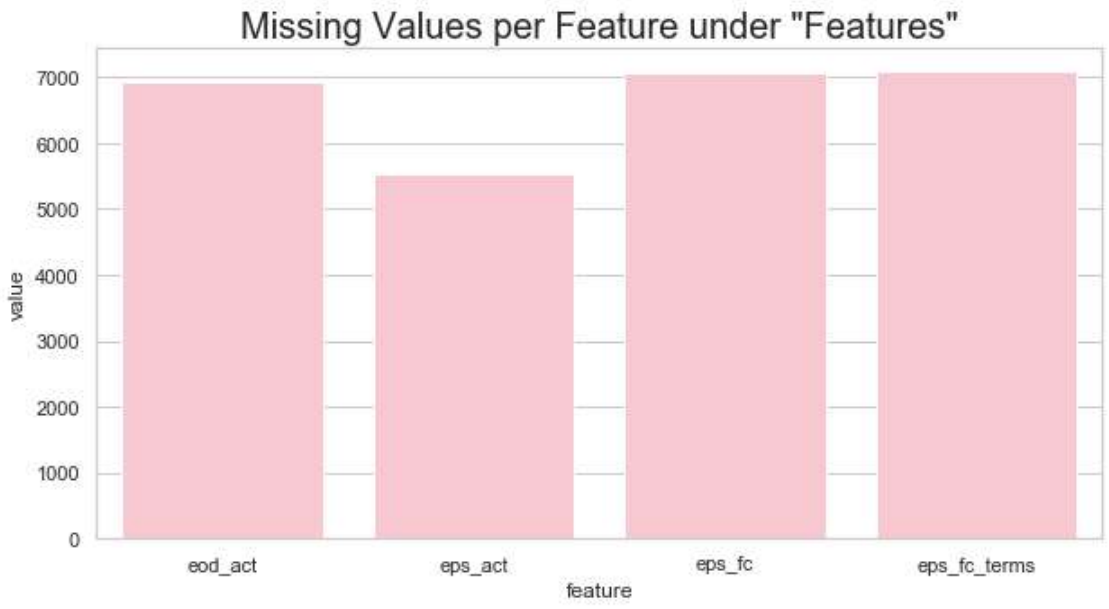

Figure 3

The feature with the greatest amount of missing values is *eps_fc_terms,* whereas the feature with the least amount of missing values is *eps_act.* Additionally, *eod_act*, *eps_fc,* and *eps_fc_terms* all contain a very small gap in missing data relative to each other, but much larger than *eps_act*.

To answer **Question 2**, below is a list of each feature and their corresponding estimated number of missing values:

- *eod_act*: 7,900
- *eps_act:* 5,500
- *eps_fc*: 7,050
- *eps_fc_terms:* 7,100

Additionally, *eps_fc* and *eps_act* contain around 12,000 missing values in total, which is consistent with the previous observation.

The features *eod_act* and *eps_fc_terms* have around 14,000 missing values in total, which is also consistent with the previous observation.

Missing Values (avgs.csv)

By using Matplotlib, I generated a bar plot depicting the count of missing values per column under **avgs.csv**.

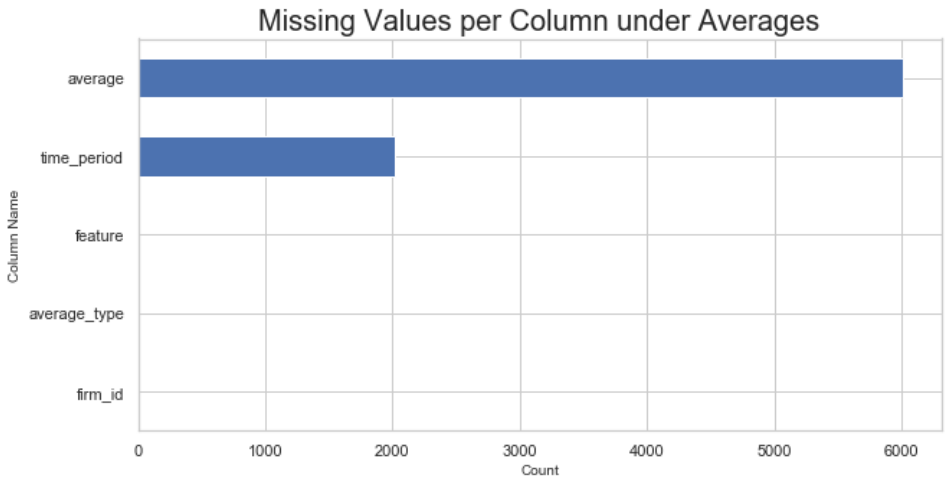

Figure 4

From the above visualization, I observed that the **average** column contains exactly 6,000 missing data entries. The **time_period** column contains exactly 2,000 missing data entries. The gap in missing values between **average** and **time_period** is 4,000. However, I also noted that by default, all entries with **average_type** of *twenty_year* contain NaT fields for **time_period.**

To address the above problem, I pose these questions:

Question 3: After dropping all entries with an NaT field under **time_period**, how large will the gap in missing values be between *values with the NaT dates* and *values after dropping the NaT dates*?

Question 4: How many missing **average** values does each **average_type** contain?

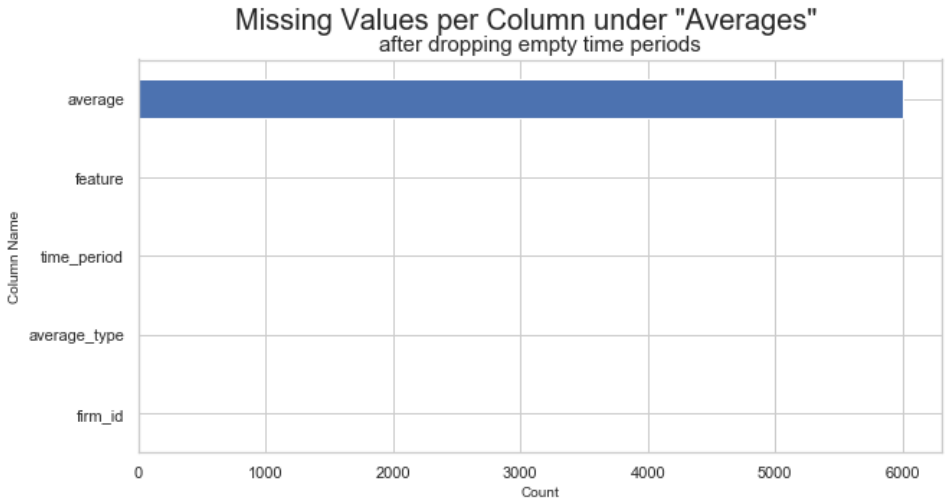

Figure 5

To answer **Question 3**, I observed that after dropping all entries containing NaT dates under **time_period**, there are still 6,000 missing averages. This number is consistent with

the number of missing averages even before dropping the NaT dates. Therefore, **all entries with a missing period** (all *twenty_year* average types) **did not contain any empty data.**

In summary, the number of missing **average** values remains the same, whether I drop all entries with NaT entries, aka all *twenty_year* average types.

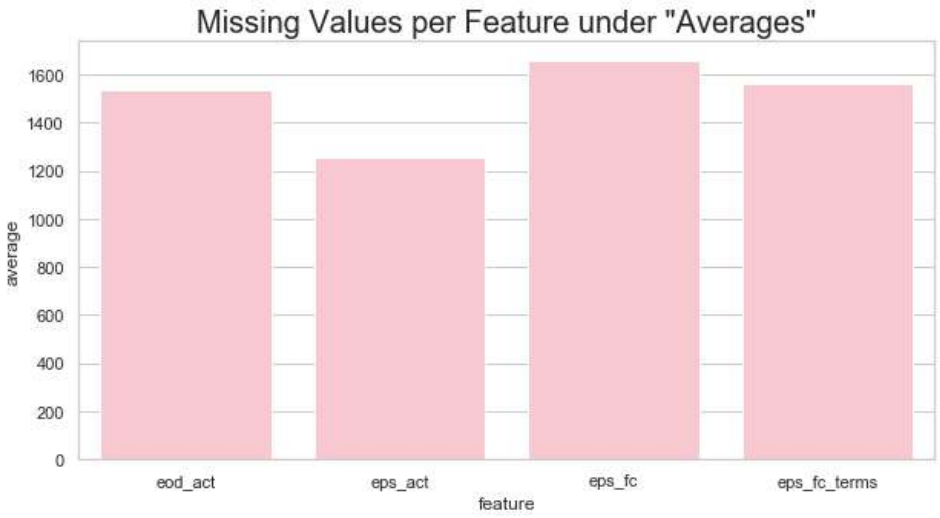

Figure 6

The feature *eps_fc* has the most missing values at around 1,650. Also, *eps_act* contains the least amount of missing values at around 1,250.

Under **avgs.csv**, the gap in missing values between each average type is ***larger*** than the missing values under **features.csv**. Additionally, each feature has the following approximate number of missing averages:

- *eod_act* : 1,550
- *eps_act* : 1,250
- *eps_fc* : 1,650

- *eps_fc_terms*: 1,590

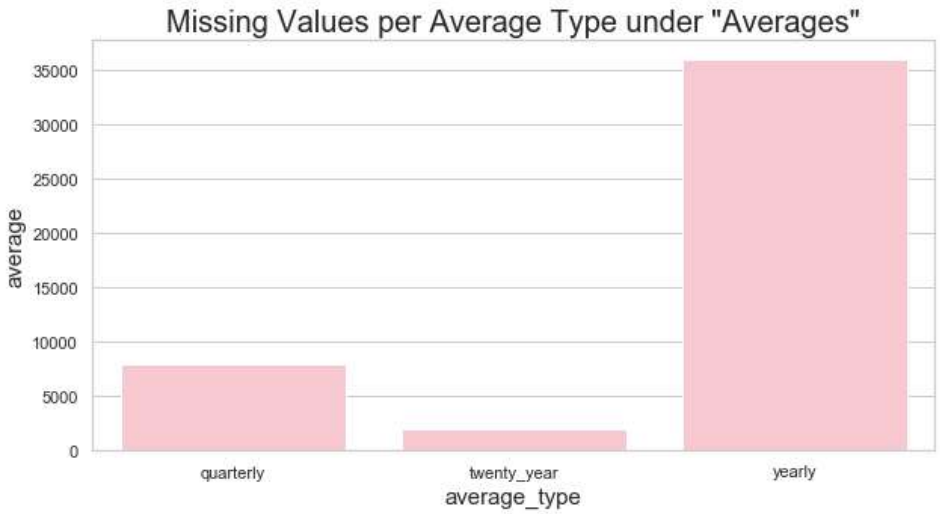

Figure 7

To answer **Question 4**, here is a breakdown of missing values per average type:

- *quarterly*: 7,500
- *twenty_year:* 2,500
- *yearly:* 3,600

Additionally, the gap in missing values is greatly diversified among all average types.

***firm_id* (features.csv)**

Since 505 firms would cause a single visual to be crowded, I decided to narrow my focus on the ***top 20 most common firm ids*** and the ***20 rarest firm ids.***

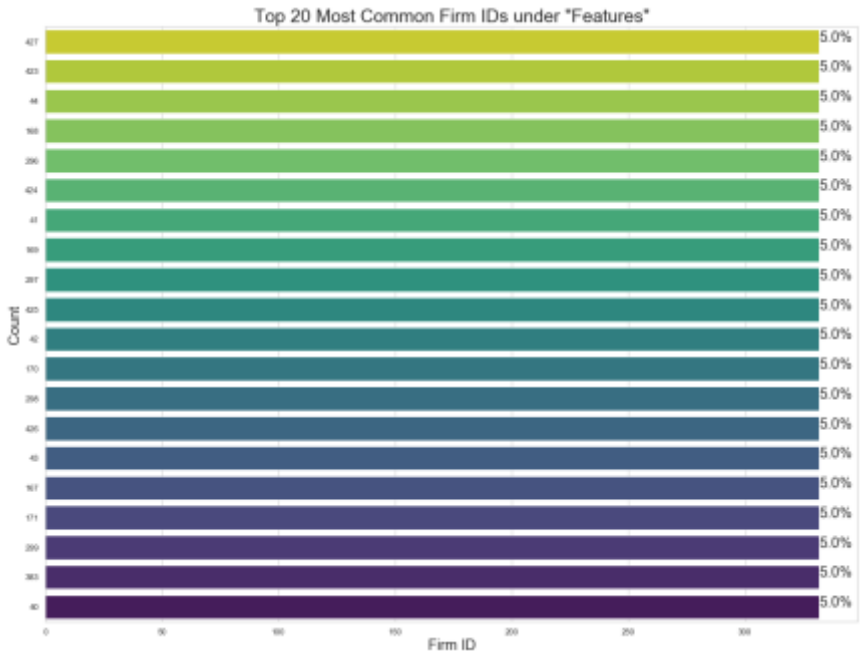

Figure 8

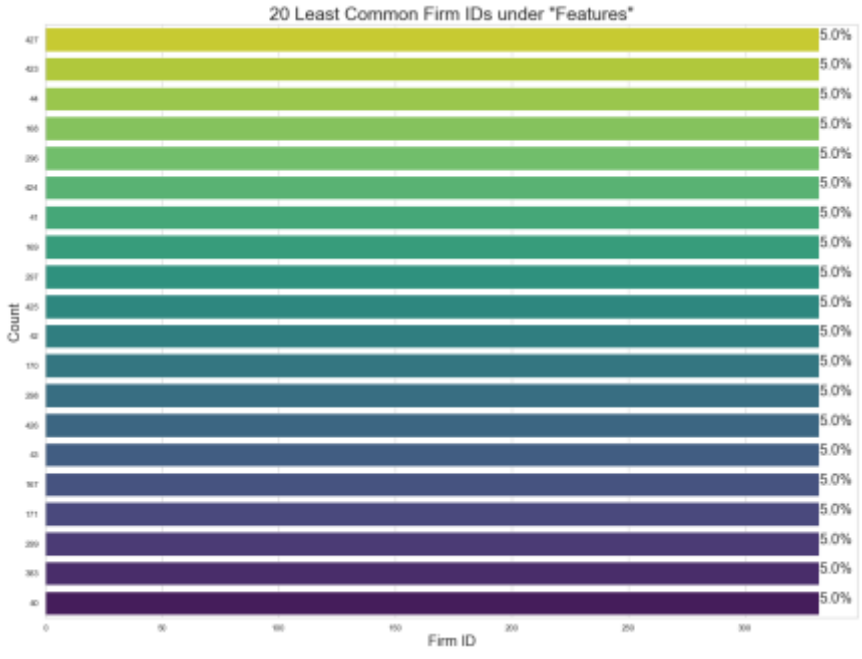

Figure 9

A curious detail that I derived from the above two bar graphs is that ***both*** the 20 most common and least common firm IDs all individually make up the same proportion of existing Firm IDs: exactly **5% each.** This means that under **features.csv**, there is a ***consistent count among all Firm IDs*** at around 335. But just to double-check, I inserted this piece of code to check the exact number of firm IDs under this DataFrame:

```
#check count consistency among firm_ids
firm_counts = features.firm_id.value_counts()
np.unique(firm_counts.sort_values().values)
```

Thus, I discovered that ***all firm ID counts are consistent across the entire*** **features.csv** ***dataset,*** at 332 entries per firm ID. Therefore, there are ***no null firm ids.***

feature **(features.csv)**

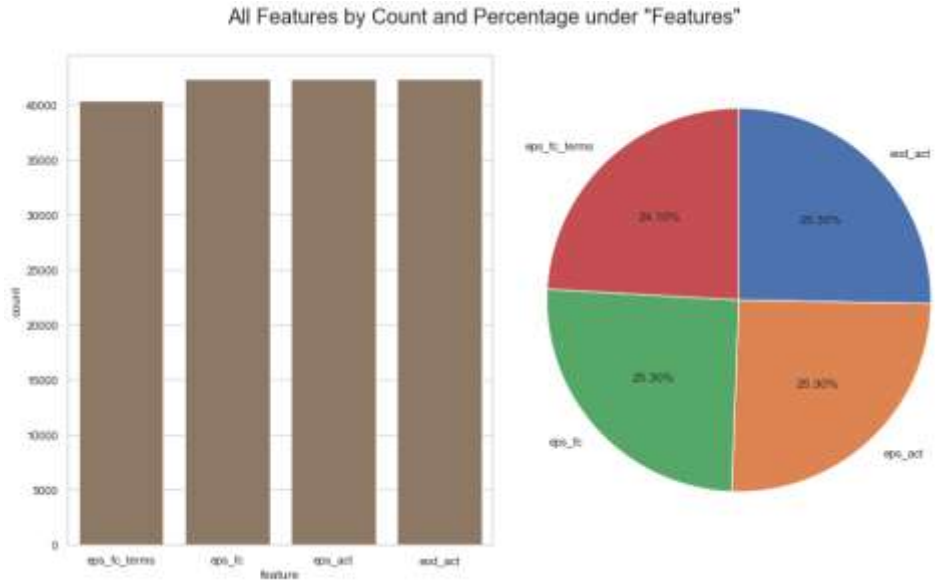

Figure 10

From the above histogram and pie chart, *eps_act*, *eps_fc*, and *eod_act* all show consistent counts at around **42,500 entries** (25.30%) each. Also, *eps_fc_terms* is the only **feature** type to deviate from the others, having less entries at around **41,000** (24.10%). It makes complete sense that *eps_fc_terms* contains more missing data, because the year 1999 was not included while gathering this data. Effectively, that should remove 2,020 entries, and this can be seen in the gap between *eps_fc_terms* and any of the other bars.

date (features.csv)

Since we acknowledged that the **date** column is set to NaT for *eps_fc* and *eps_act*, we must write all our interpretations accordingly. Thus, all graphs under **date** do not take into account the features *eps_fc* and *eps_act*.

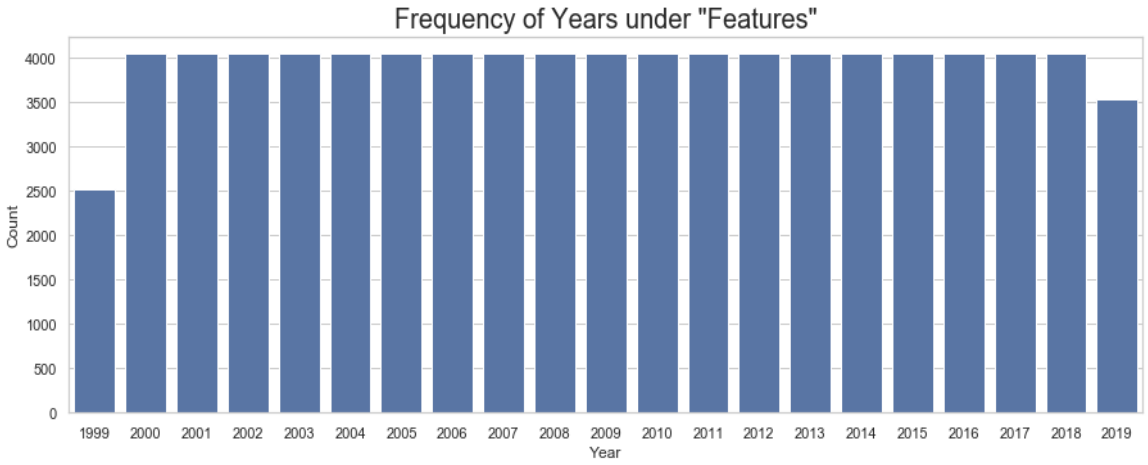

Figure 11

All years between 2000 and 2018 show consistent counts at around 4,000. This makes sense, because the year 1999 is missing from all *eps_fc_terms* entries. Also, the years 1999 and 2019 are both *inconsistent* and *less than* the number of 4,000 counts for all other years. The year 1999 has 2,500 non-null entries, while 2019 has 3,500 non-null entries.

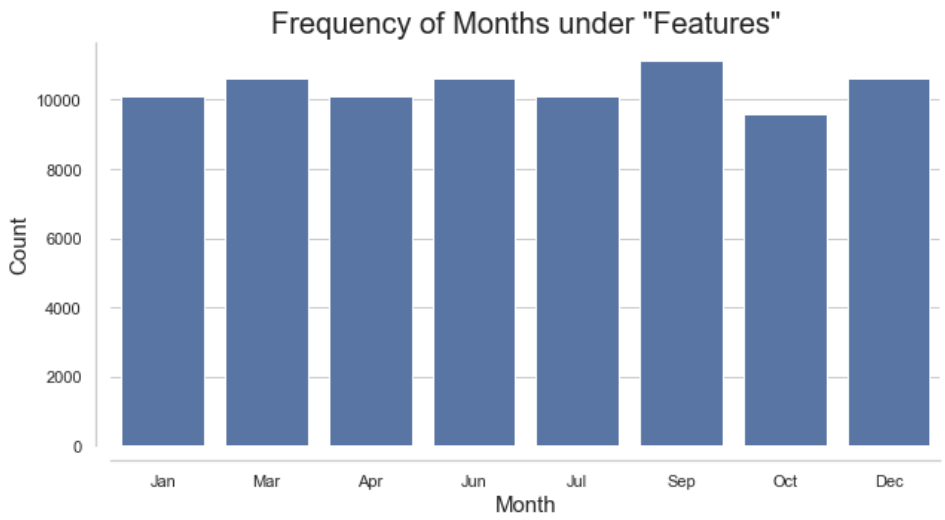

Figure 12

September is the month with the most number of non-null entries, at around **15,000.** In contrast, October has the least number of non-null entries at around **9,000.** The trend in counts of months under the **Date** column fluctuates; the trend by month does not display a linear or exponential pattern.

Most importantly, it appears that there is a "peak" in counts every 2nd *recorded* month from January. For example, March peaks at around 11,000 counts, then June peaks at around 10,500 counts, and September at around 15,000—all months within a consistent range from each other on the x-axis.

After examining the **date** column, I decided it safe to conclude that the **date** column is unreliable when examining ***terms*** and ***years*—**conducting any analysis with this column should be avoided. It is better to use the **date** column *only* when referring to specific dates instead of general terms.

term (features.csv)

As noted earlier, there are no missing values under the **term** column. Therefore, all graphs below do account for *eps_act* and *eps_fc*.

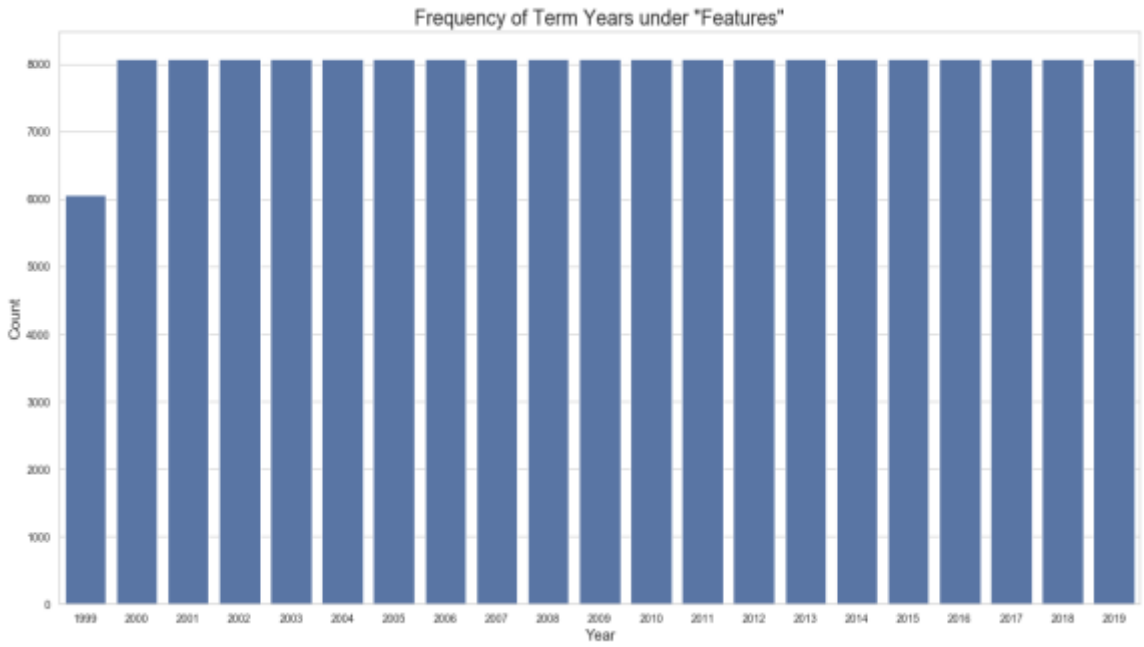

Figure 13

All years from 2000 to 2019 have consistent counts at **8,000** per year. And unlike the years under the **date** column, 1999 is the only year that does not follow the general trend of the bars from 2000 to 2019. There are 6,000 recorded entries containing 1999, which means that *2,000* entries do not contain the year 1999.

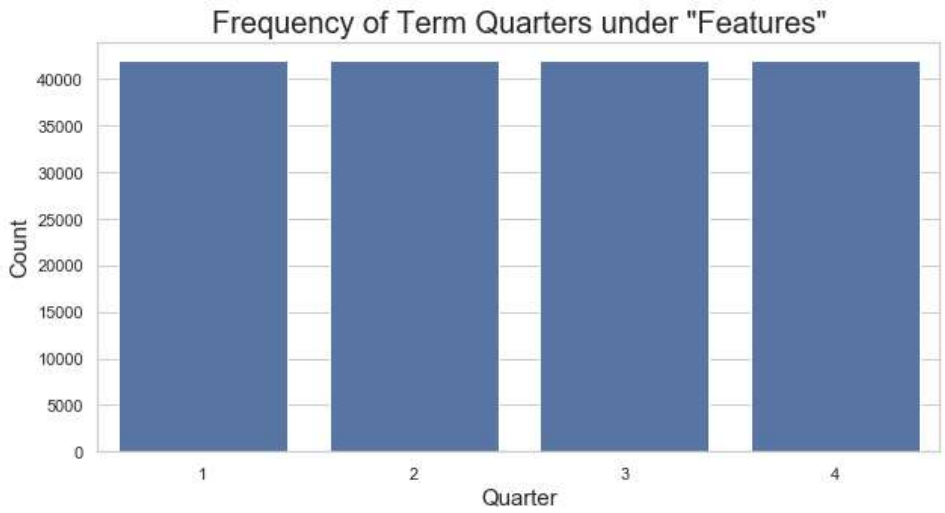

Figure 14

Unlike the previous graph depicting years, there is a consistent number of quarterly entries under **term**. This means that *quarterly* data is a more stable, reliable variable to examine under the **term** column unlike *yearly* data, which should be examined more closely regarding the feature(s) being examined in that year.

value (features.csv)

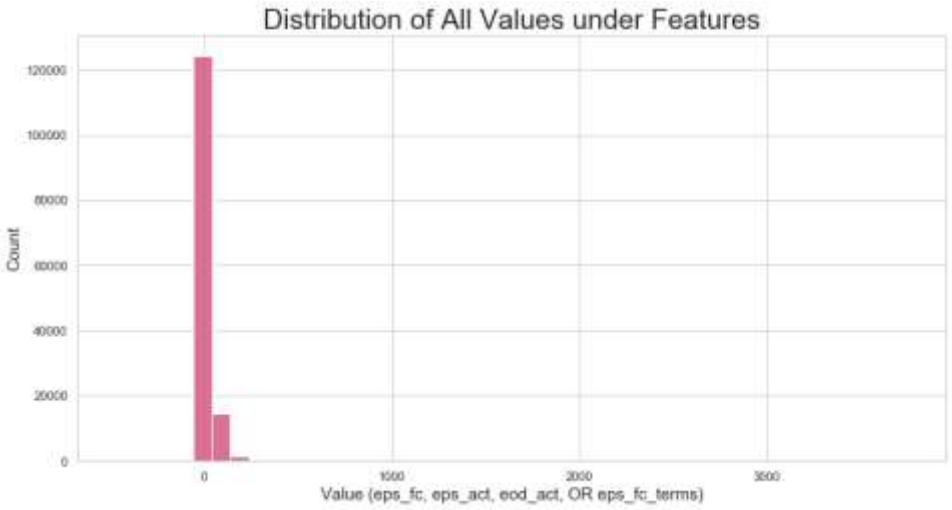

Figure 15

The trend in value counts is ***heavily right-skewed.*** On the x-axis, the value range 0-100 contains the highest concentration of data, with over 120,000 entries. Also, the value range 0-300 contains the "bulk" of the entire dataset, which means that the bars in all surrounding x-axis values are all outliers.

Question 5: How do value counts under **features.csv** look like after removing all outliers around the value range 0-300?

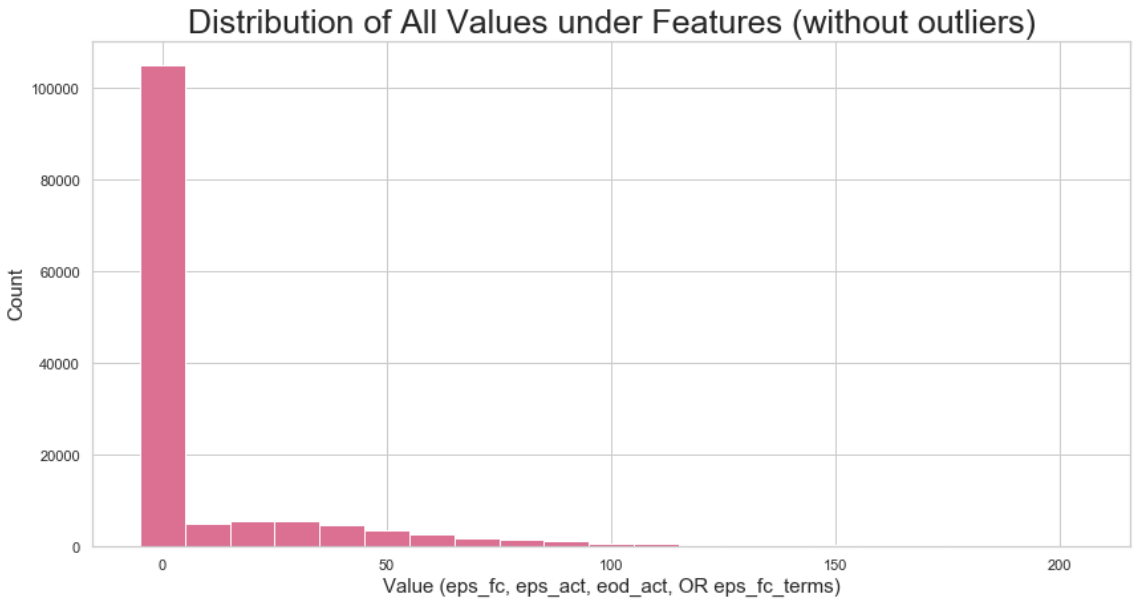

Figure 16

To answer **Question 5**, the graph still appears to be a normal distribution with a ***strong right skew.*** This is unlike the previous graph *with* outliers, where the right skew is much more pronounced and prominent.

The value range 0-10 contains the "bulk" of all the data; this x-axis range contains the ***highest bar***, with counts of around 120,000. The previous graph showed that the bin range 0-100 contains around 120,000 points, and coincidentally, this graph *also* shows that the

bin range 0-10 contains around 120,000 points. Expanding from these observations, I can safely conclude that ***it is exactly the bin range 0-10 that contains the bulk of all the data.***

Also, instead of separating the histogram bins by bin widths of 100, the bin width here is 10.

Just to be more thorough in our breakdown of missing values under this column, I broke down the 0-10 bin range even further:

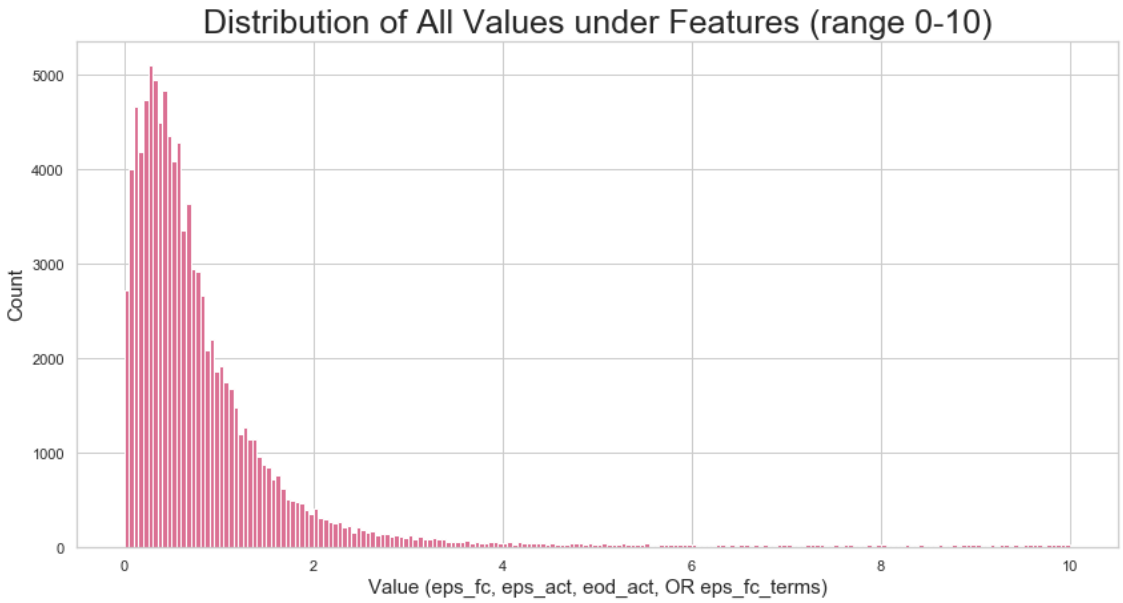

Figure 17

And the trends stay consistent: the graph, when zoomed in, still retains a strong right skew. Now, we can see that the "bulk" of the data lies around the value range **0.02 - 0.05: the "peak"** of the distribution. But just to double-check the status of being a right-skewed distribution, I created a kernel density curve:

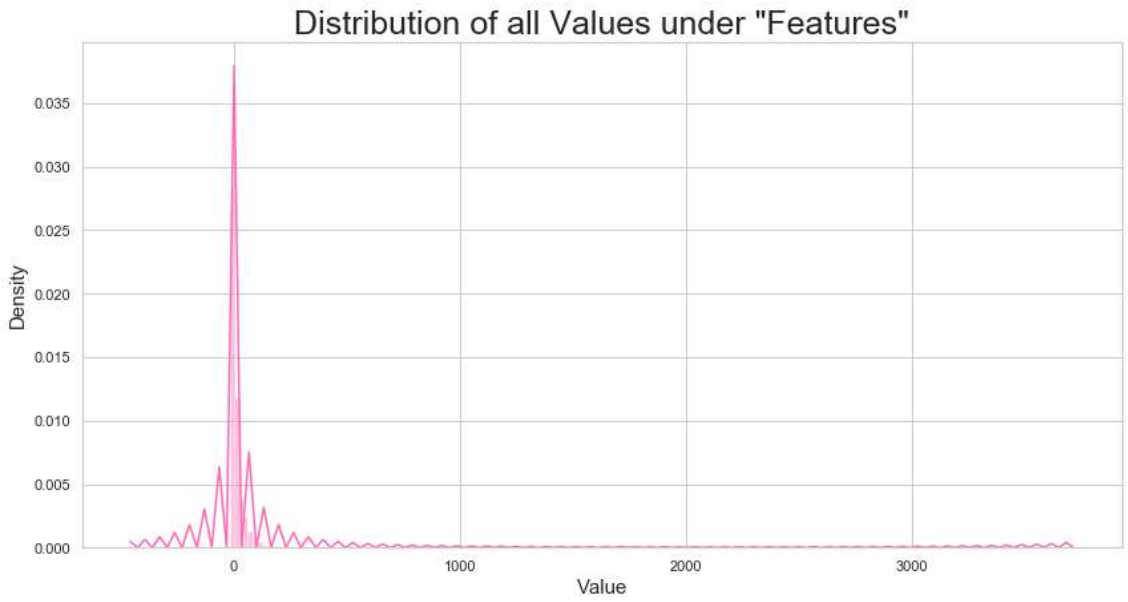

Figure 18

As shown by the kernel density curve, the distribution of all **values** under **features.csv** remains heavily right-skewed. This means that ***most values are clustered around the left side of the distribution*** where the mean, median, and mode are all located.

A **key takeaway:** There is strong congruence between the distribution patterns of *stock prices under* ***features.csv*** and ***average*** *prices* ***under avgs.csv***.

firm_id (avgs.csv)

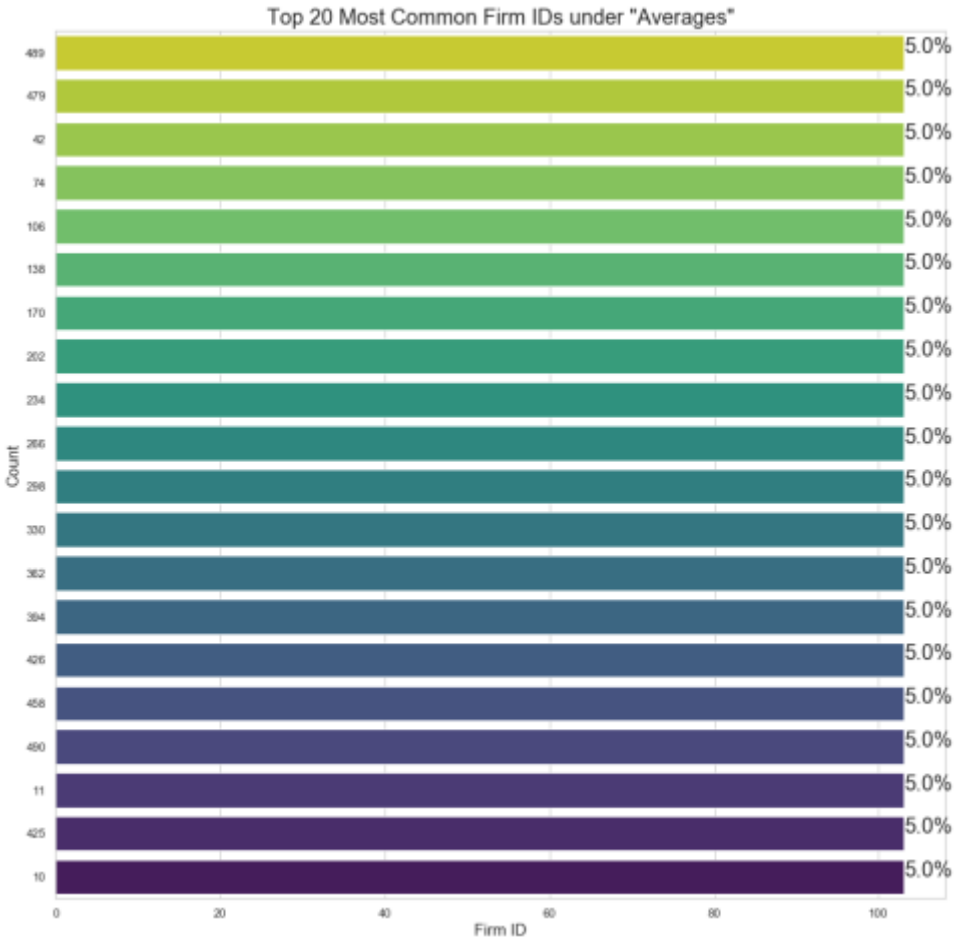

Figure 19

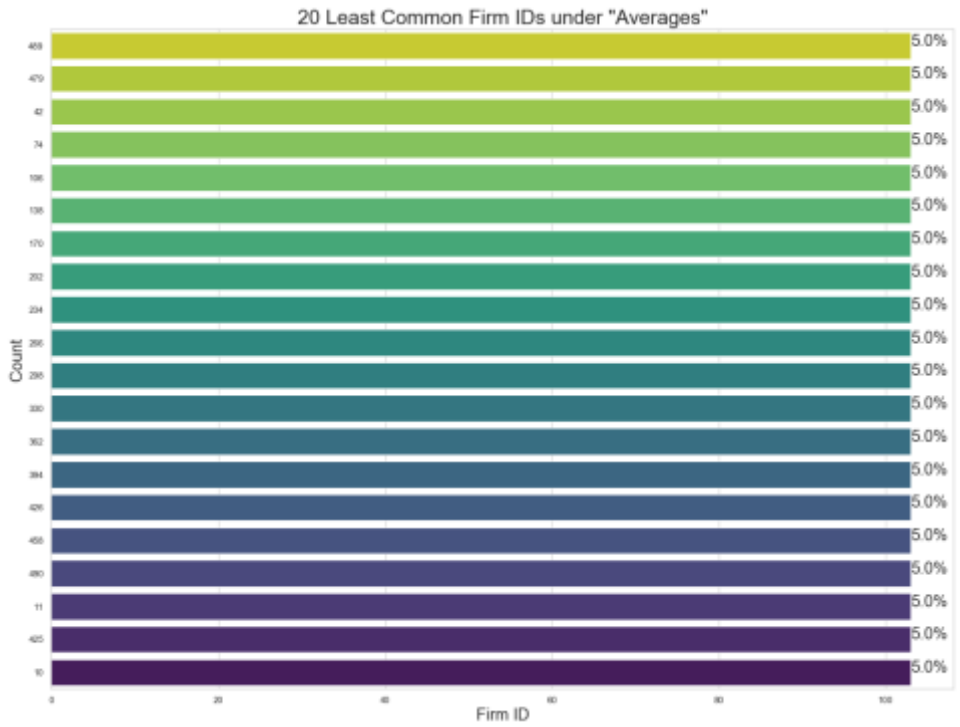

Figure 20

Both the 20 most common and least common firm IDs under **avgs.csv** all make up the same proportion of existing firm IDs: exactly 5% each. This means that under **avgs.csv**, there is a ***consistent count among all Firm IDs*** at around 105.

This means that **firm_id** counts are **consistent** across the entire **avgs.csv** dataset at 105 entries per firm ID, where null **firm_id** fields do not exist.

average *(*avgs.csv)

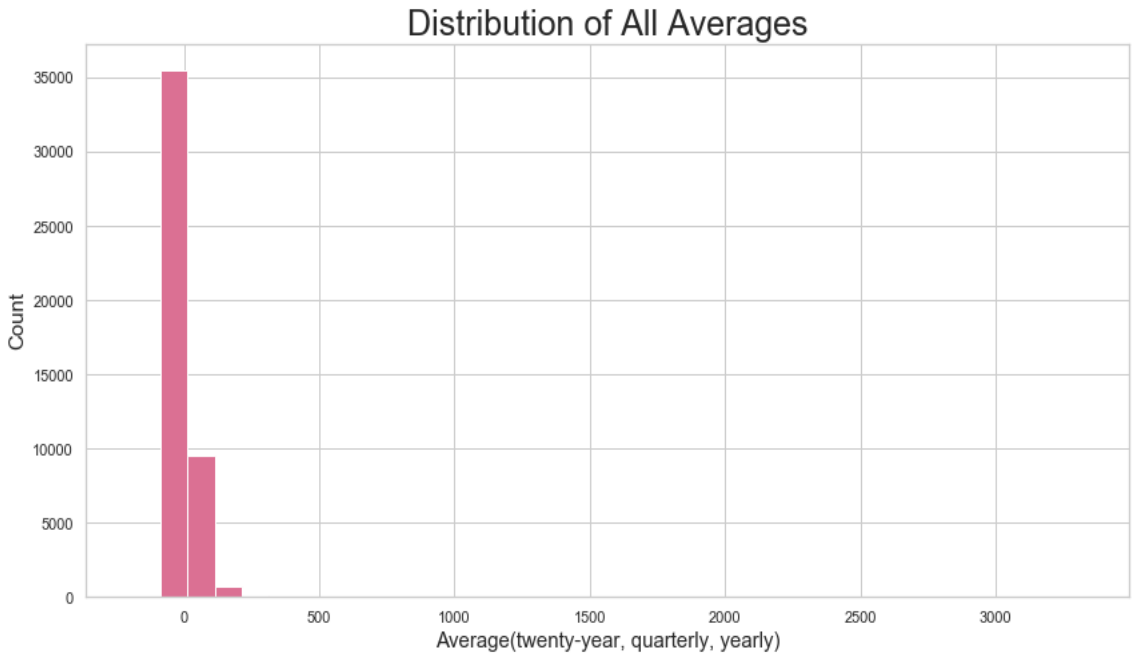

Figure 21

The trend of all **average** counts is ***heavily right-skewed***. On the x-axis, the range 0-100 contains the highest concentration of data, with around 36,000 entries. Also, the range 0-300 contains the "bulk" of all the data, which means the surrounding x-axis values contain outliers. This means that ***all values, whether under* features.csv *or* avgs.csv, *share a common theme of taking up the "bulk" of data in the same bin range.***

Question 8: How do value counts under **avgs.csv** look like after removing all outliers around the value range 0-300?

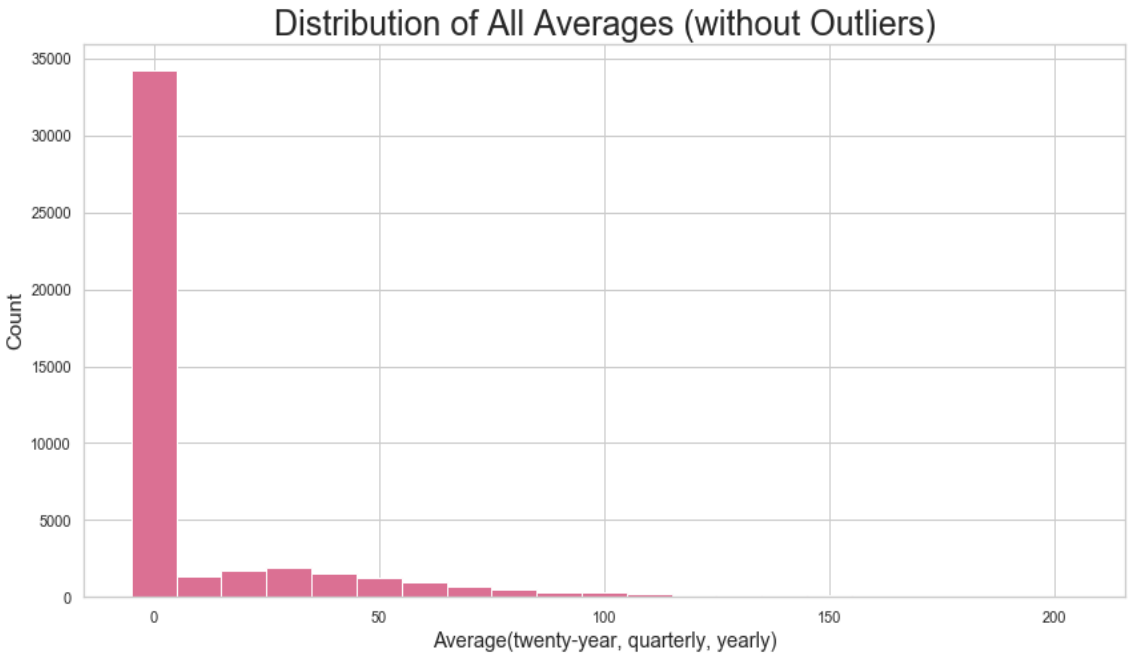

Figure 22

To answer **Question 8**, the graph appears to have, yet again, a ***strong right skew.*** This is unlike the previous graph *with* the surrounding outliers, where the right skew was much more pronounced and prominent.

The value-range 0-10 contains the bulk of all the data. This bin range houses the highest bar, depicting counts of around 12,000. The previous graph shows that the bin range 0-100 contains around 36,000 points, while this graph's 0-10 bin range contains around 34,000 points. Drawing from these observations, we can conclude that ***it's really the bin range 0-10 that contains the bulk of the data***. The remaining approximate 2,000 counts would be scattered among the bins from x-range 10 to 110. And among those remaining bins, the range 30-40 contains the bulk of that data, at around 2,000 counts.

Similar to my approach for the **value** column under **features.csv**, here I also broke down the 0-10 bin range further to examine the bulk of all the **averages**.

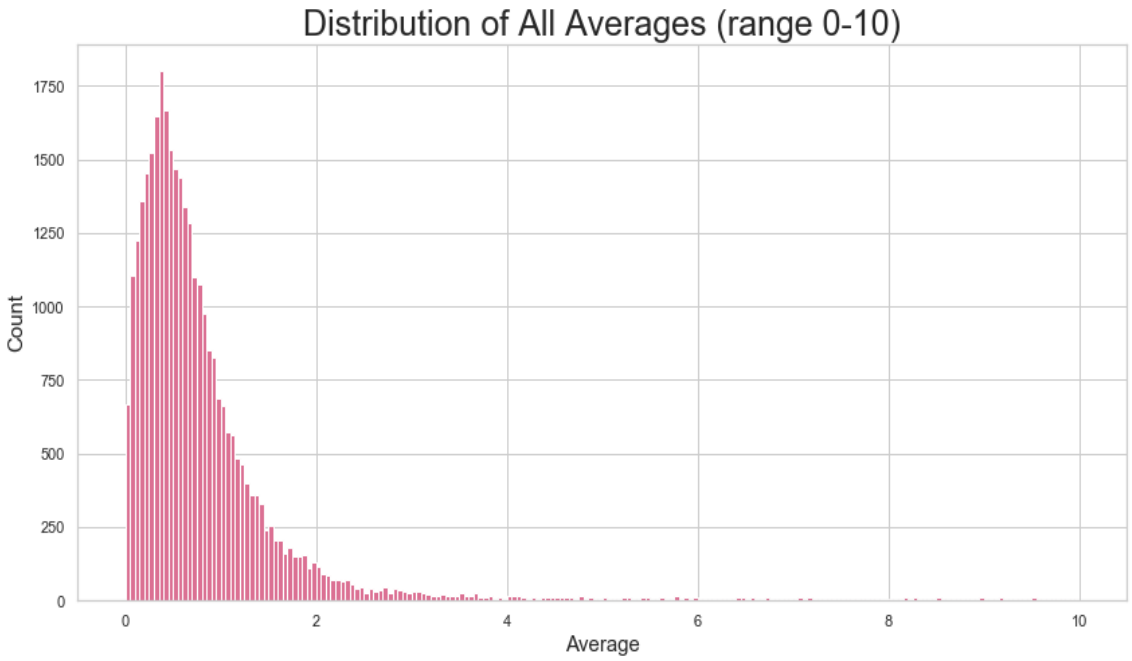

Figure 23

This graph, even when zoomed in, still retains a strong right skew. Additionally, the "bulk" of the data lies around the x-value range **0.02 to 0.05: the "peak"** of the distribution. And just to test the validity of the right-skew, I created a kernel density curve:

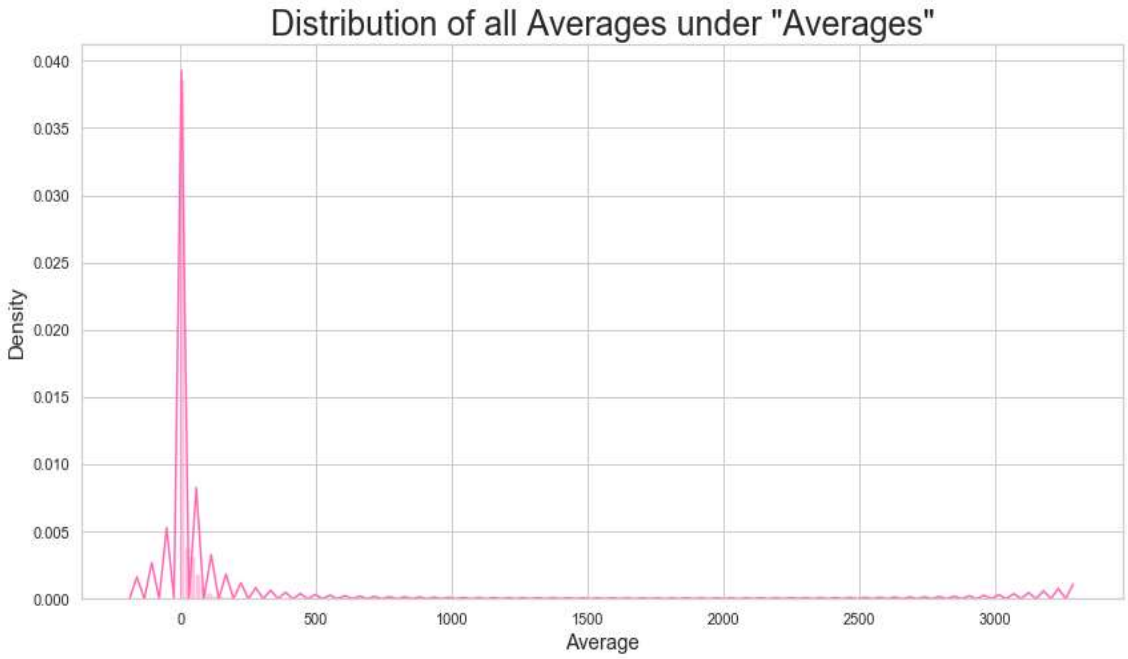

Figure 24

As shown by the above kernel density curve, the distribution of **averages** is still heavily right-skewed. This means that ***most average values are clustered around the left side of the distribution*** where the mean, median, and mode are all located. As confirmed by this KDE graph, the "bulk" of the indeed lies in the first x-axis bin range after x = 0.

average_type (avgs.csv)

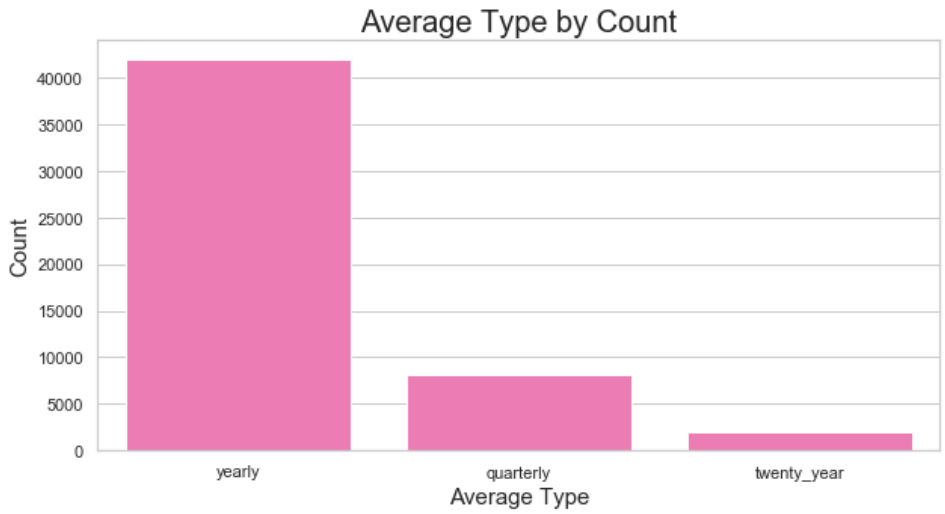

Figure 25

Under the **average_type** column, *yearly* data holds the highest number of counts while *twenty_year* data holds the least number of counts. These large gaps were to be expected, since there are more years than quarters There are more yearly averages than quarterly averages, and more quarterly averages than twenty-year averages. Here is a mathematical breakdown:

- For each firm, there exist 20 yearly averages (1999 to 2019) for each of the four **features** (80 total per firm).
- For each firm, there exist 4 quarterly averages for each of the four features (16 total per firm).
- For each firm, there is 1 twenty-year average for each of the 4 features (4 total per firm).

The above usage of "features" refers to *eps_fc*, *eps_act*, *eod_act*, and *eps_fc_terms*.

Note: the number of entries may vary for *yearly averages* because the year 1999 and 2019 are missing for some features like *eps_fc_terms*.

Now instead of examining just the average types by count, I also decided to examine them by percentage:

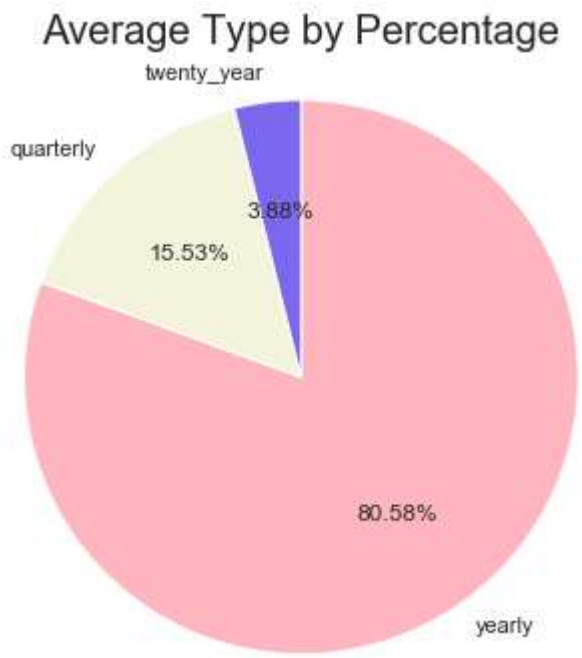

Figure 26

As expected, yearly average types make up the majority of all entries, at 80.58%. And just as equally expected, twenty-year average types make up the least portion of all entries at 3.88%. These percentages are consistent with the previous counts from the bar chart.

time_period (avgs.csv)

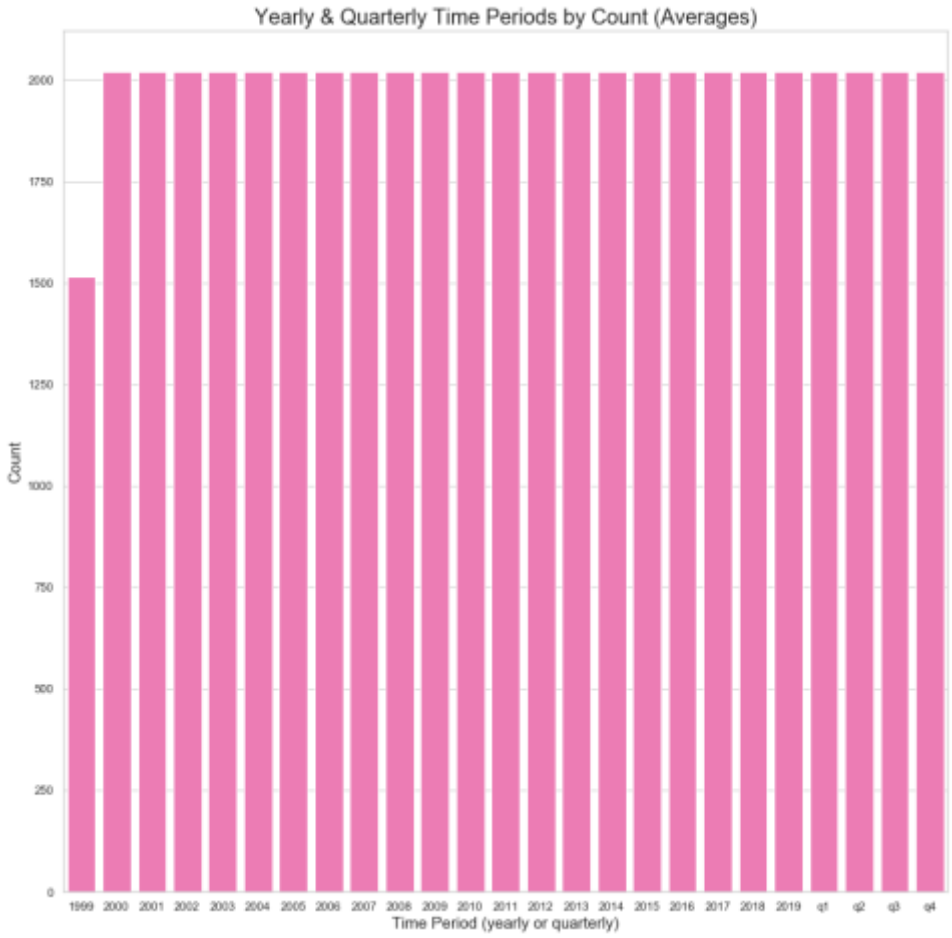

Figure 27

I decided to portray both *yearly* and *quarterly* **time periods** in the same countplot because Q1-Q4 and the years 2000-2019 show consistent counts. The relationship between years and quarters by count would be depicted more clearly through generating a countplot containing both.

All years from 2000-2019 **and** all quarters from q1-q4 show consistent counts, at just over 2,000 entries each. The year 1999 contains the least amount of counts, at only around 1,510. This is because the feature *eps_fc_terms* does not contain the year 1999. And finally, the

time_period for *twenty-year* averages does not show in the above figure because by default, all **time_period** entries with **average type** *twenty_year* are automatically set to null.

feature (avgs.csv)

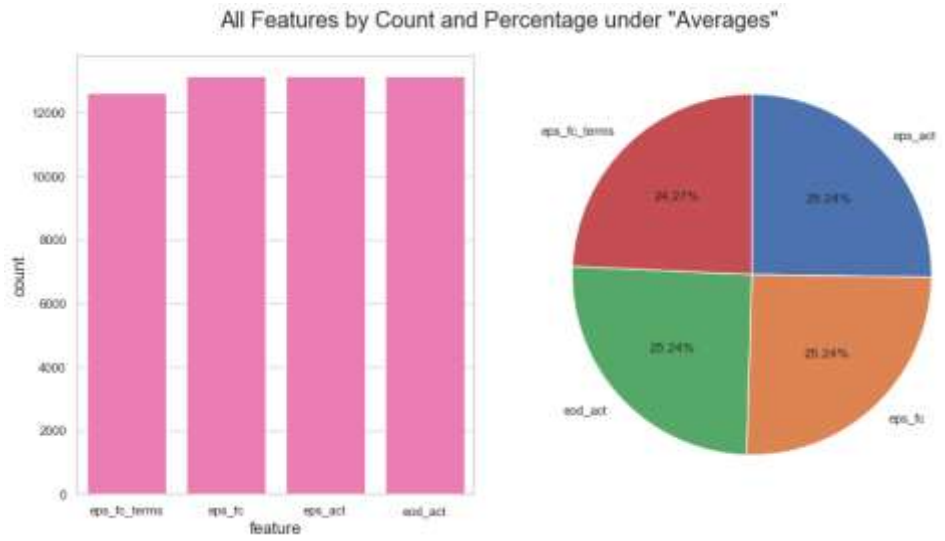

Figure 28

The *eps_act*, *eps_fc*, and *eod_act* **features** all display consistent counts at around 13,000 entries each (25.24% each). The *eps_fc_terms* feature is the only feature type to deviate in counts from the others, at around 12,500 counts (24.27%): only by 500 counts less. Additionally, it makes sense that *eps_fc_terms* contains missing data because the year 1999 was not included while gathering this data. This will effectively remove around 500 entries: not a substantial amount of data.

D) Bivariate Exploration

Stock Price Types vs. Corresponding Values (features.csv)

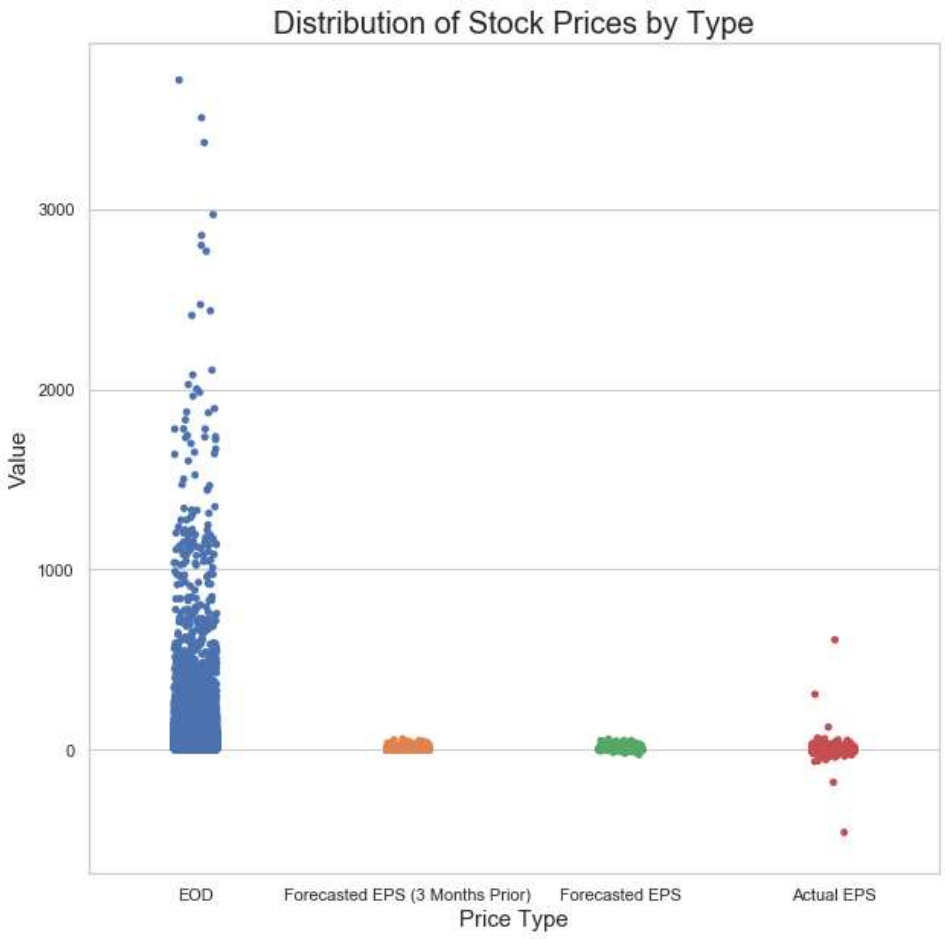

Figure 29

The EOD Price Type displays the most variance compared to the other stock types. Forecasted EPS (3 months prior) and current Forecasted EPS are the only stock prices to appear uniform in distribution: their price values are clustered around 0. Actual EPS contains the most outliers compared to the other stock price types: around *Actual EPS*, there are 5 outliers. And since *Actual EPS* has the most outliers, it turns out that ***forecasters'***

predictions have been much more uniform in the approach of predicting future EPS values, with a result of avoiding any type of variance or chance of outliers.

Average EOD Price by Term (features.csv)

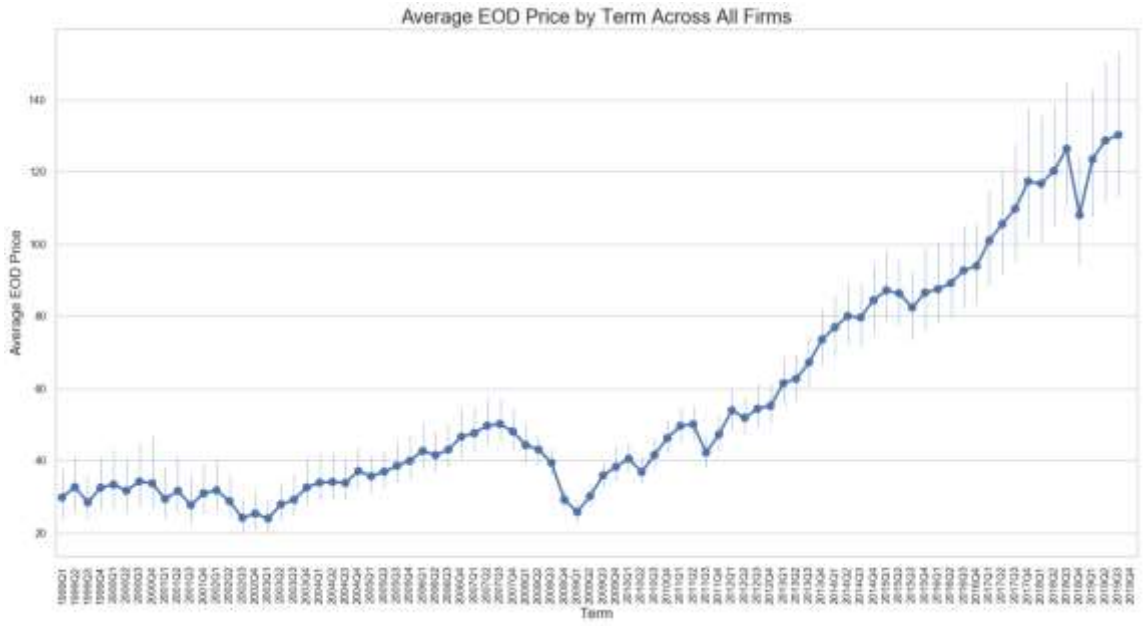

Figure 30

The above graph depicts a positive linear trend for all quarters spanning from 1999 - 2019. The average EOD Price troughs during 2009, Quarter 1. The start of the trend in this temporary downfall starts in 2007, Quarter 4, but the average EOD price recovers shortly after at only 1 quarter later on 2009, Quarter 2. In layman's terms, this generally positive linear trend shows that ***all 505 firms in the S&P 2019 Index have been getting "richer" over the past 20 years***.

In order to smooth out the above trend to display general patterns of the data, I decided to create a *moving average* version of the graph. That is, instead of relying on the raw data, I

would instead rely on 3-year rolling averages—the average of all past values up to the 3rd year, then for the 4th year, and so forth.

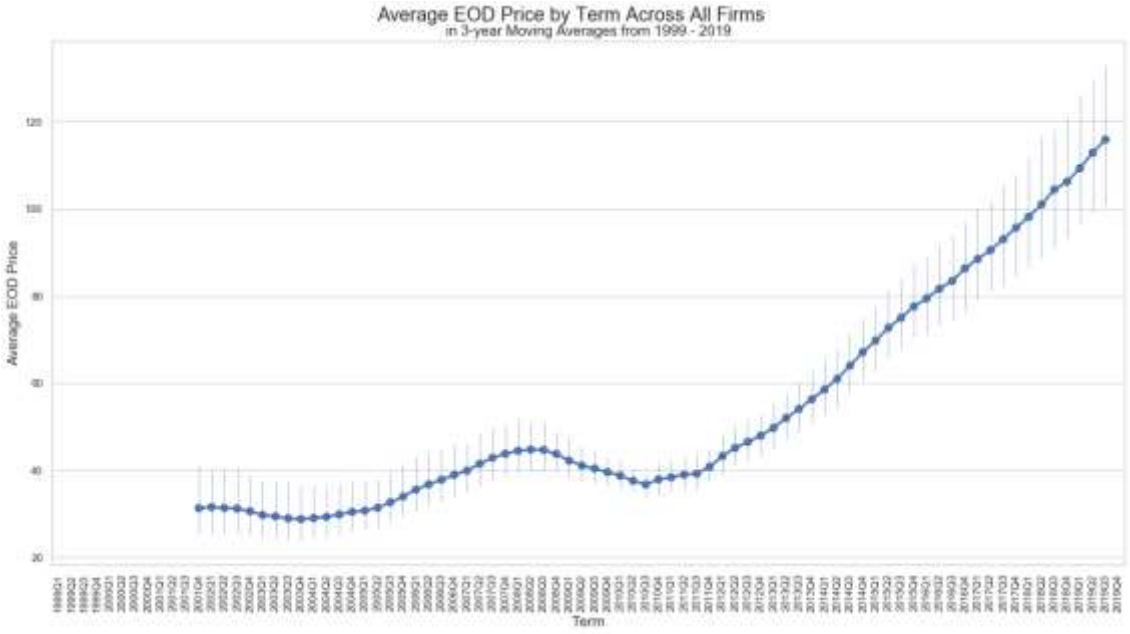

Figure 31

After smoothing out the "bumps" caused by individual points, this graph displays the general trend of the previous graph. Here we can observe the consistent patterns of a miniature mountain and a trough, before EOD prices started increasing in the direction of a more linear, stable trend after 2010, Quarter 2.

Prediction Error by Term (features.csv)

Prediction error measures the *difference between actual and forecasted EPS:* that is, *eps_act* - *eps_fc*. **Negative prediction errors** indicate that the forecasters were generally ***optimistic*** in their predictions for that fiscal year (*forecast* > *actual*); **positive prediction errors** indicate that the forecasters were generally ***pessimistic*** in their predictions for that fiscal year (*forecast* < *actual*).

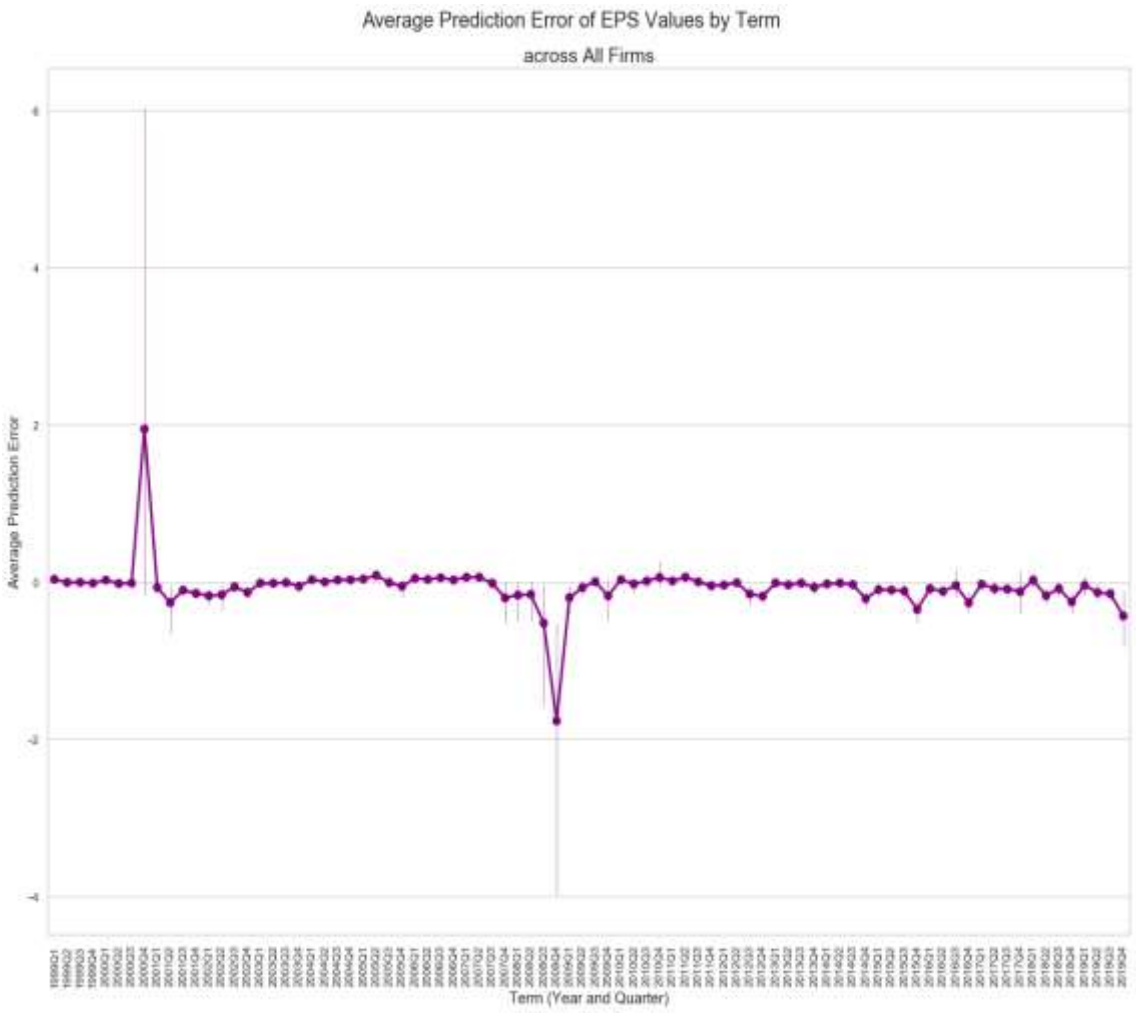

Figure 32

Forecasters were most optimistic in 2008, Quarter 4. Meanwhile, forecasters were most pessimistic in 2000, Quarter 4. The overall trend (ignoring the previous two outlying points) indicates no linear relationship. Over the past twenty years, forecasters' forecasted EPS appear to be generally consistent with the actual EPS prices, as all average prediction errors center around 0.

To examine the trends among years instead of terms, I created the following figure:

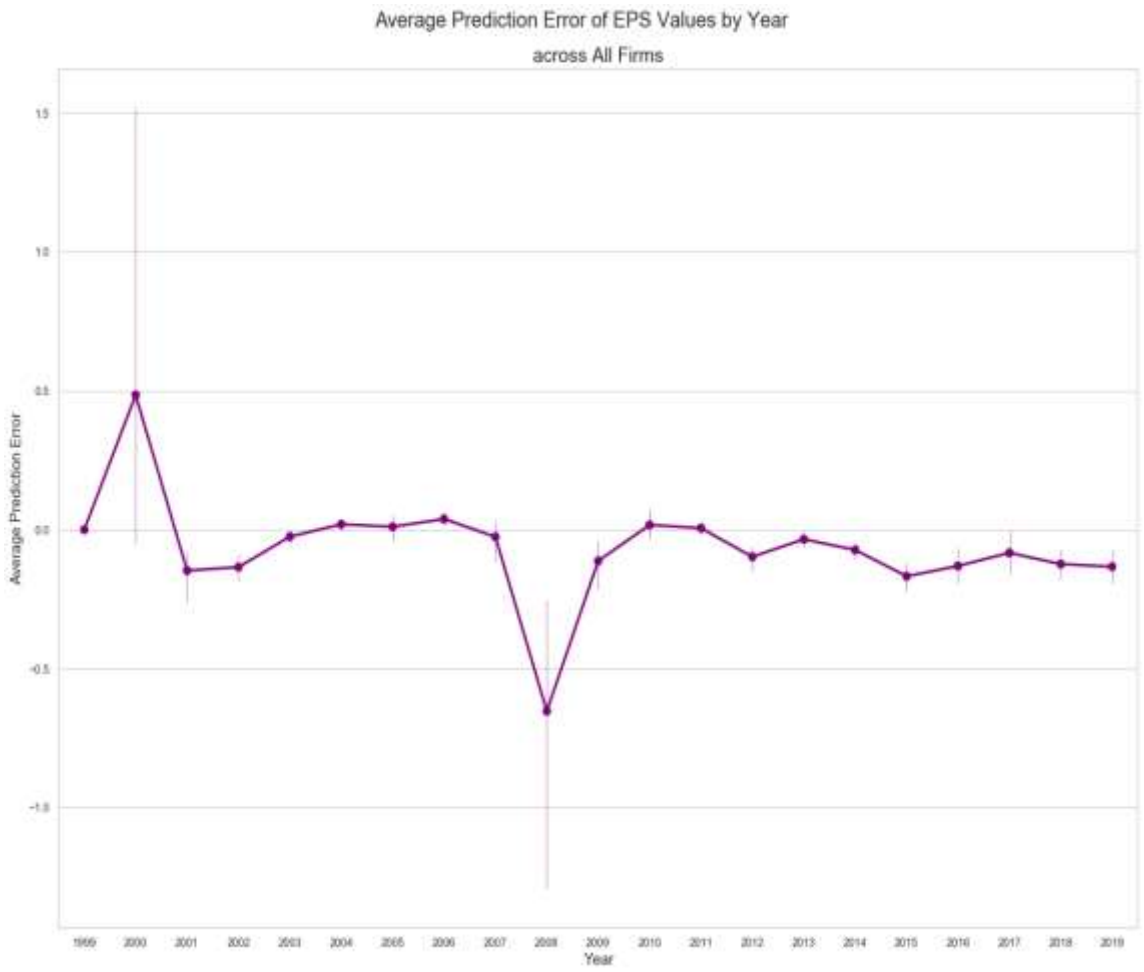

Figure 33

Forecasters were most pessimistic in the year 2000. From the displayed error bar, the year 2000 displays one of the widest variances among average prediction errors ranging from 0 to 0.5. Likewise, forecasters were most optimistic in the year 2008. 2008's error bar displays one of the widest variances among average prediction errors: ranging from -0.25 to -1.5. The overall trend stays consistent with the previous trends by term: consistent, relatively stable, and containing no slope.

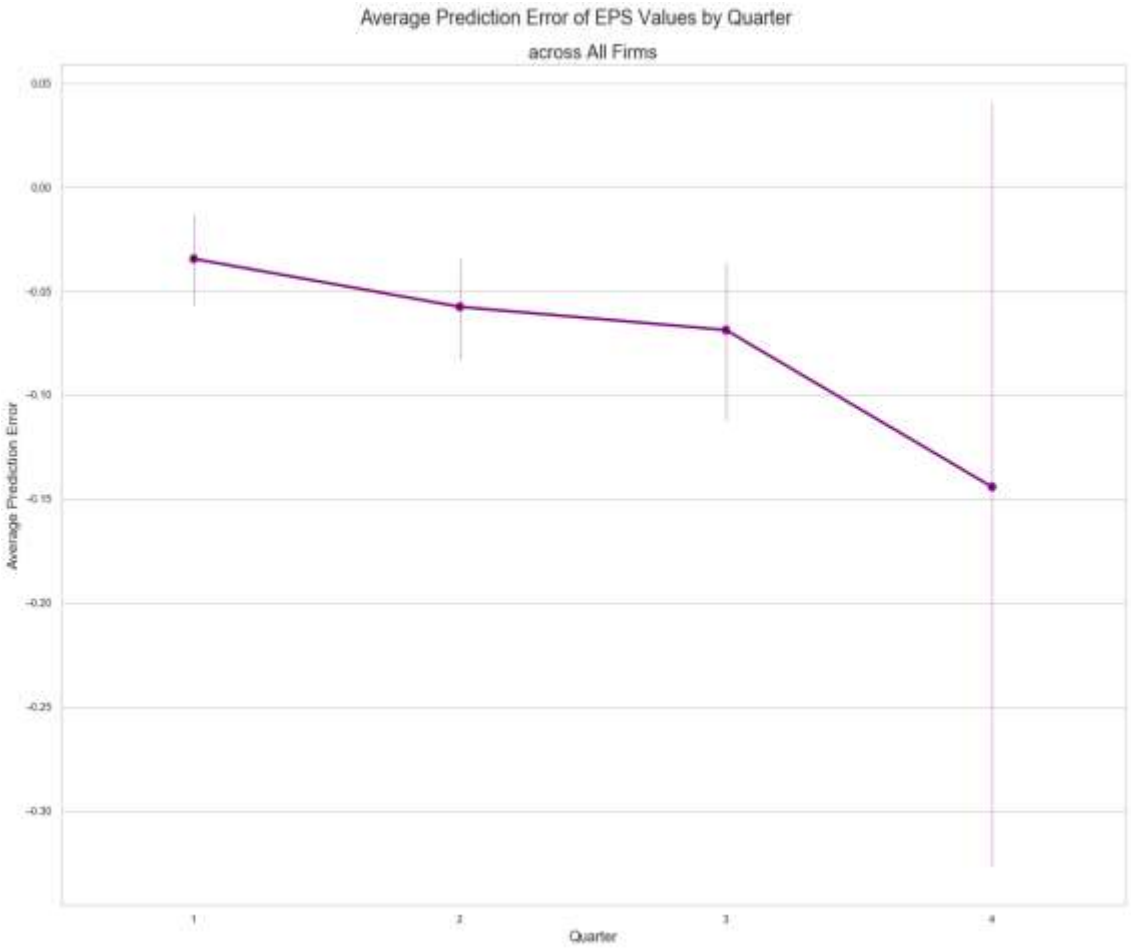

Figure 34

The later the quarter, the more optimistic forecasters become in their predictions. I draw the connection that this is because people need time to familiarize themselves with the current year's stock market dynamics to create more confident predictions. Each new quarter brings the familiarity of previous mistakes made in previous quarters of that year. This, combined with building familiarity of the year as it advances, could be two psychological causes to this tendency to become more optimistic over EPS forecasted in any given year.

Moving Average Prediction Error by Term (features.csv)

Just to "smooth out" the previous graphs to display general trends, I plotted a 3-year rolling average prediction error against term periods in a Seaborn pointplot.

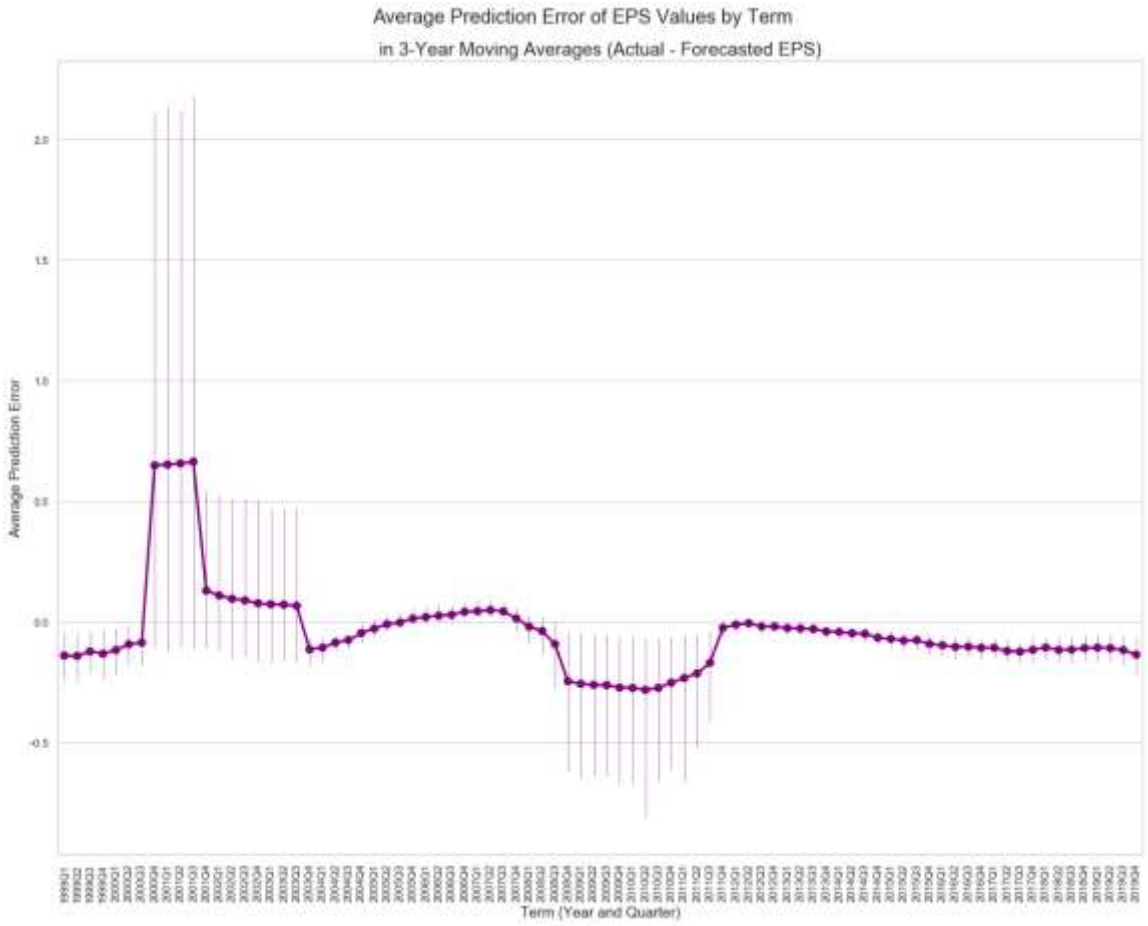

Figure 35

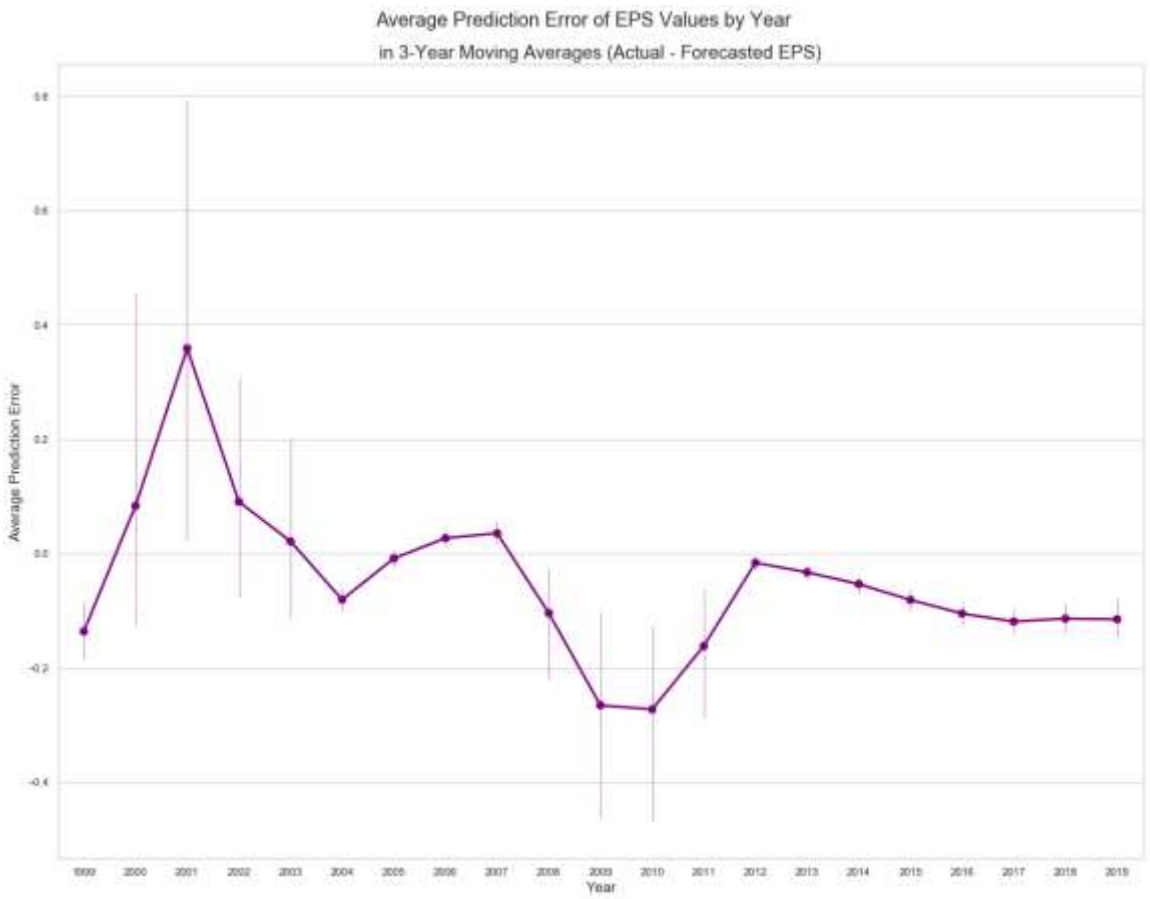
Average Prediction Error of EPS Values by Year
in 3-Year Moving Averages (Actual - Forecasted EPS)
Average Prediction Error
Year

Figure 36

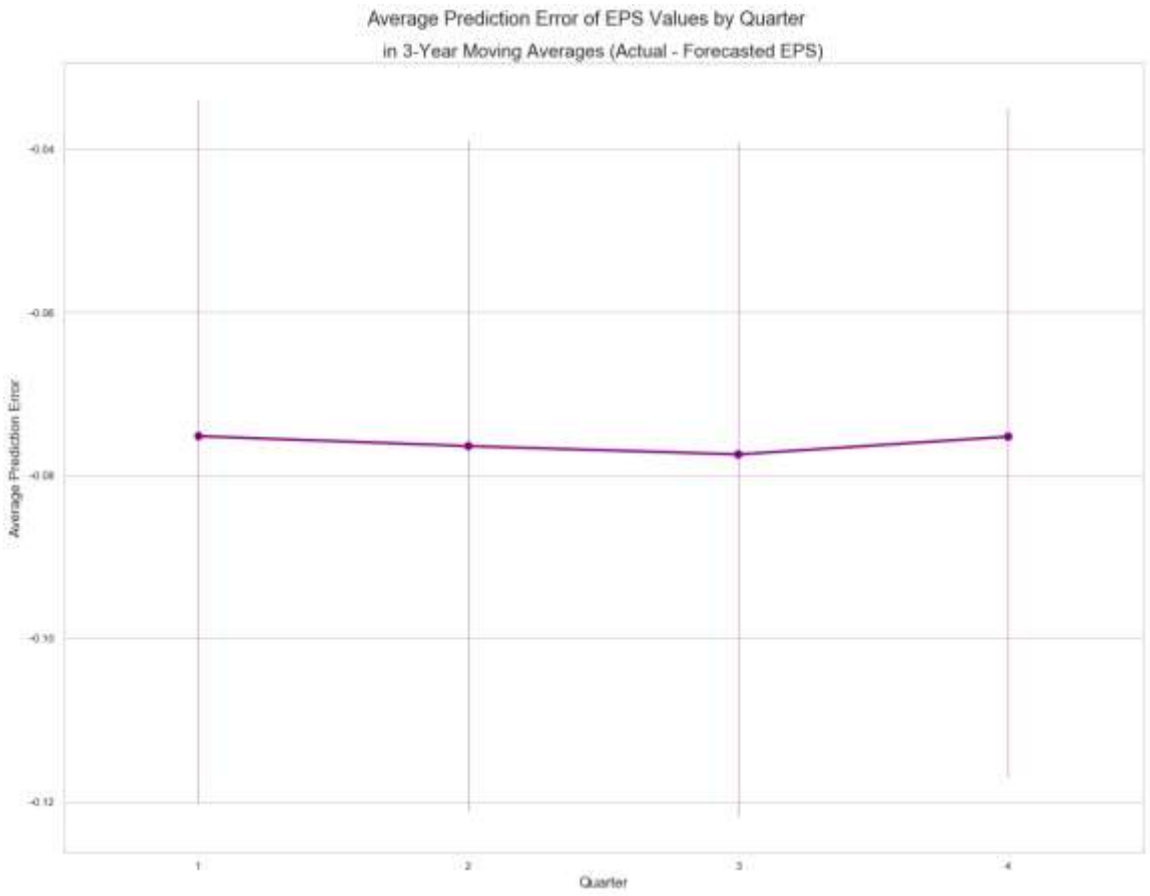

Figure 37

Compared to their non-moving-average counterparts, the above pointplots depicting 3-year moving averages all display a consistent slopeless trend, even including small periods of optimism vs. pessimism.

Prediction Errors Among Twenty-Year, Yearly, and Quarterly EPS Averages (avgs.csv)

I initially wanted to compare all 505 firms, as categorical variables in the x-axis, to certain numerical values filtered by feature or average type. However, crowding 505 firm tickers on a single axis would cause an unnecessary amount of noise in the data, and would blur

the overall picture. My approach to remedy this problem is to ***isolate the top 20 most inaccurate firms*** (largest absolute prediction error) ***and 20 most accurate firms*** (smallest absolute prediction error).

First, I convert all prediction error entries to their absolute values. Then I filter two separate DataFrames—for the ***most inaccurate firms***, I sort the DataFrame in descending order by the **difference_abs** column, which holds all the *absolute prediction errors.* Then, I need to isolate the first 20 unique occurrences by firm ID, dropping all duplicates, so that we have 20 unique firm IDs signifying to be the most inaccurately predicted firms. Then, I take that list of 20 firm IDs and plot their *real predicted error* by firm ticker.

Twenty-Year Averages

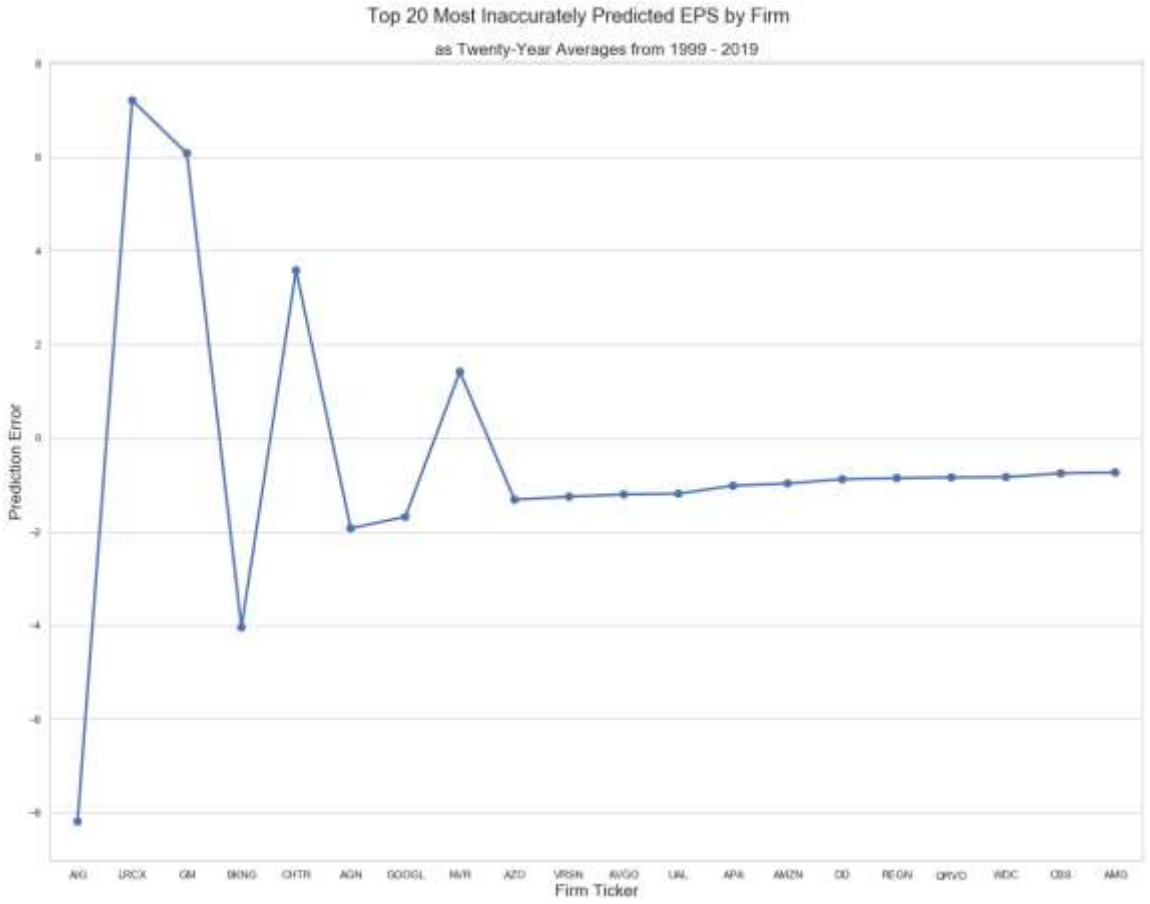

Figure 38

The 9 most inaccurately predicted firms by *twenty-year* averages are AIG, LRCX, GM, BKNG, CHTR, AGN, GOOGL, and NVR.

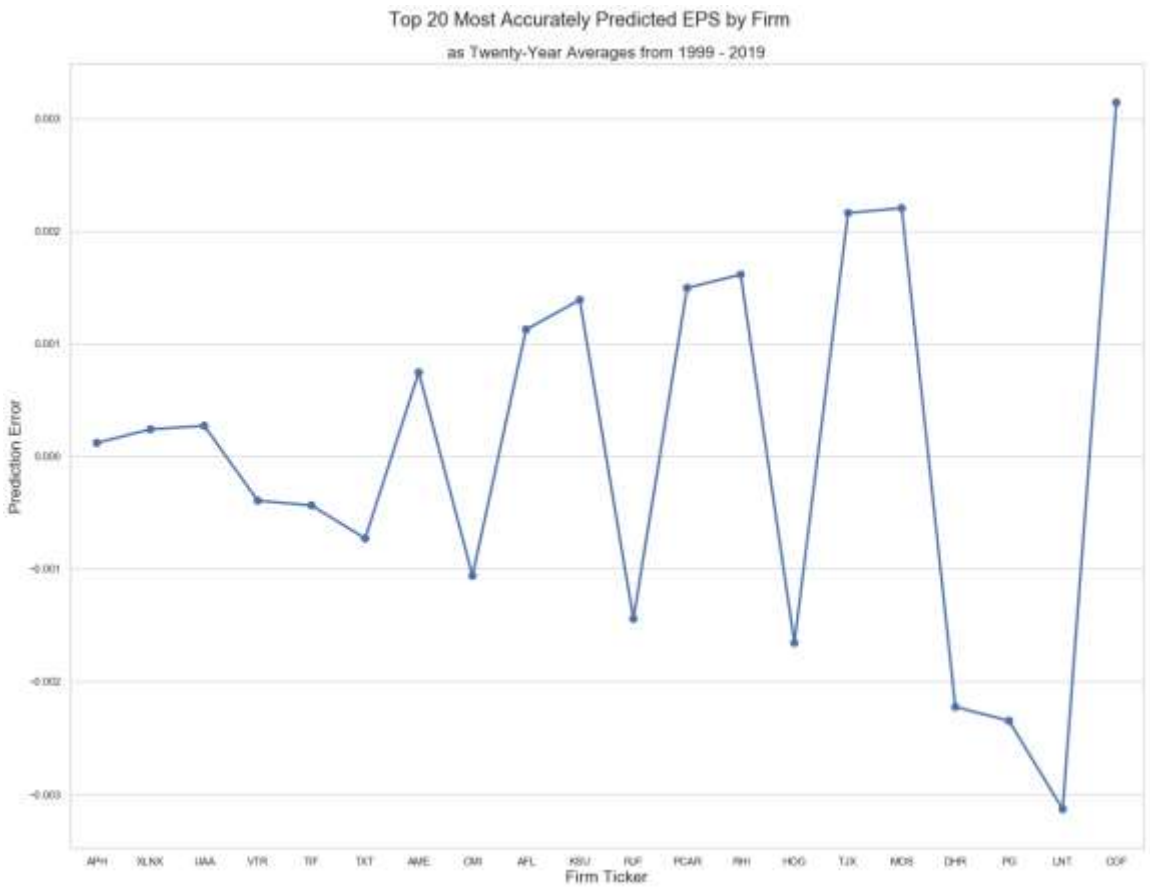

Figure 39

The above trend may initially appear unstable, but this is a tricky and wrong assumption. The points above do not depict the *magnitude* of prediction errors, but rather the raw prediction error. All firm tickers were ordered by *magnitude* of prediction error rather than the *value* itself. This is because only magnitude can accurately order firm IDs ***regardless of forecasters' optimism or pessimism*** in any given period.

Visualizing the absolute y-value corresponding to each firm ticker above proves that the trend among the top 20 most accurately predicted firms is a steady, positively linear pattern. This indicates that the most accurate forecasts always approach a predicted error of 0.

Quarterly Averages

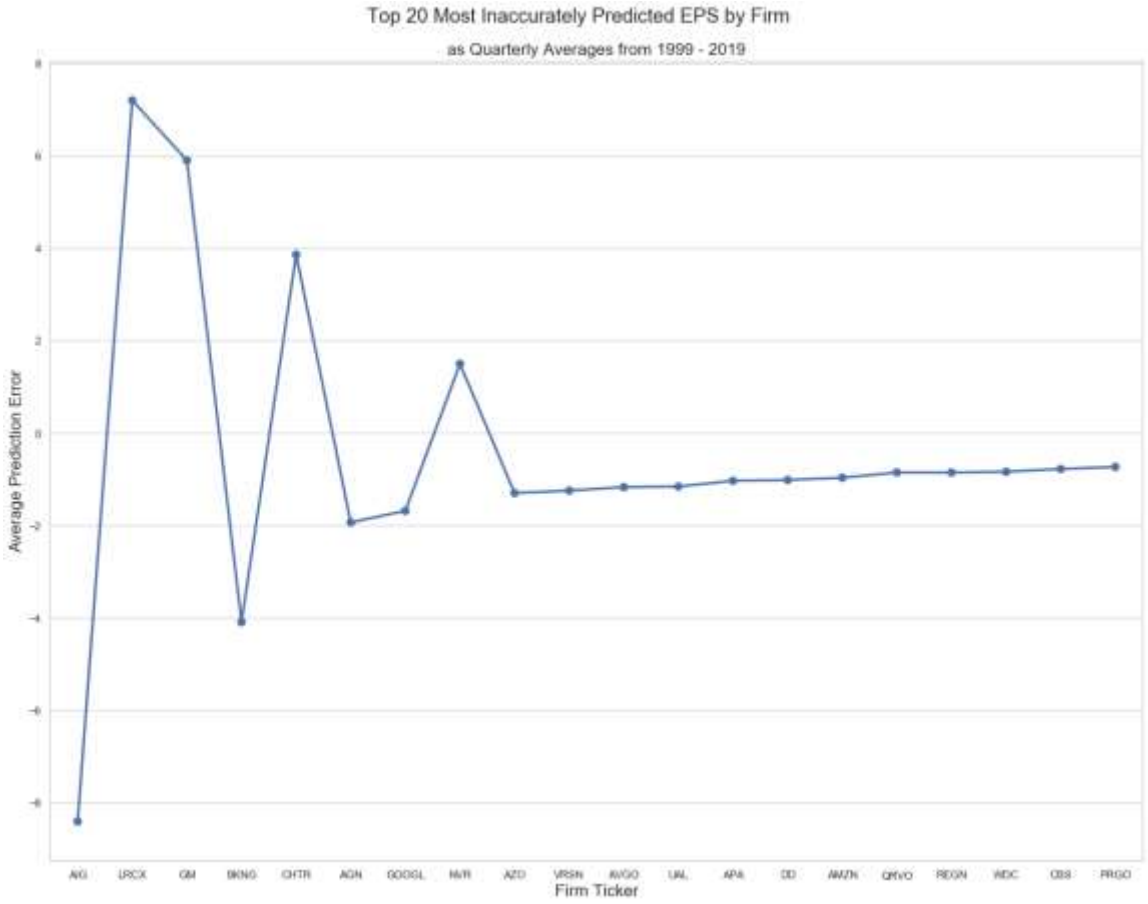

Figure 40

The 9 most inaccurately forecasted firms by *quarterly* average prediction errors are AIG, LRCX, GM, BKNG, CHTR, AGN, GOOGL, NVR, and AZO.

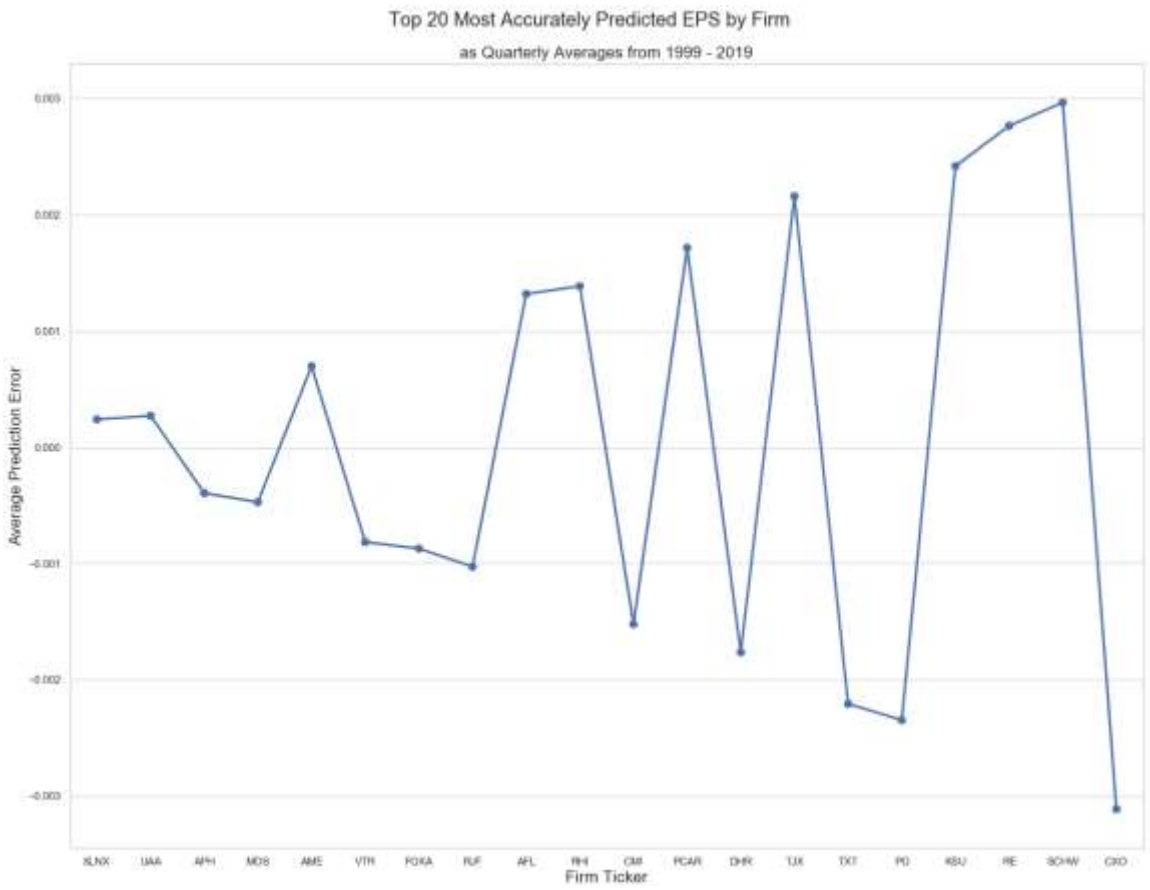

Figure 41

Similar to the previous twenty-year prediction point plot, this graph shows a general oscillation between negative and positive y-values. However, this ultimately does not matter, because the absolute distance between 0 and all above y-values decreases, therefore approaching x = 0.

Yearly Averages

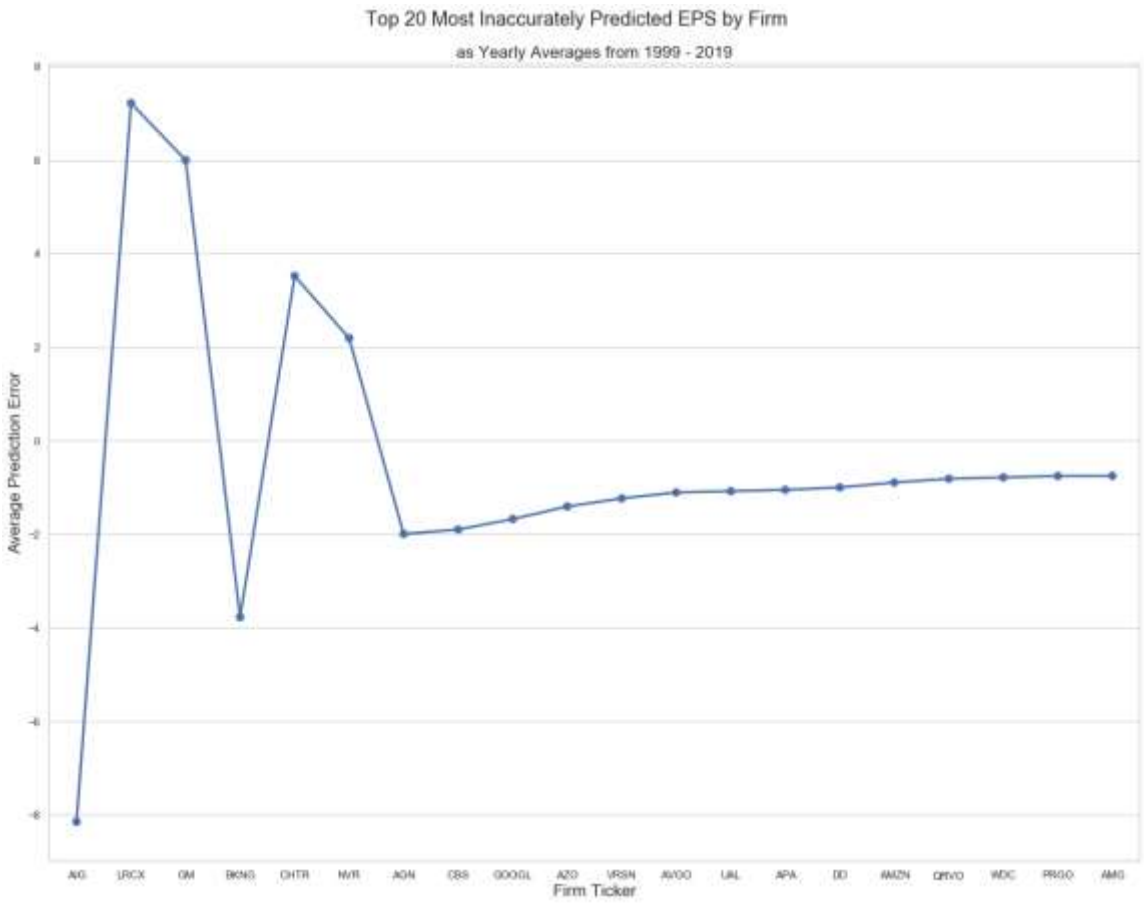

Figure 42

The top 7 most inaccurately predicted firms by *yearly* average prediction EPS error are AIG, LRCX, GM, BKNG, CHTR, and NVR.

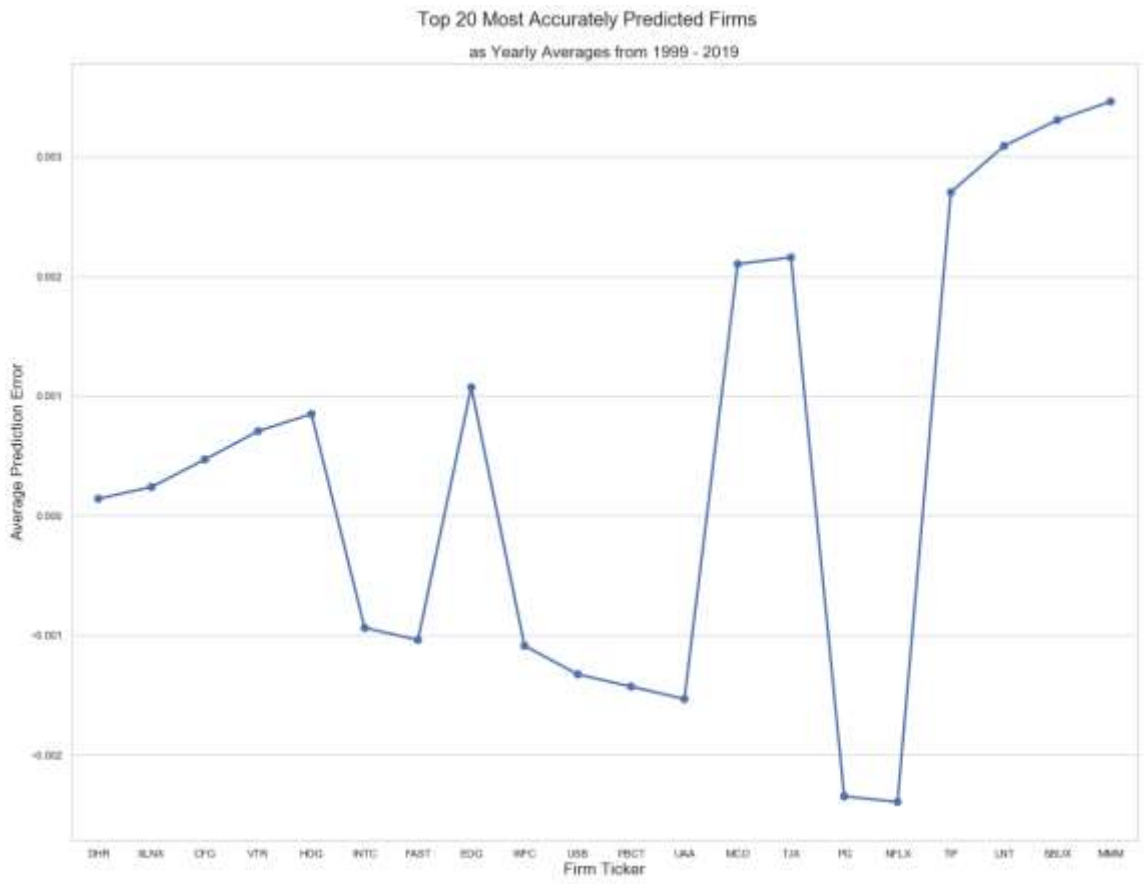

Figure 43

Consistent with previous trends, this graph of the 10 most accurate firms by *yearly* average predicted errors is varied in its collection of positive and negative values inching closer towards zero.

All Average Types Combined

To generate graphs accounting for all twenty-year, yearly, and quarterly averages, I decided to concatenate all three DataFrames containing data only to that specific average type. After combining all three DataFrames to include data for all averages among all average types, I generated the graphs below.

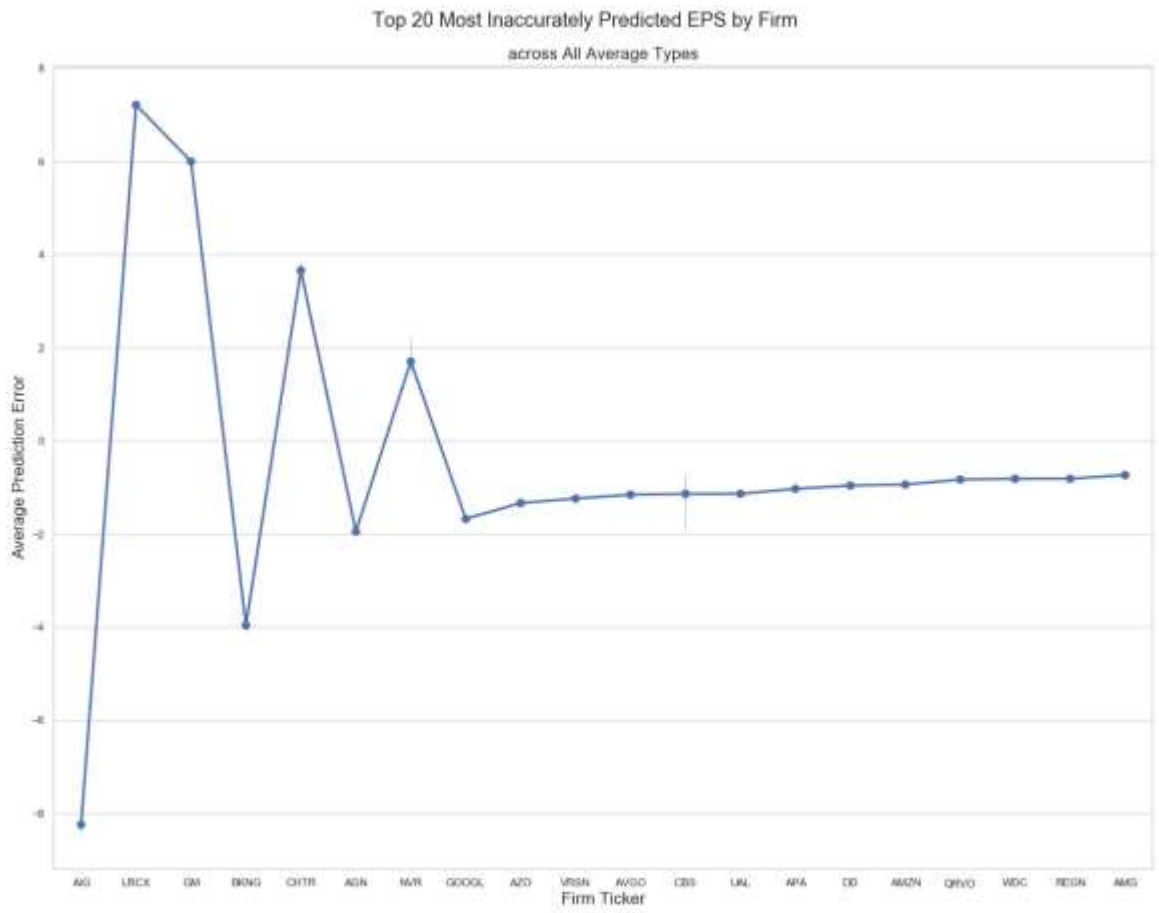

Figure 44

The top 8 most inaccurate firms by all average types are AIG, LRCX, GM, BKNG, CHTR, AGN, NVR, and GOOGL.

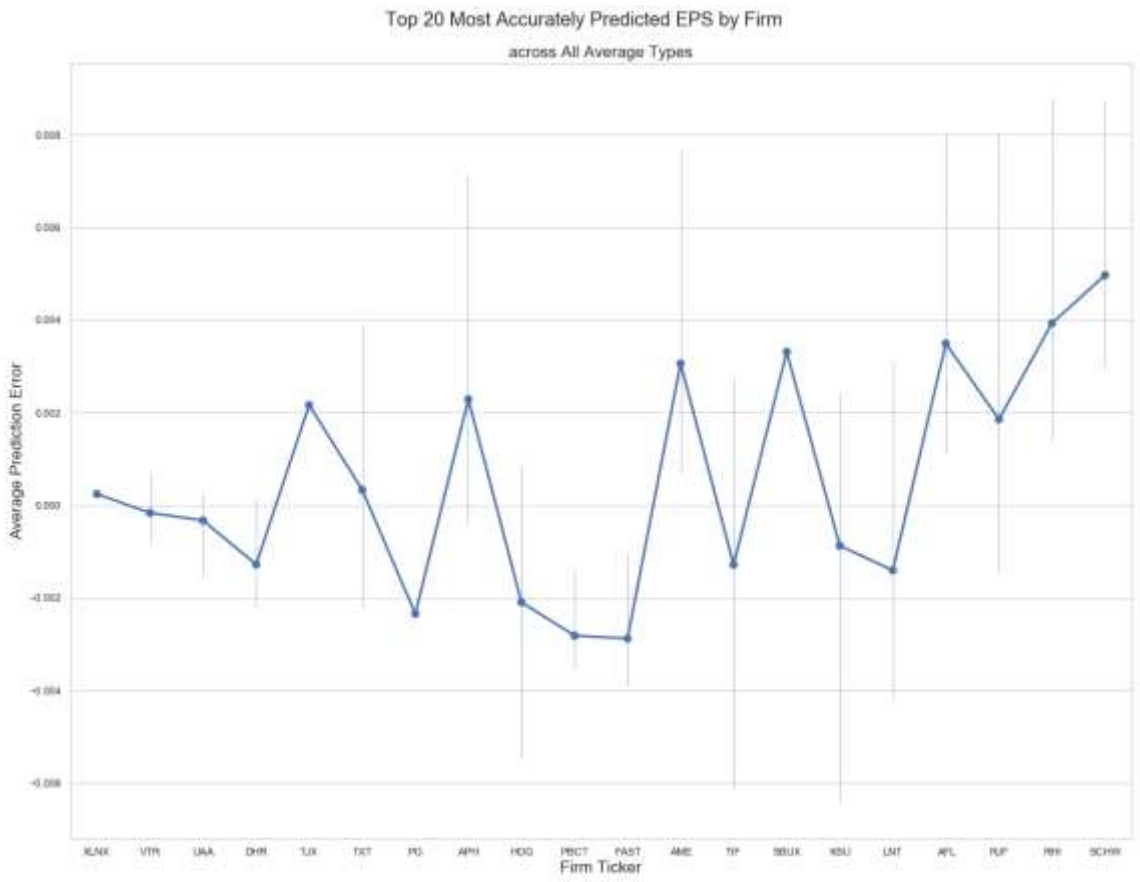

Figure 45

No new observations made for this figure; the same trend of all values' absolute values steadily approaching 0. Nothing new.

Now let us summarize the above graphs and isolate only on the top 20 most inaccurately predicted firms. Beneath each of the four above graphs, I isolated the first couple of firms before the general trend dipped and began to normalize, approaching 0.

- *Twenty-year* average prediction errors: AIG, LRCX, GM, BKNG, CHTR, AGN, GOOGL, and NVR

- *Quarterly* average prediction errors: AIG, LRCX, GM, BKNG, CHTR, ANG, GOOGL, NVR, and AZO
- *Yearly* average prediction errors: AIG, LRCX, GM, BKNG, CHTR, and NVR
- *All* average prediction errors: AIG, LRCX, GM, BKNG, CHTR, AGN, NVR, and GOOGL

After examining these three summaries of firms pre-stabilization, some interesting observations pop up. For one, AIG is consistently the most inaccurately predicted firm across all *twenty-year*, *quarterly*, and/or *yearly* average types.

The firms all average types share—AIG, LRCX, GM, BKNG, CHTR, and NVR—are all ranked in the same order on the x-axis. Across all average types, these firms are consistent in their order of inaccuracy before average prediction error trends start normalizing towards 0.

AGN and GOOGL appear in all the above DataFrames of average types except *yearly*. This means that GOOGL EPS is more accurately forecasted solely when examined on a yearly basis. And finally, AZO appears only in the top 20 most inaccurate *quarterly* average prediction error DataFrame.

Regression Plots for EPS Averages (avgs.csv)

To visualize the relationship between forecasted and actual EPS where forecasted EPS is the independent variable, I generated a mixture of scatterplots with regression lines depicting a 95% confidence interval.

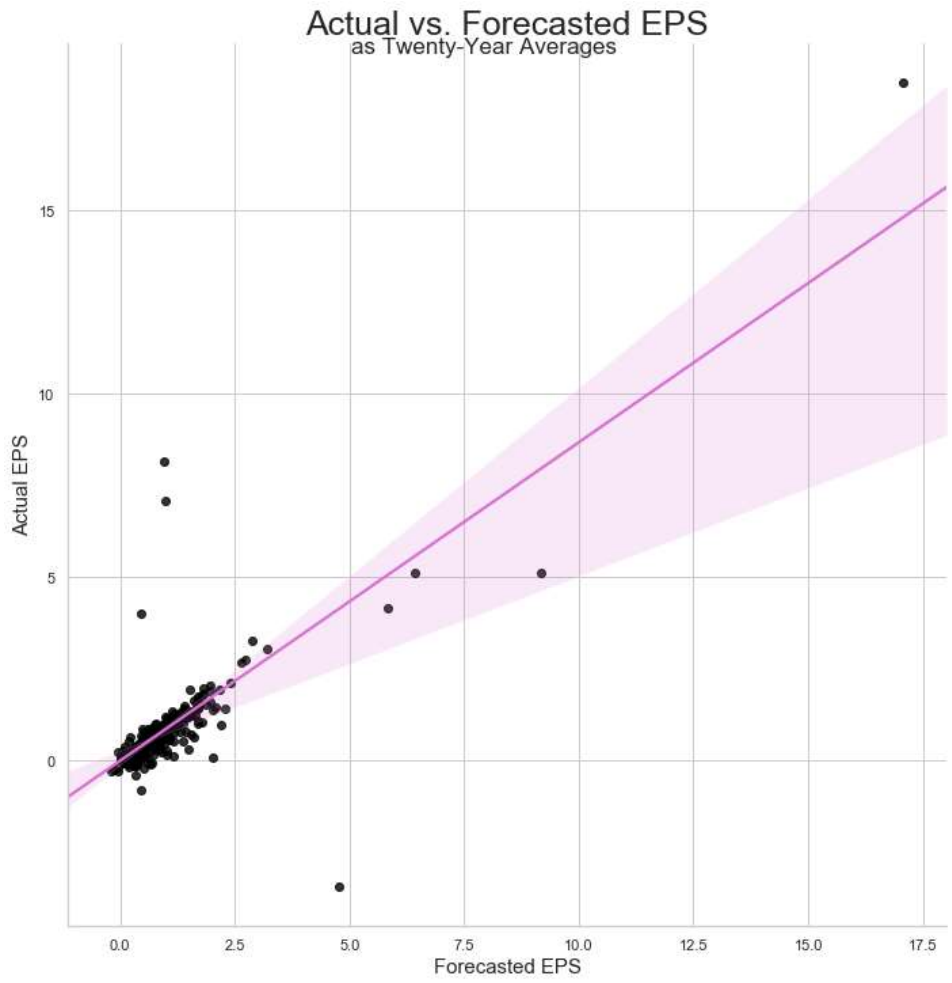

Figure 46

The translucent band lines above represent a *bootstrap 95% confidence interval* generated for the estimate.

The relationship between actual and forecasted EPS shows a wide variance from x = 0.0 to x = 9.0. Also, most of the points are clustered between 0.0 and 2.5 on the x-axis. Within that range there is only one outlier at around x = 0.025. Outside of that x-range, all other points are outliers, which contribute not only to the "scattered" type of variance shown in the scatterplot, but also lead the trend towards a generally positive linear direction (after ignoring the outlier at x = 5.0, y = 0.25, of course).

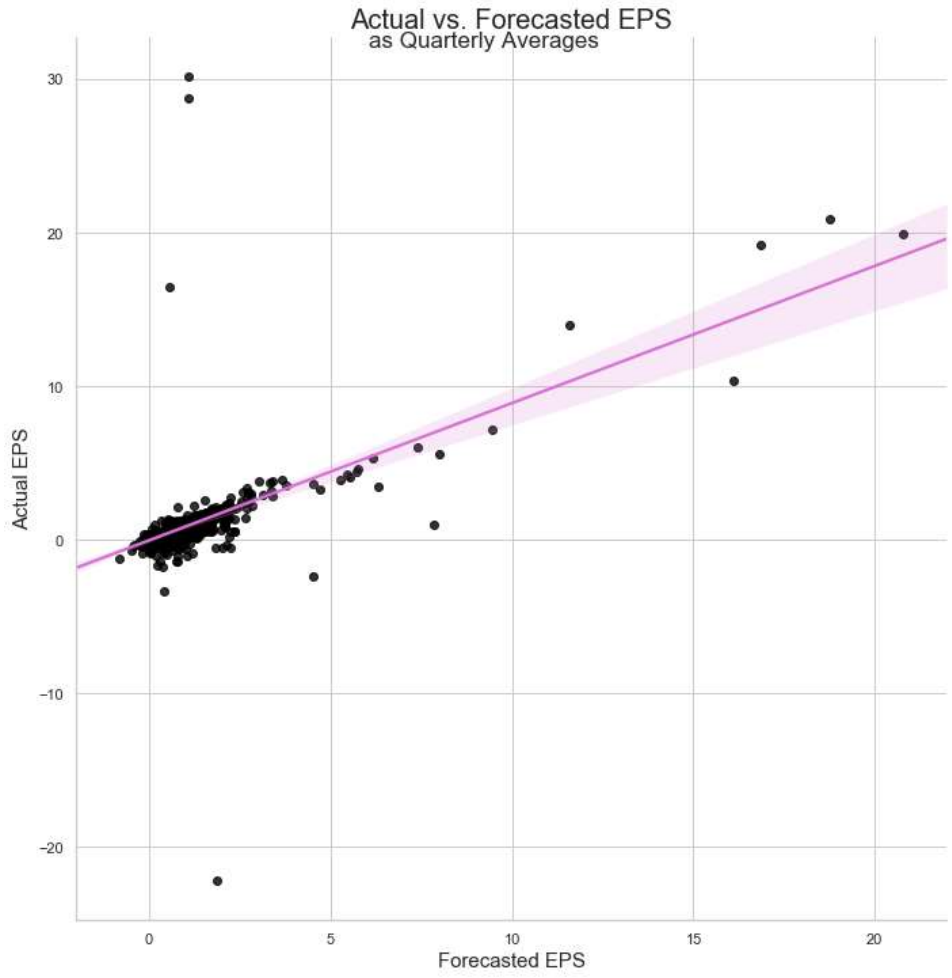

Figure 47

Unlike the previous scatterplot depicting twenty-year EPS averages, the confidence interval for quarterly averages is much ***narrower.***

There is a similar pattern of all points being “clustered” at the beginning of the x-axis, with outliers being abundant outside of that initial x-range. Here, that range is from x = -1 to x = 3.5.

Outliers among EPS quarterly averages are much more varied in scale, which causes this regression line to be much less steep than the regression line for twenty-year EPS averages.

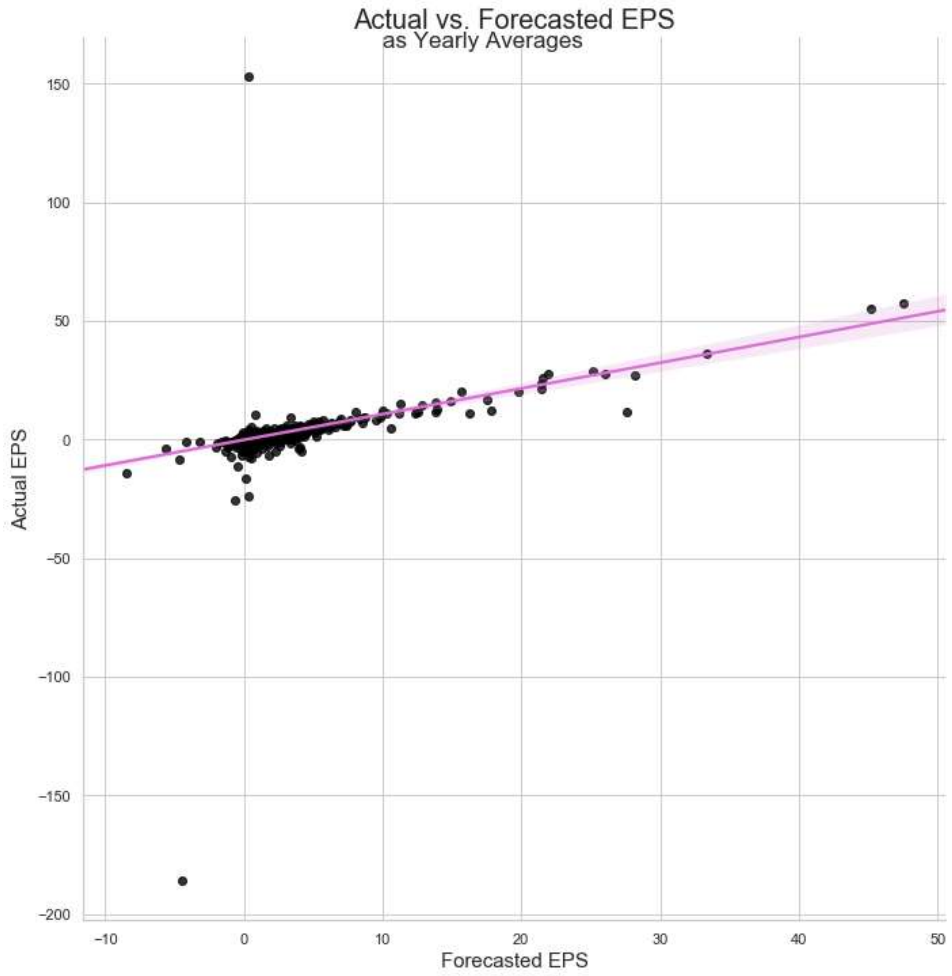

Figure 48

There is significantly less variance among yearly EPS averages, with only 2 outliers at x = -4 and x = 0.5. Also, compared to twenty-year and quarterly EPS averages, the 95% confidence interval for *yearly* EPS averages is much narrower.

The points lying on either side of the regression line—the residuals—tend to stay closer to the depicted trend, "hugging" the regression line as it slopes upward. This implies that there is ***lower bias*** in the relationship between forecasted and actual ***yearly EPS averages***.

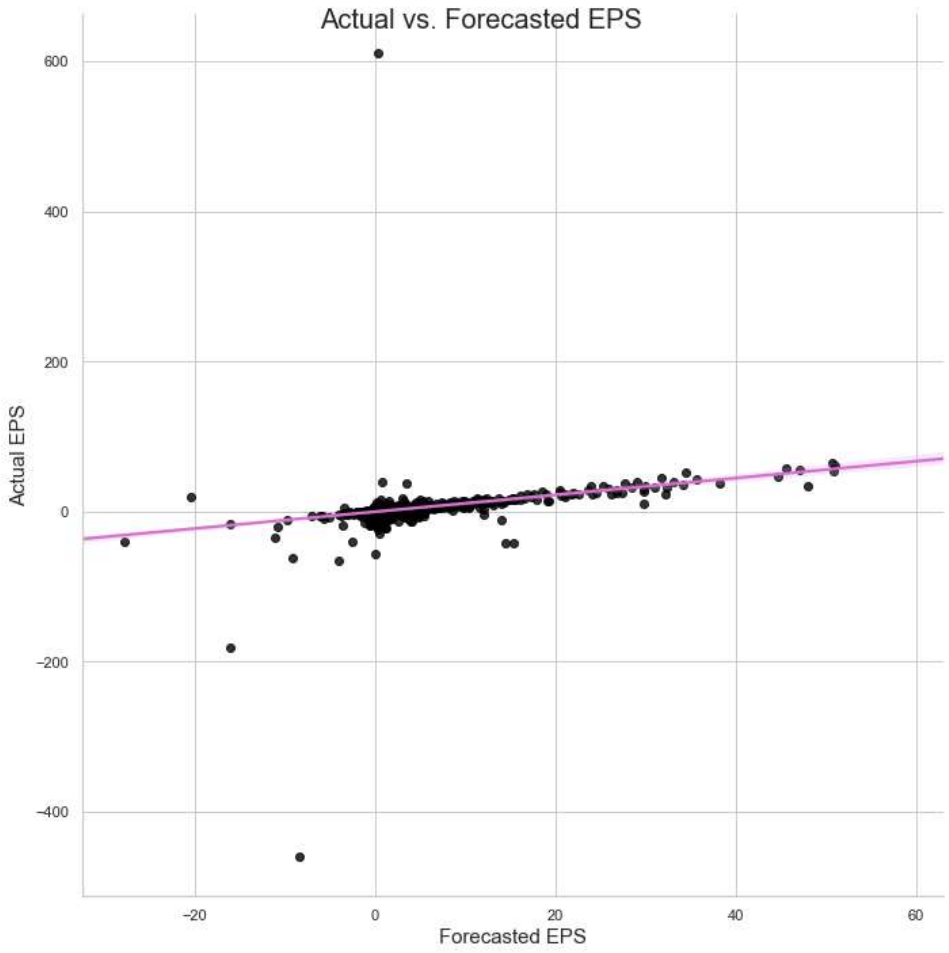

Figure 49

The raw actual vs. forecasted EPS scatterplot above helps paint a better overall picture of forecasting trends. The previous graphs depicting *averages* instead of *raw data* all contain clustered points at the beginning of the x-axis, but the above scatterplots start with 4 outliers until the first "cluster" of points takes place at around x = 0. From x = 0 onward, the initial cluster of points gradually declusters afterwards, all while maintaining minimal distance from the regression line as residuals.

Additionally, the slope for raw EPS data is less steep than all of the three previous graphs, and this relationship depicts a slight positive linear relationship: the ***weakest linear trend out of them all.*** Also, the 95% confidence interval is much narrower, and virtually non-existent,

Out of all the regression plots depicting actual vs. forecasted EPS, the raw data above displays the least amount of variance. Similarly, this could also mean the lowest amount of bias as well.

Summary

Interestingly enough, the graph encompassing the broadest data (the *raw data*) has the most stable 95% confidence interval as depicted by the regression line. In contrast, the twenty-year and quarterly average regression lines are the most unstable, with only outliers defining the trend after the first "cluster of points," and wide 95% confidence intervals relative to all other graphs.

Naive Forecasts

In addition to forecasted EPS, I have decided to implement my own method of forecasting: the ***naive forecast.***

Here is a breakdown of the steps I take to generate a new column called: ***naive_prediction***, which contains all of my own calculated forecasts.

1. Create a DataFrame depicting *eps_act* last quarter vs *eps_act* this quarter. That is, I pair every *eps_act* value with its previous value.

2. To calculate my naive forecast: take the ***average*** of the previous 2 *eps_act* values. This means all my naive forecasts are ***2-quarter moving averages of actual EPS.***

3. Compare my naive forecasts to the Bloomberg experts' forecasts.

Cross-Validation Testing

To compare my naive forecasts to Bloomberg forecasts, I decided to find the RMSE of each firm. The RMSE stands for the root mean square error, and 2 values are calculated for each firm: the naive RMSE, and the Bloomberg RMSE.

$$RMSE = \sqrt{\frac{\sum((predicted - actual)^2}{N}}$$

Top 10 Most Inaccurate Firms by RMSE

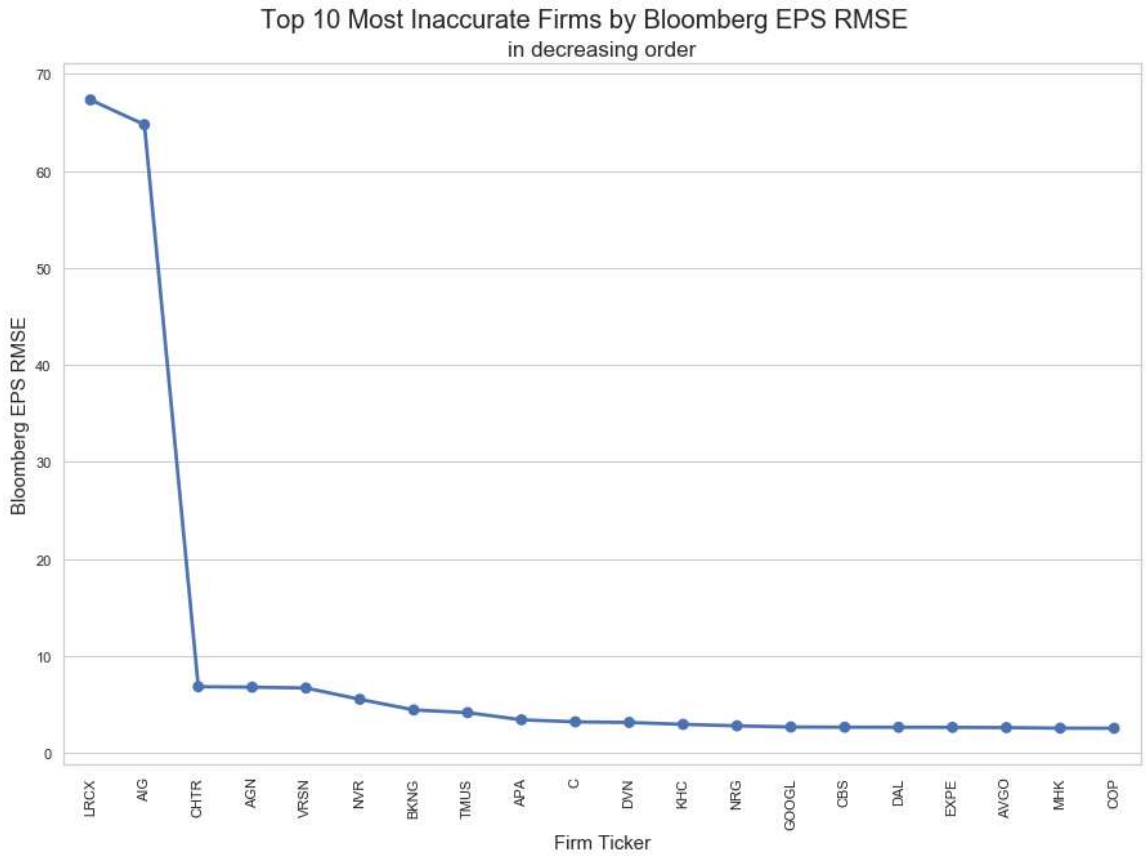

Figure 50

The 2 most notable outlier firms are LCRX and AIG, with values at approximately 68 and 65, respectively. After those 2 firms, RMSE values for the rest of the most inaccurate firms (in descending order) follow a more stable pattern: a decreasing negative linear trend approaching 0.

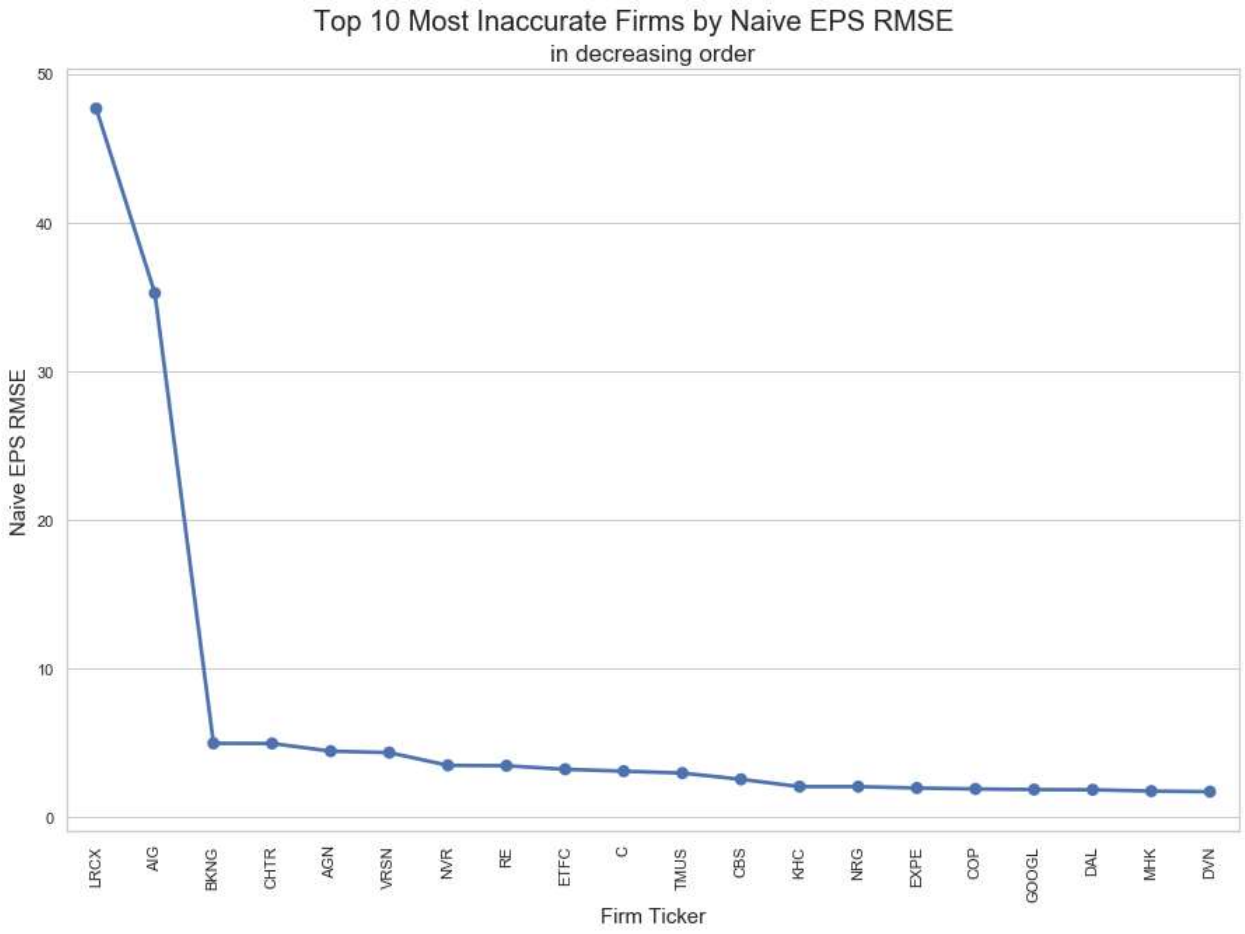

Figure 51

The 2 most notable outlier firm tickers are LCRX and AIG, with values are approximately 48 and 36, respectively. These firm tickers for naive EPS RMSE are consistent with the outlying firm tickers for the Bloomberg EPS RMSE. After those 2 firms, the RMSE values follow a more stable pattern, with a decreasing negative linear trend approaching 0.

Out of all 10 isolated firms, 4 firms are not shared between the top 10 most inaccurate firm tickers by decreasing RMSE values. The firm tickers not shared between those two graphs are APA, AVGO, ETFC, and RE.

Figure 52

Out of the 20 randomly selected firm tickers, WM, VLO, TDG, MNST, LEG, HON, GD, EXR, DISCK, DAL, CMI, CELG, CDN, ARNC, AMAT, ADSK, and A have Bloomberg RMSE values that are ***greater*** than my naive RMSE. Those are 17 firm tickers in total, where Bloomberg forecasts were a worse fit than my naive forecasts.

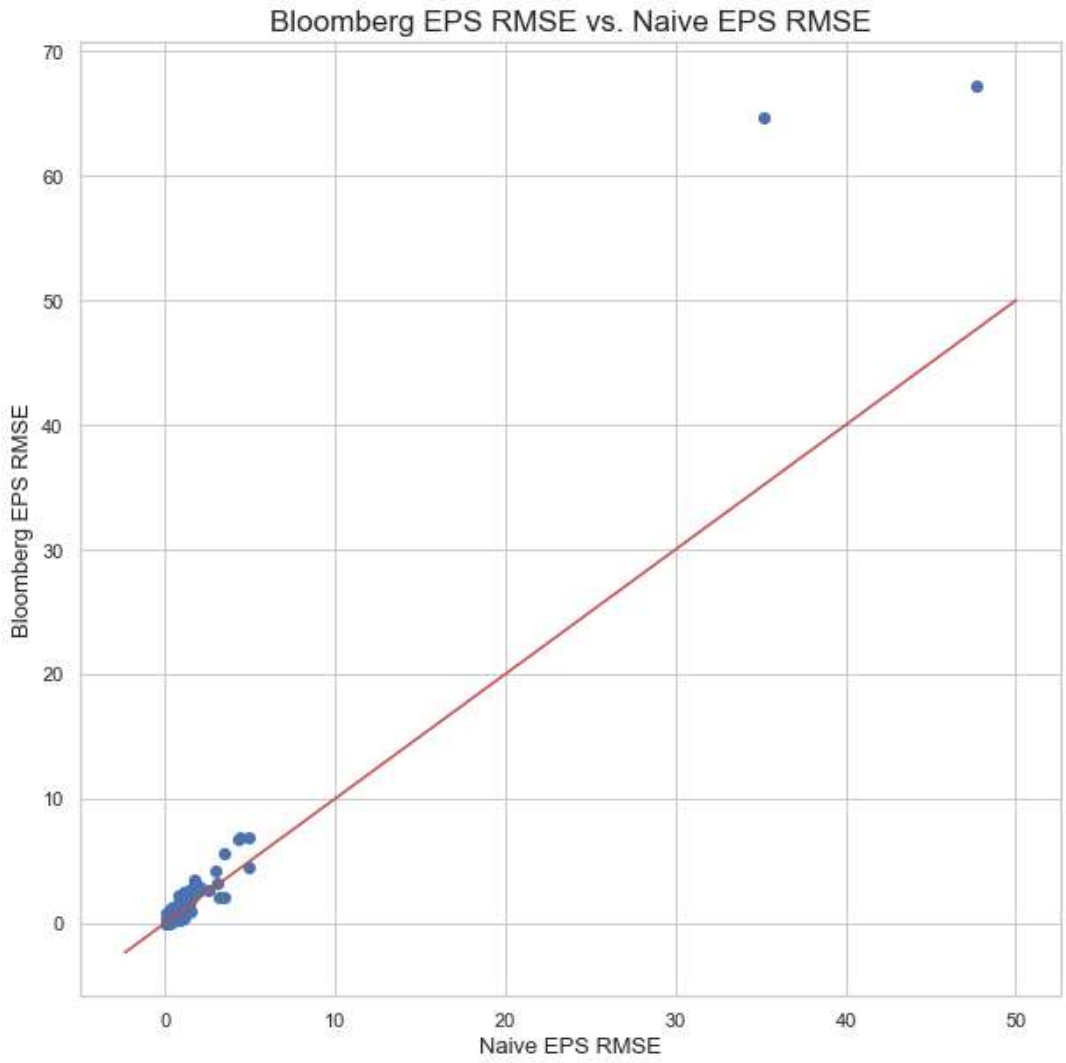

Figure 53

All points above the red y = x line represent Bloomberg EPS RMSE that is greater than my naive EPS RMSE.

For the 2 outlying points (which represent the firms LCRX and AIG), their Bloomberg RMSE is greater than my naive RMSE. This means that my naive EPS forecasts were a better fit for these 2 firm tickers than the Bloomberg forecasts.

Even when trying to ignore outliers, it is still challenging to draw a conclusion on the ratio of points above the y = x line vs. below. Let us ignore the outliers and zero in on the bulk of the data like below:

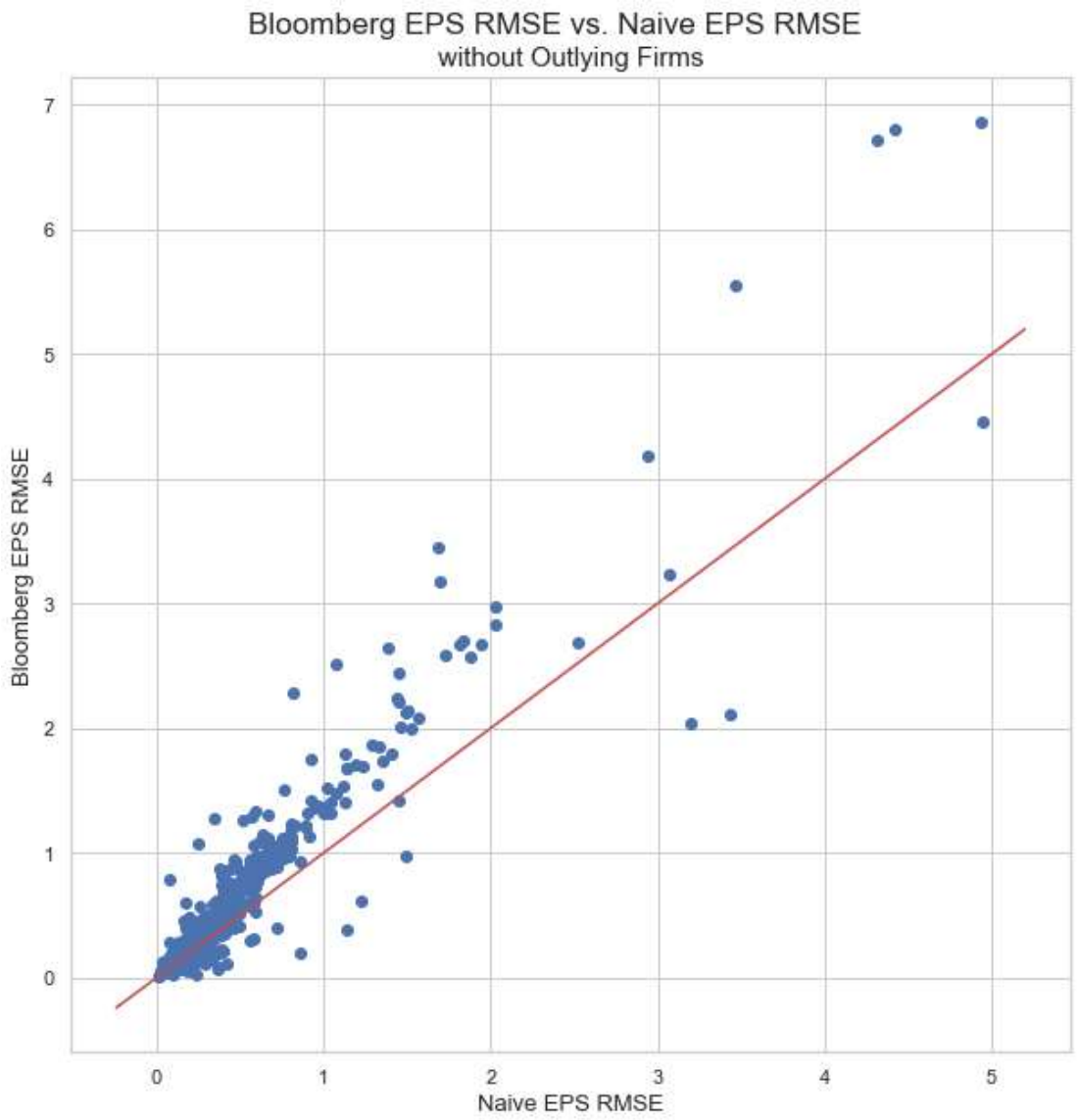

Figure 54

My initial observation was correct: there are, in fact, ***more Bloomberg EPS RMSE values that are greater than my naive EPS RMSE values*** per firm. This means that for most of the data, my naive forecasts were much more reliable than the Bloomberg EPS forecasts.

Of all RMSE values, I manually calculated that **86.95%** of my naive EPS RMSE values are less than the Bloomberg EPS RMSE values. For each firm, I subtracted their naive RMSE from their respective Bloomberg RMSE. Any difference above 0 indicates that the Bloomberg RMSE was much larger, and therefore, a worse fit. This percentage is consistent with the pattern produced in the above visual.

E) Multivariate Exploration

Prediction Errors (Naive vs. Bloomberg Forecasts)

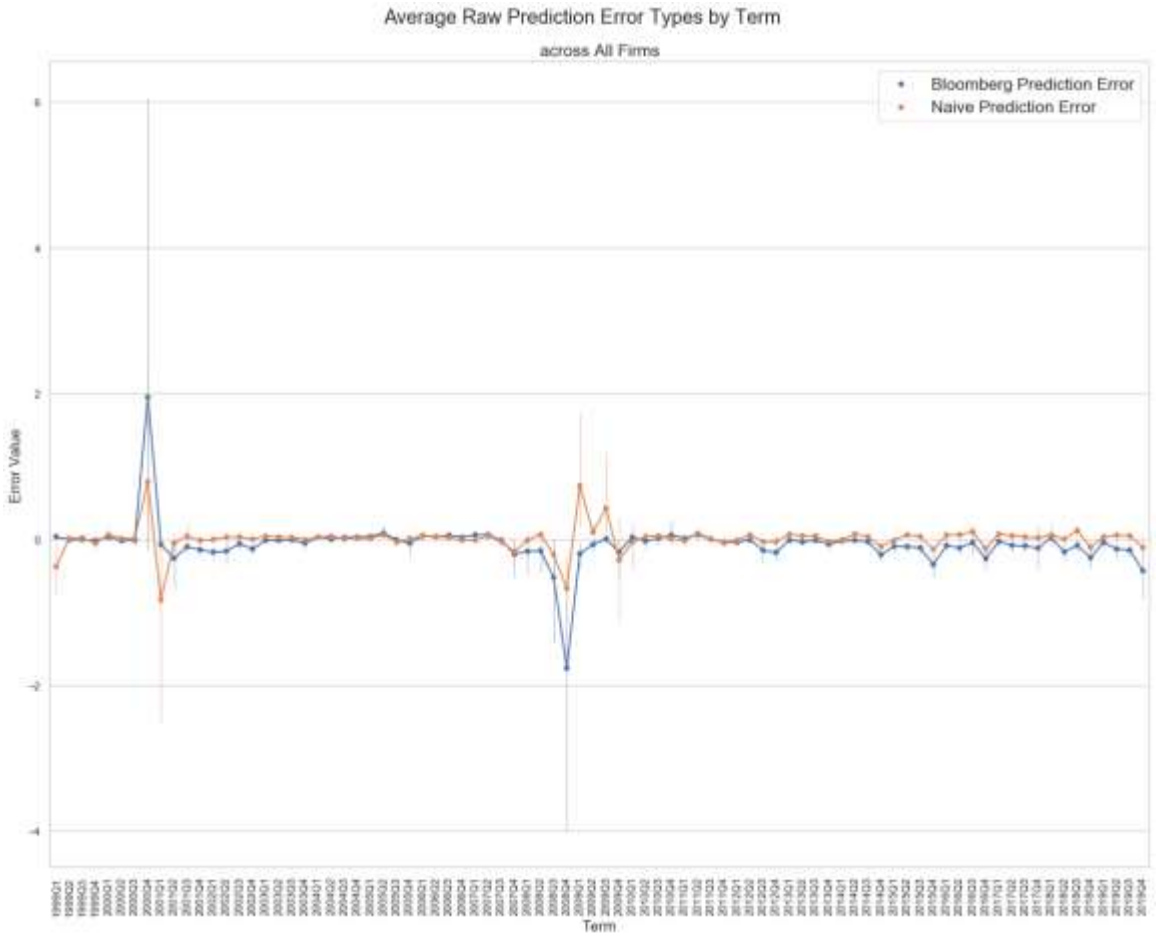

Figure 55

There are 2 notable terms: 2008Q4 and 2000Q4. Bloomberg forecasters were overly optimistic in 2008Q4, while overly pessimistic in 2000Q4, on average. Additionally, there are 4 terms where my naive prediction errors stood out: 2000Q4, 2001Q1 and 2008Q4, 2009Q1. My naive forecasts were overly optimistic in 2001Q1 and 2008Q4 (the latter being consistent with the Bloomberg prediction errors). My naive forecasts were overly

pessimistic in 2000Q4 and 2001 (the former being consistent with the Bloomberg prediction errors).

From 2012Q3 onwards, my average naive forecasts stay closer to 0 while the Bloomberg forecasts tend to be lesser than the naive forecasts: veering *away* from 0. This means that in the more recent years since 2012, Bloomberg EPS forecasts tended to be less accurate than my naive forecasts.

Both average naive forecasts and Bloomberg forecasts roughly "equalize" around 0 from 2003Q4 to 2007Q4, where they follow the same trend "clustering" around 0. Also, when observing 2000Q4 and 2001Q4, my naive forecasts tend to be overly pessimistic in the former term, but become optimistic immediately in the quarter afterwards: completely reversing this pattern. This pattern is also seen for the terms 2008Q4 and 2009Q1, where my naive forecasts tend to be overly optimistic, then become overly pessimistic immediately after entering the new year.

Special note should be taken that my naive forecasts spiked in pessimism in 2009Q1: *right* before the 2009 Recession began to recover. It is also curious that Bloomberg forecasters tended to be overly optimistic in their EPS during the harder years of the bear market.

Figure 56

The most notable years are consistent with the previous graph, *Figure 55*. Bloomberg forecasters tended to be overly optimistic in 2008, while overly pessimistic in 2000. In both cases, my average naive forecasts tended to be a better fit than the Bloomberg EPS forecasts, since they all tend to cluster around 0 while the Bloomberg ones veer *far* away from 0.

Overall, my naive forecasts tend to be more accurate and better fitting than Bloomberg forecasts across all years. The only years where Bloomberg EPS forecasts are depicted to

be a better fit are in the years 1999, 2001, 2006, and 2007, where my naive forecasts veer farther away from 0.

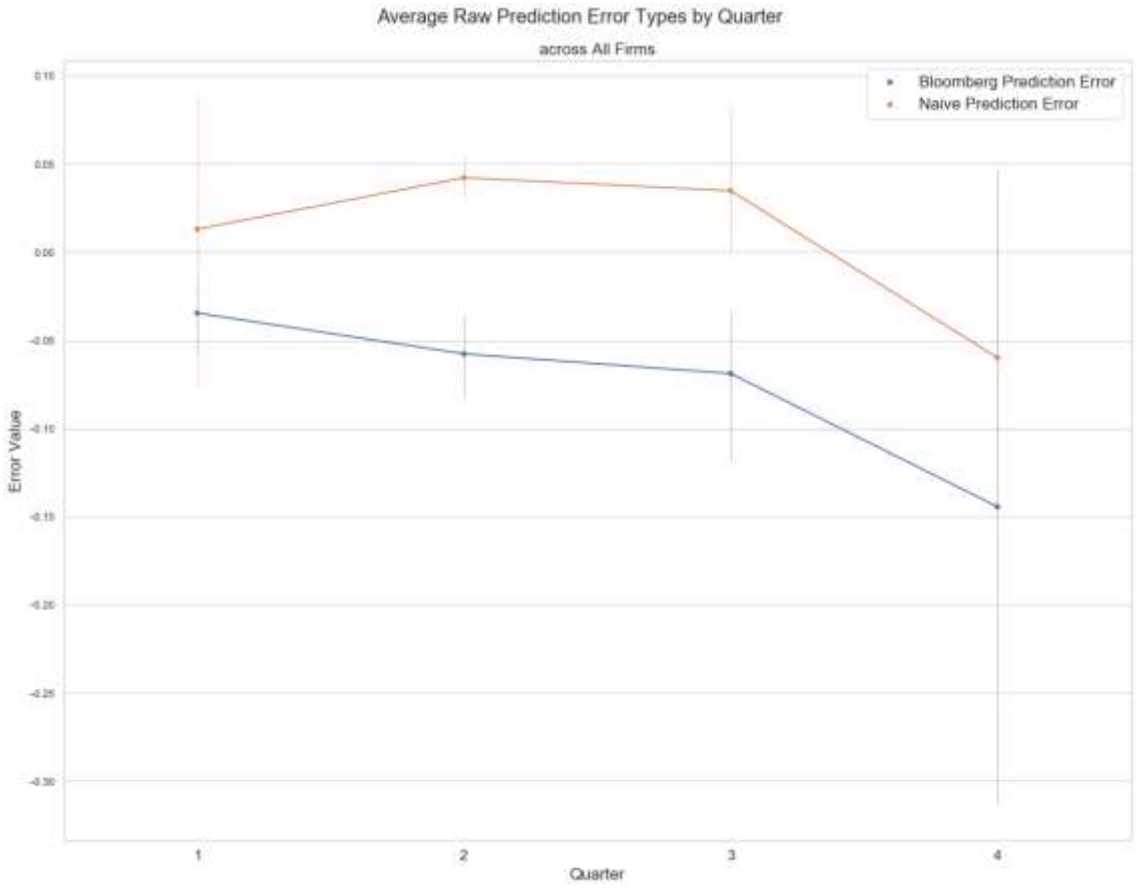

Figure 57

Across all quarters, my naive forecasts have *always* tended to be a better fit than the Bloomberg EPS forecasts. No matter the current quarter, my naive forecast, on average, has been proven to *always* be more reliable. This is another key takeaway. Bloomberg EPS forecasts tend to be more optimistic in Q2 of any year, while my naive forecasts tend to be more pessimistic in Q2 of any year. Similarly, in Q4 across all years, *both* forecast types tended to be more optimistic. This makes intuitive sense: I would assume that as familiarity with the trends of the current year builds, the more confident forecasters would be in their

EPS predictions. These results are consistent with Figure *55*, where Bloomberg forecasts have tended to be overly optimistic during 2000Q4 and 2008Q4: both outliers occur during the same quarter. It would be an interesting approach to examine any outliers that occur during this quarter and examine if the above quarterly trend would fare differently without those outliers.

More Raw Prediction Errors

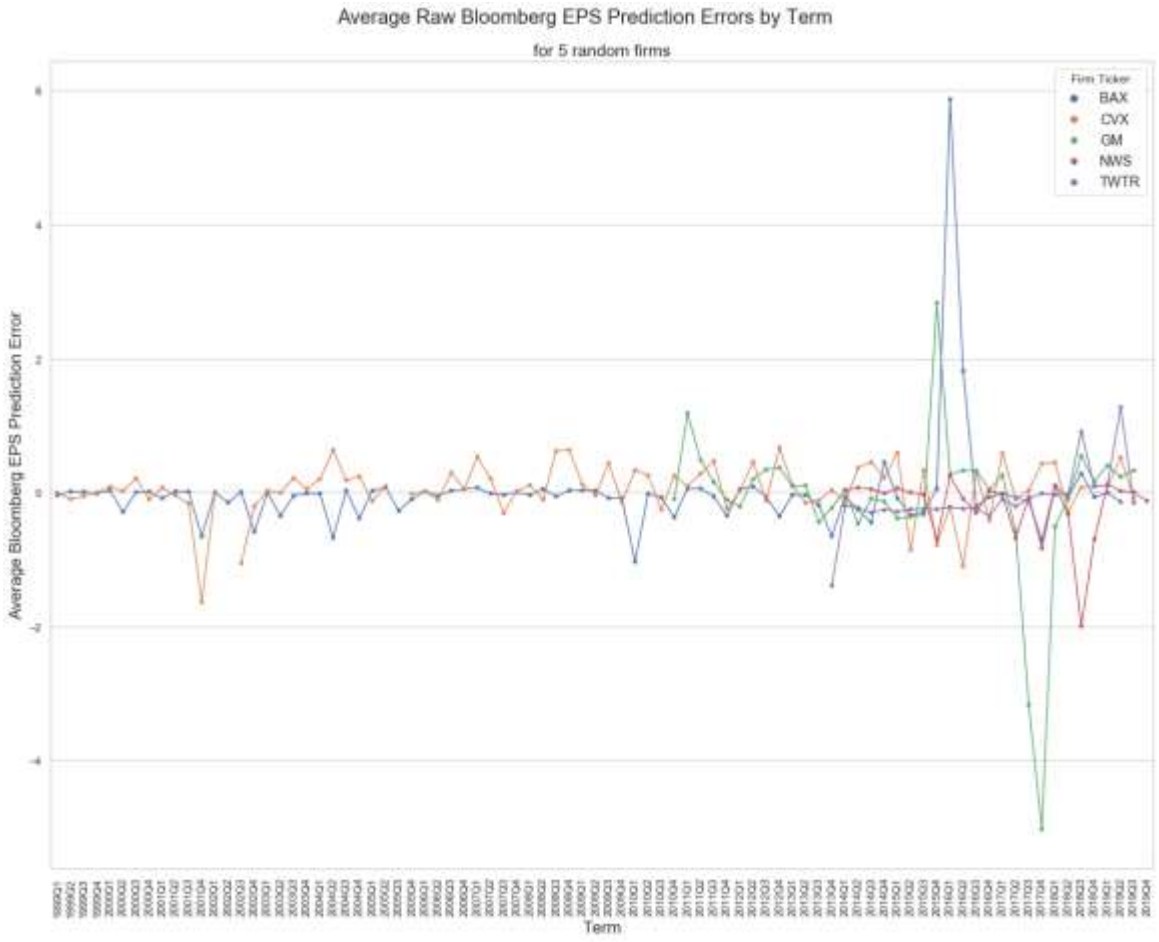

Figure 58

For the above 5 random firms, the average Bloomberg EPS prediction error trends are scattered, inconsistent, and erratic. Here, no outliers appear during 2000Q4 or 2008Q4.

This is a key takeaway: a few outlying prediction errors, in fact, *do* affect the overall trends in those 2 terms. Additionally, one notable outlier occurs at 2017Q4, where 3 random firms consistently show an average overly optimistic EPS forecast from Bloomberg: GM, NWS, and TWTR. Another notable outlier occurs at 2016Q1, where BAX is an outlying firm showcasing an average overly pessimistic EPS forecast from Bloomberg.

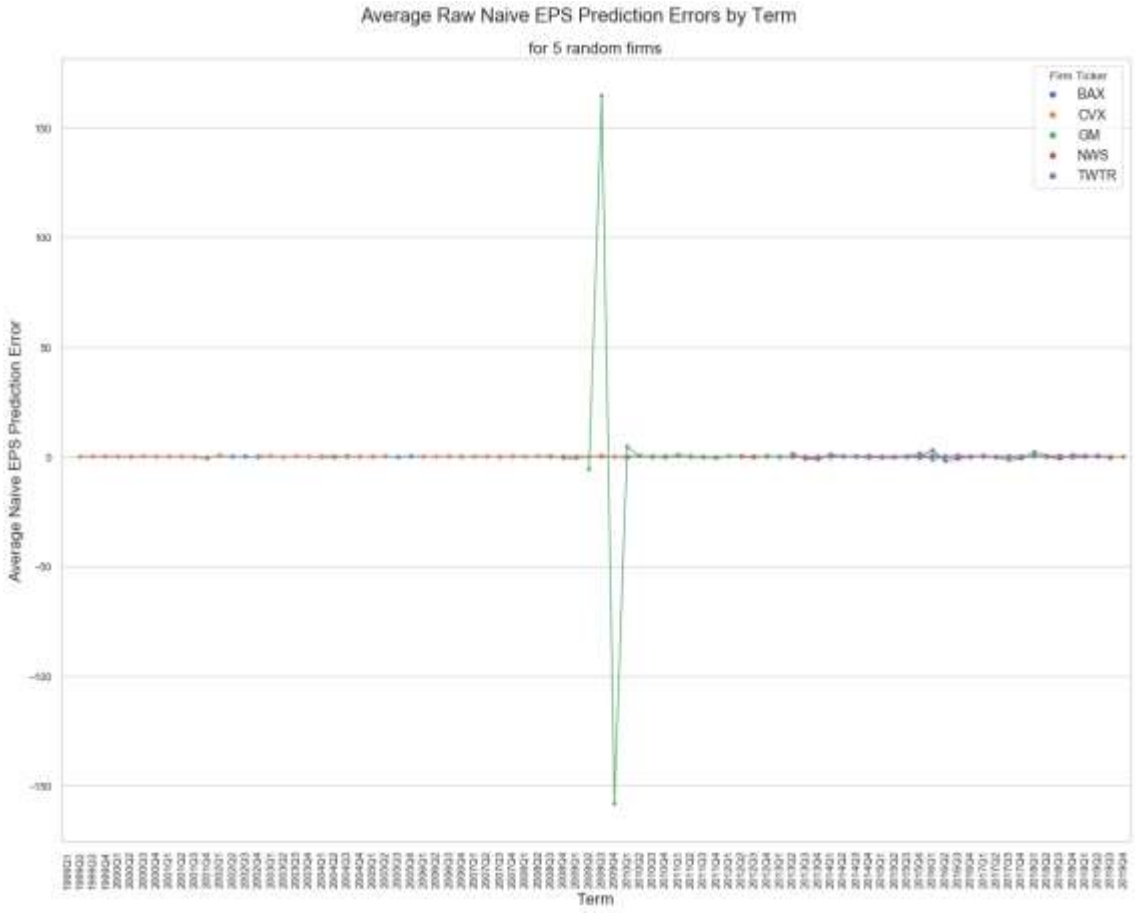

Figure 59

Unlike the previous *Figure 58*, my naive forecasts depicting the same 5 random firms are much more consistent and better fitting, since all average prediction errors tend to cluster around 0. Also, there are only 2 notable outliers among those naive forecasts: at 2009Q4 and 2009Q2. My naive forecasts tended to be overly optimistic for the former term, and

overly pessimistic for the latter; these 2 outliers represent 1 firm: GM. And as it turns out, my naive EPS forecast was a much better fit for the GM firm.

A notable detail: *Figure 59* depicts a much larger scale on the y-axis: *Figure 58* depicting average Bloomberg forecasts ranges from -6 to 6, while *Figure 59* depicting average naive forecasts ranges from -150 to 170 on the y-axis. This means that for the 5 random firms selected, my naive forecasts tended to be a better fit. But whenever an outlying firm appears, those outlying values stray *further* from 0, which means my naive forecasts were a worse fit. In summary, my naive forecasts are usually more reliable than the Bloomberg forecasts, *only* when outliers are not present.

Percentage Error vs. Term Period by Top 5 Most Inaccurate Firms

Percentage Error by Year

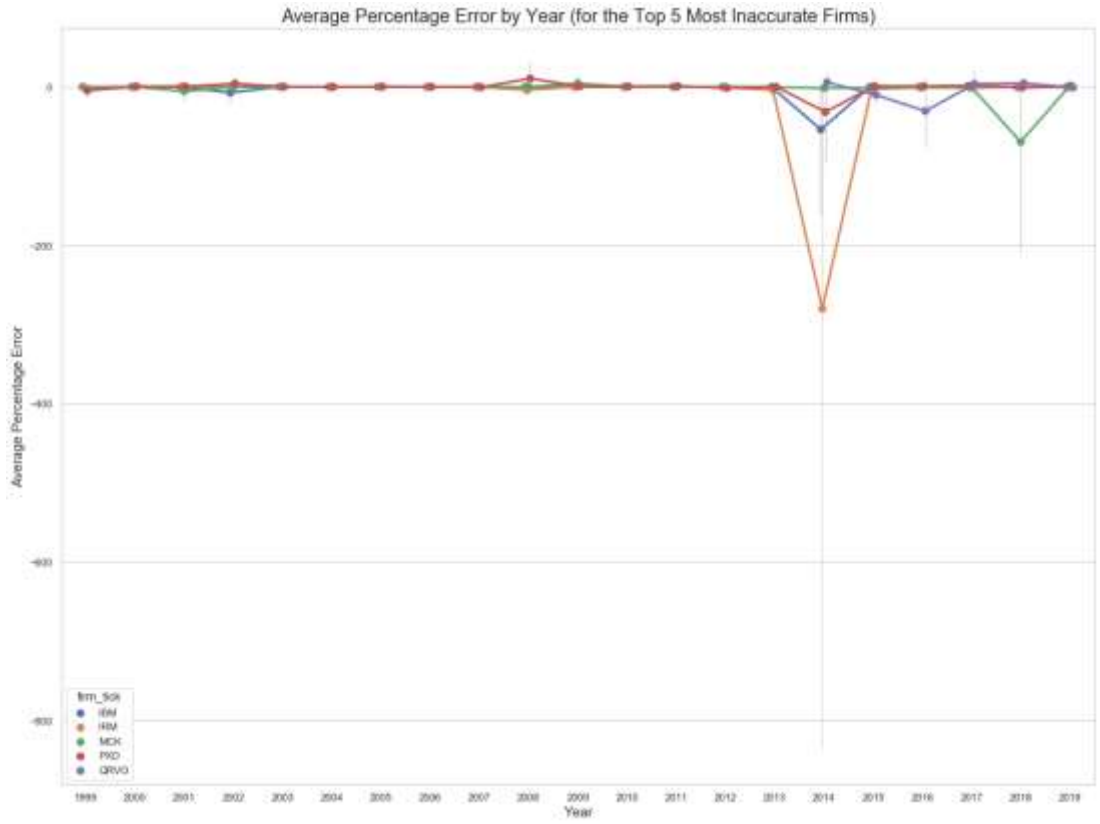

Figure 60

The above visual is not great for analysis. Let us look at a strip plot distribution instead:

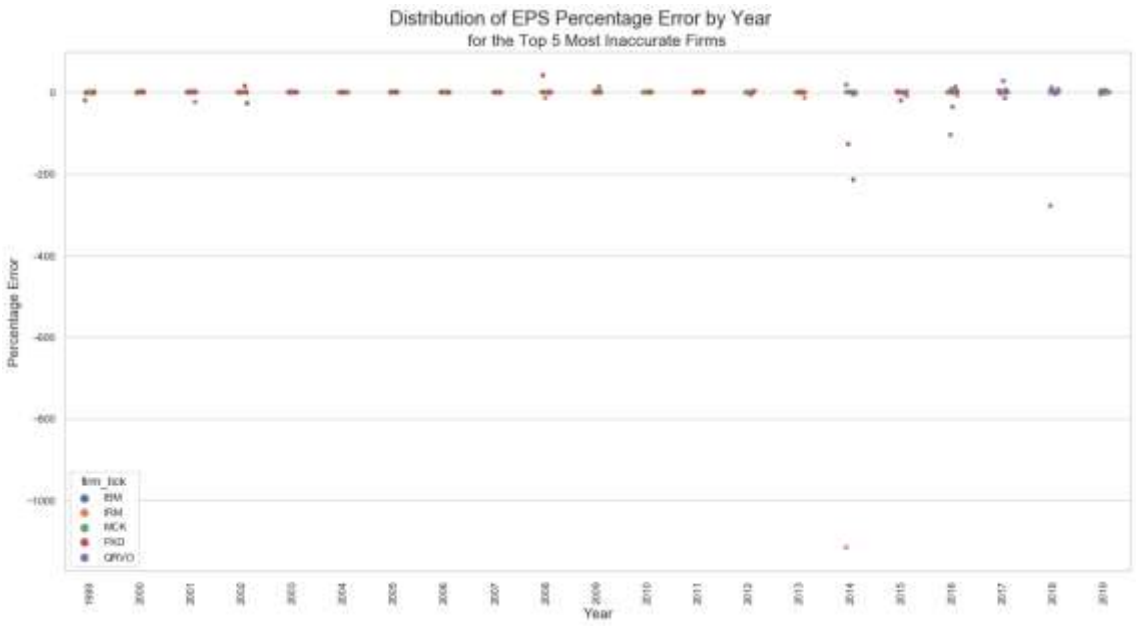

Figure 61

The most notable outlier above is IRM, which has a single EPS percentage error of over -1000 in the year 2014. BIM, IRM, MCK, PXD, and QRVO are the five most inaccurately predicted firms in percentage error by *year.* All other firms besides IRM in the years 2014 - 2018 share large distance gaps from IRM in 2014.

Outliers only started to appear in distributions starting in the year 2014. Therefore, ***EPS forecasts have become more inaccurate in the more recent years, starting from 2014***.

Percentage Error by Quarter

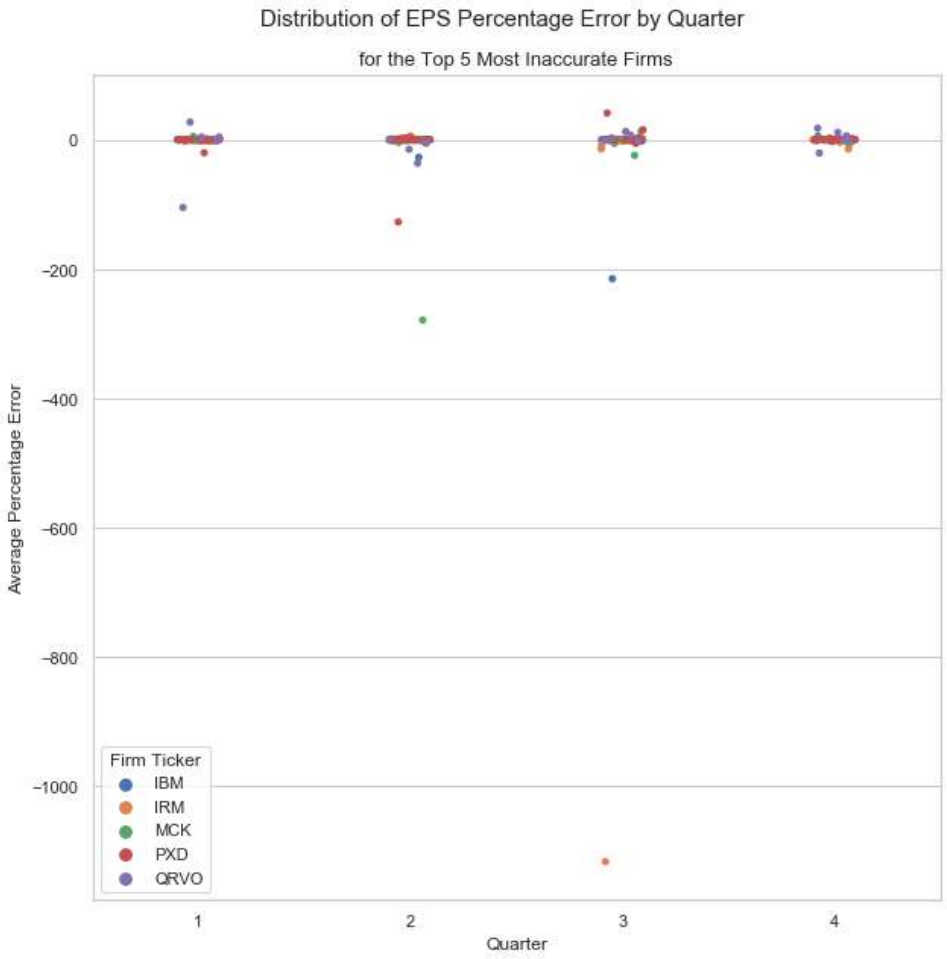

Figure 62

The most notable outlier is IRM yet again, with a percentage error of over -1000 for EPS percentage error in Quarter 3. IBM, IRM, MCK, PXD, and QRVO are the five most inaccurately predicted firms in terms of percentage error by *quarter.* This list is the same as the list taken from the legend in the previous stripplot and pointplot depicting *years* instead.

Quarter 4 is also the only quarter with no outliers. This makes intuitive sense because as the quarters in any singular year pass and familiarity with the current fiscal year builds, the more accurate forecasts would come to be. However, this is only a theory; let us verify this through the following pointplot:

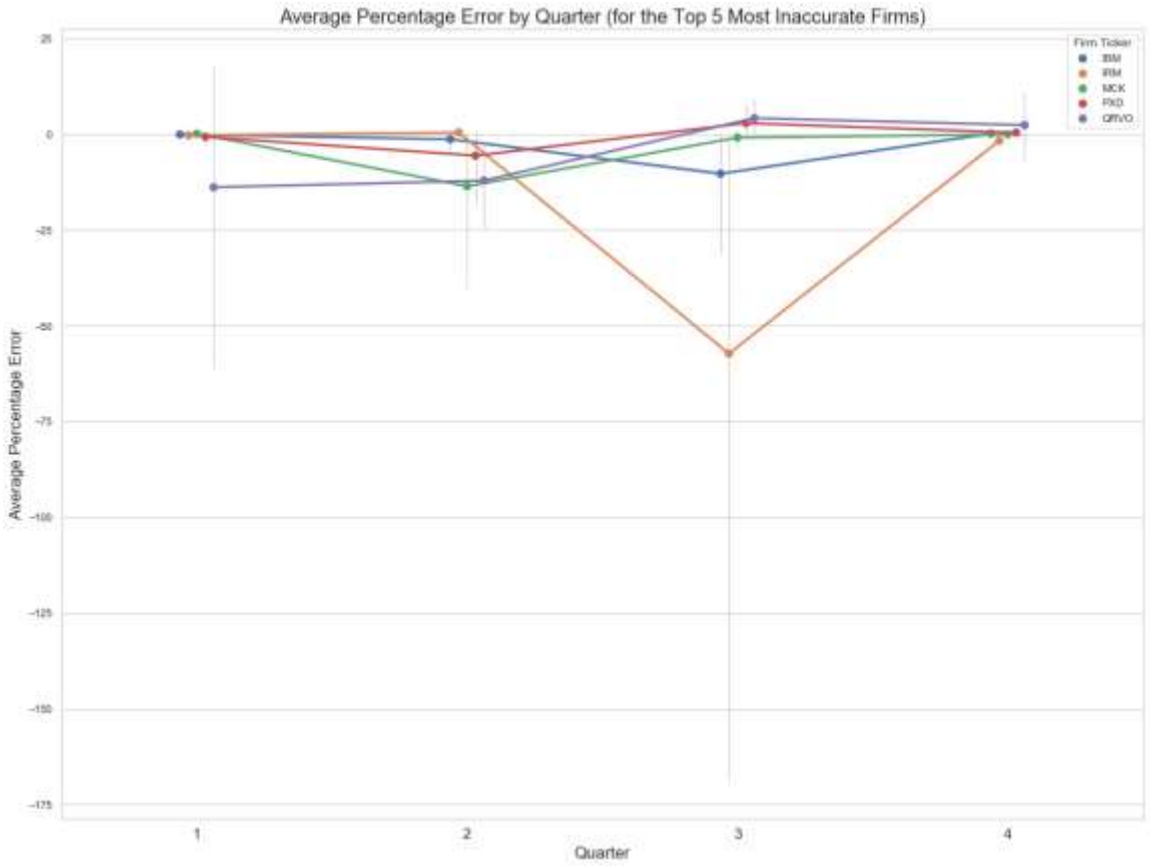

Figure 63

The firms IRM and IBM deviatate greatly during Quarter 3, where forecasters' predictions veer away from zero. Meanwhile, the other firms cluster more closely around an average percentage error of 0. Therefore, ***the most inaccurate predictions, on average, happen to be made during Quarter 3 of any given year.*** This observation ***disproves*** my initial intuition that forecasters' predictions would become more accurate by the quarter.

Percentage Error by Year and Quarter

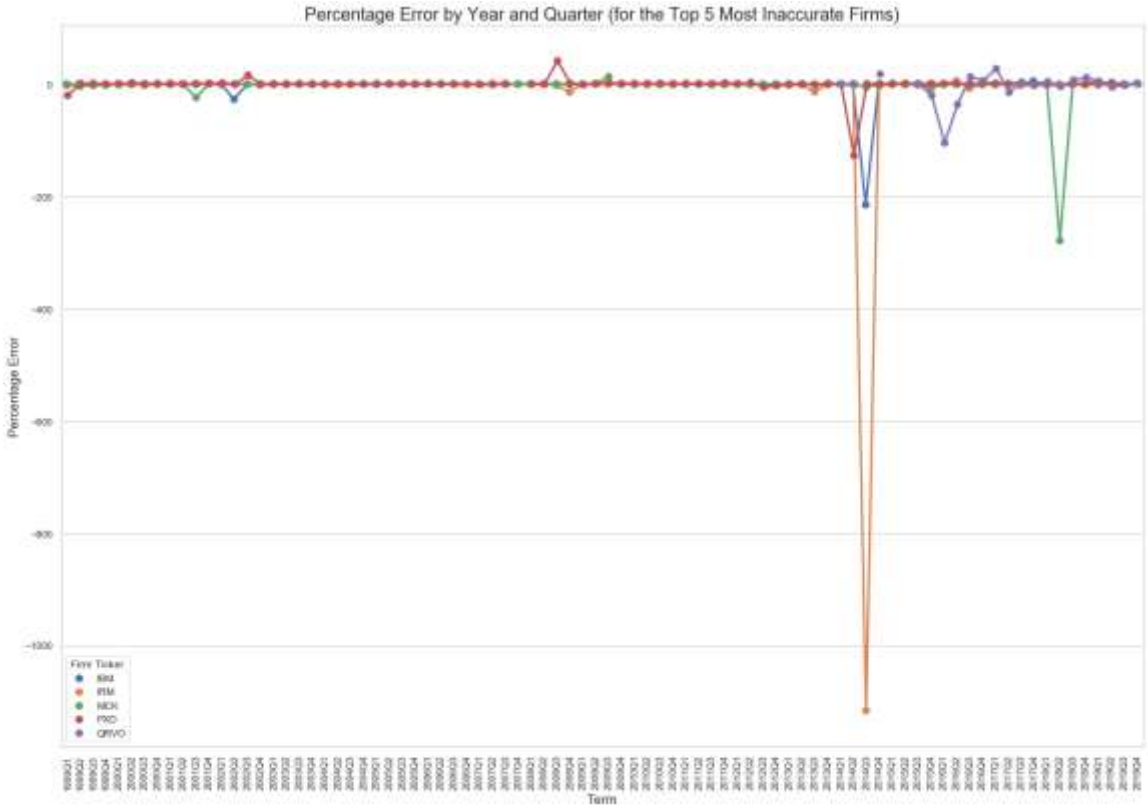

Figure 64

This isn't a very good visualization either; let us look at a stripplot instead.

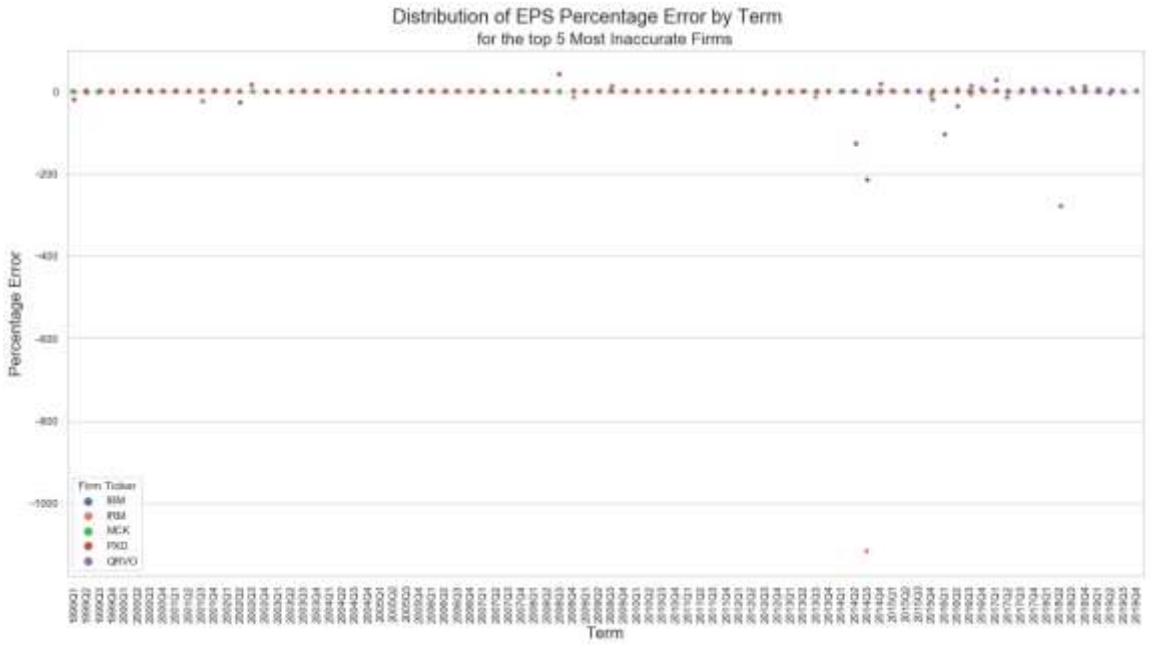

Figure 65

The most notable outlier is IRM yet again, with the same percentage error of over -1000 in the term 2014Q3. BIM, IRM, MCK, PXD, and QRVO are the five most inaccurately predicted firms in terms of percentage error by term: the ***same list as all the previous plots*** featuring standalone *quarters* and *years.* It is very convenient that the list of the most inaccurate firms has stayed the same throughout all the above investigations.

Outliers begin appearing only after 2014Q3. In the terms afterward, outliers tend to year in Q1 and Q3 exclusively, but much more Q3. All other firms besides IRM share large distance gaps from 2014Q3 - 2018Q3 from IRM in 2014Q3. And since outliers only started showing up in distributions starting with the year 2014, this means that ***EPS forecasts have become more inaccurate in the more recent terms, starting from 2014Q3.*** This observation is consistent with the ones made in the quarterly and yearly stripplots and pointplots.

Prediction Error vs. Term Period by Top 5 Most Inaccurate Firms

Prediction Error by Year and Quarter

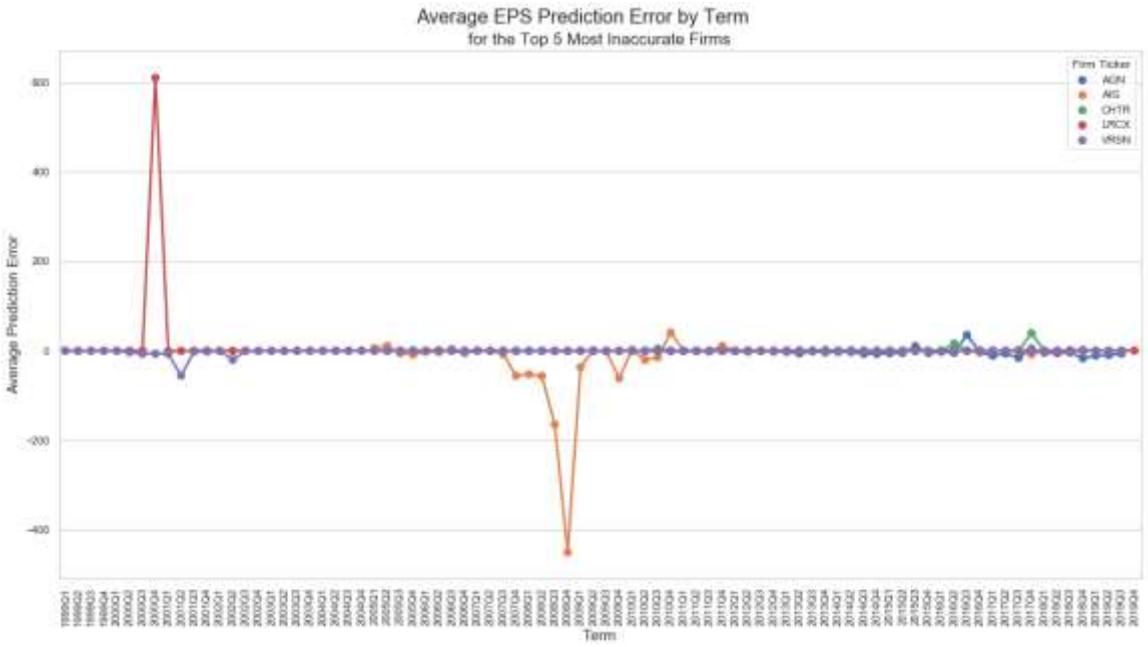

Figure 66

Prediction Error by Year

The 2 most noticeable outliers are the firm AIG in 2008Q4 (***optimistic*** forecasts), and LRCX in 2000Q4 (***pessimistic*** forecasts). The list of the 5 most inaccurately predicted firms in terms of ***prediction error*** are AGN, AIG, CHTR, LRCX, and VRSN. Out of the 5, VRSN is the most stable firm among the top 5 most inaccurate firms; its average prediction error hugs most closely to a slopeless y = 0 trend.

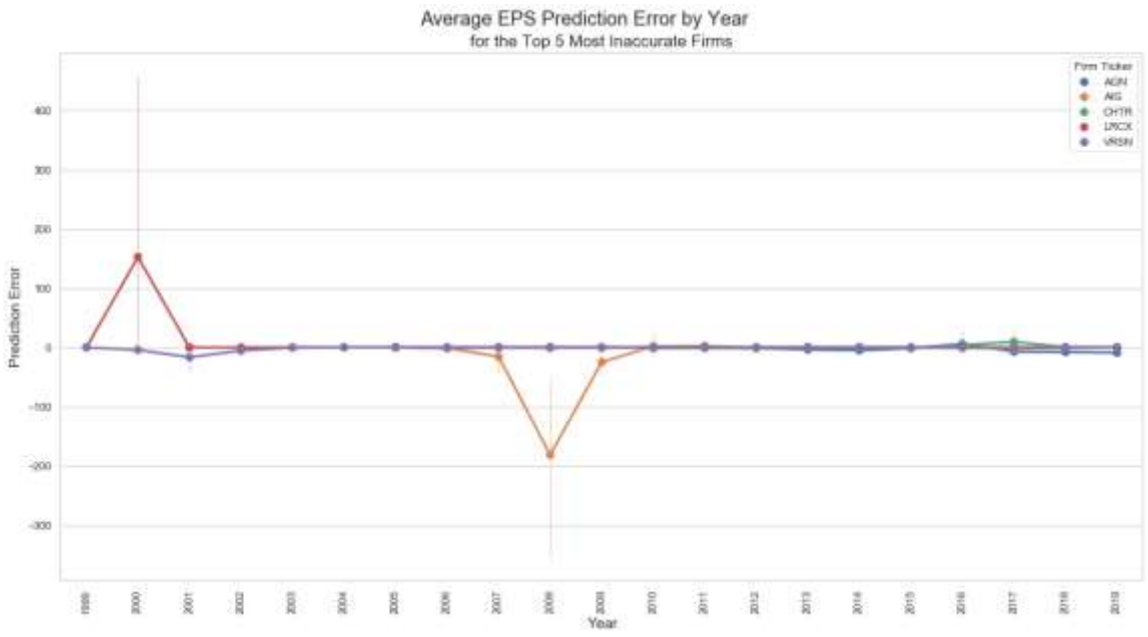

Figure 67

Yet again, the 2 most noticeable outliers are the firm AIG in 2008 (***optimistic***) forecasts, and LRCX in 2000 (***pessimistic*** forecasts). The list of the 5 most inaccurate firms by yearly average prediction error are AGN, AIG, CHTR, LRCX, and VRSN. Of the five, VRSN is yet again the most stable of the top 5 most inaccurate firms; it contains no outliers. Data for the firm CHTR shows up only in 2017.

Prediction Error by Quarter

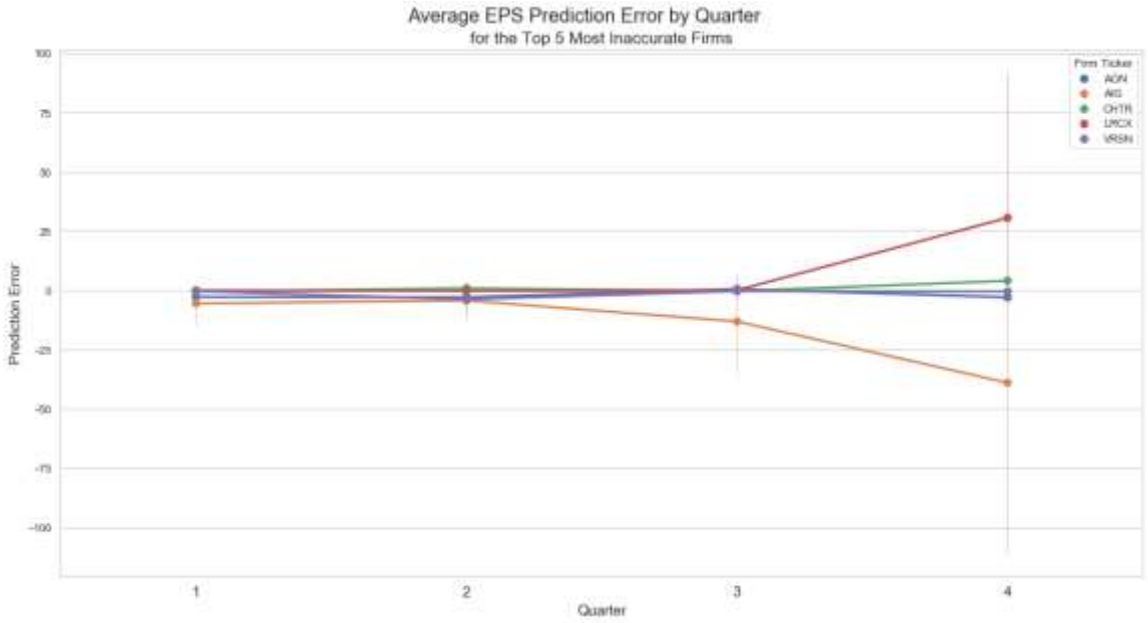

Figure 68

One most noticeable outlier is AIG, which is, on average, the ***only outlier*** in Q3 of any given year. The top 5 most inaccurate firms by quarterly prediction error are AGN, AIG, CHTR, LRCX, and VRSN. This is the same list as generated through the previous yearly and term basis.

There 5 firms display relatively low prediction errors until Q4, where both LRCX and AIG "branch off" into opposite directions. This means that ***forecasters, on average, are likely to make inaccurate EPS forecasts in Q4 of any given year.***

As expected, VRSN stays yet again as the most stable of the 5 firms; it contains no outliers across all 4 quarters. Data for the firm CHTR shows up only in 2017.

After conducting my entire data exploratory stage in Jupyter Notebook, I converted the entire exploration process into an HTML file:

```
#convert notebook to HTML
from subprocess import call
call(['python', '-m', 'nbconvert', 'data_exploratory.ipynb'])
```

Conclusions

The data exploration stage has helped me generate many interesting visuals, as can be seen with all figures created in the univariate, bivariate, and multivariate stages. Each question is presented below, along with all key takeaways and discussions that are relevant to the current question.

A) Research Questions

Question 1: *Does average EPS prediction error depict any differences in trends whether by a yearly, quarterly, or full-term basis?*

As shown in *Figure 32*, forecasters were most optimistic in 2008Q4, and most pessimist in 2000Q4. Likewise, *Figure 33* shows that forecasters were most pessimistic in the year 2000 and most optimistic in the year 2008: results congruent with the trend depicted in *Figure 32*. Coincidentally, the years 2000 and 2008 display the widest variances among average prediction errors: values in the year 2000 ranging from 0 to 0.5, and -0.25 to -1.5 for 2008.

The 2 years where forecasters made seemingly impulsive overly pessimistic/optimistic predictions, 2000 and 2008, also coincidentally contain the highest variance. Therefore, all outlier prediction errors are very spread out from the mean average prediction error. The

stock market crashed in 2008[28] as part of the Great Recession, so I find it *very* curious that optimism among Bloomberg forecasters skyrocketed in Q4 of that year.

As seen in *Figure 34*, the later the quarter, the more optimistic forecasters become in their predictions. This intuitively makes sense; I conjecture that as the familiarity of the year increases and clearer patterns become established, the more forecasters would become confident in their EPS predictions. Additionally, there is much more variance among quarterly EPS prediction errors than by a yearly or termly basis.

Nonetheless, the overall trend between the first 2 time-series plots, *Figures 32* and *33*, depicting average EPS prediction error by quarter and year is consistent. When ignoring the pessimistic and optimistic outliers, there is no slope; all average EPS forecasts fluctuate around 0.00 as the years go by. It is only by a quarterly basis that we see a predictable pattern emerge, where outliers (usually optimistic) happen exclusively during Q4.

Therefore, when comparing the quarterly trends to the full-term prediction error trends, indeed, the only 2 outliers occur in Q4 of 2000 and 2008. This means that the yearly and quarterly figures, *Figures 33* and *34*, display congruent trends & patterns with *Figure 32*, which depicts full terms (YYYYQQ) as the independent variable.

Question 2: *I generate naive EPS forecasts by calculating the rolling mean of the 2 actual EPS values from the past 2 quarters. How do my EPS forecasts compare to Bloomberg's EPS forecasts?*

[28] Amadeo, Kimberly. "The Great Recession of 2008 Explained With Dates." The Balance, Mar 20, 2020.

My naive forecasts tended to be overly optimistic in 2001Q1 and 2000Q4, the latter being consistent with the Bloomberg forecasts. On the flipside, my naive forecasts tended to be overly pessimistic in 2000Q4 and 2001Q1. From 2012Q3 onwards, my average naive prediction errors stayed closer to 0 than the average Bloomberg prediction errors do, which means my naive EPS predictions tended to be a better fit from 2012Q3 onwards.

When looking at 2000Q4 and 2001Q4, my naive forecasts tend to be *overly pessimistic,* then turn *overly optimistic* in the quarter immediately after. This pattern is reversed for the years 2008Q4 and 2009Q1: my naive forecasts tend to be overly optimistic, then become overly pessimistic in the quarter immediately afterward. These 2 observations imply that my average naive prediction errors tend to change "mood" when entering a new year from 2000 to 2001 and 2008 to 2009.

Overall, my naive forecasts tend to be more accurate and better-fitting than Bloomberg forecasts across all years, as depicted in *Figure 55.* When examining these errors on a quarterly basis, however, my naive forecasts have *always* tended to be a better fit than the Bloomberg EPS forecasts, as shown in *Figure 57.*

Question 3: *What differences and similarities emerge when analyzing the prediction error and percentage error of EPS forecasts?*

For prediction errors, Bloomberg forecasters, on average, are more likely to be inaccurate in their predictions during Q4 of any given year. This is depicted in *Figure 68,* where the top 5 most inaccurate firms by prediction error all display relatively low average prediction errors until Q4, where the firm tickers LRCX and AIG "branch off" into opposite directions on the x-axis.

For percentage errors, EPS forecasts have become more inaccurate in the more recent terms, starting from 2014Q3. Outliers only start to appear in the trend depicted through *Figure 68*, starting in the year 2014. This conclusion is consistent with the observations made in the quarterly and yearly stripplots: *Figures 66* and *67* for quarterly data, and *Figures 68* and *65* for yearly data.

Figure 66 shows the top 5 most inaccurately predicted firms in terms of absolute *prediction error*: AGN, AIG, CHTR, LRCX, and VRSN. Respectively, these firm tickers stand for Allergan plc, American International Group Inc, Charter Communications Inc, Lam Research Corporation, and Verisign. From the figure alone, the most notable outliers are AIG and LCRX—AIG sees a spike in optimistic forecasts in 2008Q4, while LCRX sees a spike in pessimistic forecasts in 2000Q4. *Figure 68* depicts that on average, the most inaccurate EPS forecasts emerge in Q4 of any given year.

On the other hand, *Figure 68 displays* the top 5 most inaccurately predicted firms in terms of absolute *percentage error:* IBM, IRM, MCK, PXD, and QRVO. These firm tickers stand for IBM Common Stock, Iron Mountain Inc, McKesson Corporation, Pioneer Natural Resources, and Qorvo Inc, respectively. The most notable outlier in *Figure 68* is IRM, which holds a soaring percentage error of over -1000 for EPS in 2014Q3. Outlying average percentage errors only started to appear after the year 2014.

It is also helpful to note that the prediction error charts are *not* consistent with the average percentage error charts. The top 5 most inaccurately predicted firms are very different, and this is to be expected. Percentage error tells you how "big" your errors are**,** while prediction errors simply tell you how "close" your prediction error was to the accepted value.[29] This explains why different firms are to be expected after ordering all absolute percentage/prediction errors in decreasing order.

Question 4: How do my naive RMSE values compare to Bloomberg RMSE when accounting for EPS forecasts? For what percentage of firms does my naive RMSE beat Bloomberg's RMSE?

In *Figure 53,* all points above the y = x line represent a Bloomberg EPS RMSE that is greater than my Naive EPS RMSE. There are approximately 505 RMSE values: 1 calculated for each firm. For the two outlying points in *Figure 53*, Bloomberg RMSE is greater than my naive RMSE, which means that my naive EPS forecasts were a better fit for these 2 firm tickers than the Bloomberg forecasts. Those 2 outlying points represent the firms LCRX and AIG, which are also the outliers found in *Figures 50* and *51.* However, it was hard to determine a rough percentage through visualization alone, so I decided to remove the outliers and zero-in on the bulk of the data instead.

[29] Stephanie. "Percent Error & Percent Difference: Definition & Examples." Statistics How To, Nov 8, 2016.

Figure 54 represents all Bloomberg EPS RMSE vs. naive EPS RMSE without the 2 outlying firms. In fact, this figure depicts that more Bloomberg EPS RMSE values are greater than my naive EPS RMSE values per firm. For the bulk majority of my data, my naive forecasts turned out to be much more reliable than the Bloomberg EPS forecasts. To check for the exact percentage, though, I subtracted each firm's naive RMSE from their respective Bloomberg RMSE. Any difference above 0 indicates that the Bloomberg RMSE was much larger. After performing this calculation, I discovered that 86.95% of all my naive EPS RMSE values are less than the Bloomberg EPS RMSE values.

Question 5: *How do EOD Prices trend across all firms from 1999 - 2019?*

EOD prices show a general positive trend from 2009 onward, as depicted in *Figure 30*.

EOD prices also see a "trough" from 2007 - 2009: the exact years of the Great Recession.

During the years of 2007 - 2009, a catastrophic bear market took hold: a period of prolonged falling prices ensued, encouraging investors to sell and short their stocks. As it turns out, the S&P 500 was in the midst of sell-offs of 57% from October 2007 to March 2009, from prior highs.[30] In 2008, the S&P was down around -38%. But in 2009, the market index made a speedy recovery, finishing at over 23%.[31] This quick recovery from a prolonged bear market is directly reflected through the EOD Price trends in *Figure 30*, where EOD prices consistently climb towards a positive linear trend after 2009.

[30] Jackson, Anna-Louise. "Here's How Quickly the Stock Market Recovered After Similar Downturns." Acorns, Feb 26, 2020.
[31] McKenna, Kristin. "What Happens to the Stock Market After a Recession?" Forbes, Apr 3, 2020.

All patterns depicted in the actual vs. forecasted EPS scatterplots, *Figures 46 - 49*, are consistent with their corresponding striplot distribution (*Figure 29)* at the beginning of the bivariate stage. Raw actual EPS contains the most outliers, and there are a few "spikes" in the Seaborn pointplots where actual EPS significantly veers away from the forecasted EPS trend line.

Overall, this generally positive linear trend shows that all 505 firms in the S&P 2019 Index have been getting "richer" over the past 20 years, where their EOD Prices have followed a steady linear increase since recovering from the bear market in 20009.

B) Limitations

After gathering, cleaning, and exploring all EPS and EOD data, I have isolated the following shortcomings of the data. These limitations directly impacted the quality of the data and restricted the amount of analysis I was able to conduct.

Overall, there was no explicit Bloomberg function for gathering forecasted EOD Prices by fiscal period.

There was also an incongruence between fiscal periods and calendar periods in the data: actual EPS and forecasted EPS were gathered by *fiscal period*, while actual EOD and forecasted EPS (3 months prior) were gathered by *calendar periods.* This was an ultimate mistake in the data gathering process, which explains why I did not touch upon *eps_fc_terms* at all during the data exploration stage.

The dataset for forecasted EPS 3 months prior was missing the year 1999.

C) Future Work

After pointing out the gaps in all the data, I realized that this project could improve in many different ways, most especially with improvements in the data-gathering stage. The following improvements could be added to improve the quality of my data analysis, all while staying true to the main research focus of analyzing historical forecast trends:

1. For *eps_fc_terms*. Look at other dates forecasted, not just 3 months before. Incorporate *eps_fc_terms* into my data analysis. Look at other EPS forecasts made from different periods, not just 3 months before. Expand this to include EOD price forecasts as well. This approach would give me greater insight into EPS forecasting trends made at various points in time before the current fiscal period. For example, a potential question to explore would be: *Examine EPS forecasts made 3 months, 6 months, and 9 months prior to the current fiscal period. How do forecasters' levels of optimism and pessimism fluctuate among those different times?*

2. Correctly implement linear regression models. Investigate multicollinearity to see if any other categorical or numerical value had any direct influence on affecting the generated p-values. Could multicollinearity among these relationships—*actual EPS vs. forecasted EPS* and *actual EPS vs. my forecasted EPS*—have directly influenced a false positive result, despite the low Type I error rate of 0.05? All p-values turned out to be exactly 0, which is an extremely odd circumstance.

3. Refocus my broad research question. Analyze relationships among all variables in question during the Great Recession. Expand on how EPS and EOD prices for firms in the 2019 S&P fluctuate *and* respond to real-time stock market dynamic shifts, which come as a response to the economic recession.

4. Discuss incorporation of NLP to scrape social media, analyzing investors' reactions to stock market news. Investigate whether or not online reactions, investor consensus, and investor sentiment have a direct impact on shaping stock market trends.

5. Include the year 1999 for *eps_fc_terms.*

6. Normalize all EPS and EOD time periods through gathering by *fiscal period* instead of *calendar period.* This will add a whole range of depth to my data analysis, and give me more freedom to explore my data as there will be no incongruent, restrictive data types.

7. Devise a more effective strategy for generating naive forecasts. Compare my forecast accuracy not only to Bloomberg's, but also to the methods presented in the studies discussed in the Literature Review section.

8. Keep track of the number of times each firm appeared in the S&P 500 Index over the past 20 years. Isolate the few firms that made only a few appearances. Analyze them individually to see how they differ from the firms that stayed longer in the Index. Similarly, keep track of the firms that stayed in the S&P consistently for 20 years. These approaches would be effective for analyzing outlier firms individually.

9. For the visualizations under Research Question 2, remove outliers that could cause skews in 2000Q4 and 2008Q1. Reexamine these graphs without the outliers, examine the impact of their removal on the modified yearly and quarterly trends, and draw new conclusions for a more in-depth analysis.

Bibliography

Amadeo, Kimberly. "The Great Recession of 2008 Explained With Dates." *The Balance,* Mar 20, 2020. https://www.thebalance.com/the-great-recession-of-2008-explanation-with-dates-4056832

Atsakalis, George S. and Valavanis, Kimon. "Forecasting Stock Market Short-Term Trends Using a Neuro-Fuzzy Based Methodology." *Expert Systems with Applications* 36 (2009), 10696 - 10707.

Baker, Malcolm and Wurgler, Jeffrey. "Investor Sentiment in the Stock Market." *Journal of Economic Perspectives* 21, no. 2 (2007): 129 - 151.

Iacurci, Greg. "Financial Literacy: An Epic Fail in America." *Investment News,* March 2, 2019. https://www.investmentnews.com/financial-literacy-an-epic-fail-in-america-78385

Jackson, Anna-Louise. "Here's How Quickly the Stock Market Recovered After Similar Downturns." *Acorns,* Feb 26, 2020. https://grow.acorns.com/how-quickly-does-the-stock-market-recover-after-downturns/

Jackson, Anna-Louise. "S&P 500 versus the Dow Jones Industrial Average: What's the Difference." *Acorns,* Dec 18, 2019. https://grow.acorns.com/sp-500-versus-the-dow-jones-industrial-average-whats-the-difference/

McKenna, Kristin. "What Happens to the Stock Market After a Recession?" *Forbes,* Apr 3, 2020. https://www.forbes.com/sites/kristinmckenna/2020/04/03/what-happens-to-the-stock-market-after-a-recession/#7a04f9392afb

McLeod, Saul. “What Are Type I and Type II Errors?” *Simply Psychology*, July 4, 2019. https://www.simplypsychology.org/type_I_and_type_II_errors.html

Picasso, Andrea, et al. “Technical Analysis and Sentiment Embeddings for Market Trend Prediction.” *Expert Systems With Applications* 135 (2019): 60 - 70.

RK, Dase and DD, Pawar. “Application of Artificial Neural Network for Stock Market Predictions: A Review of Literature.” *International Journal of Machine Intelligence* 2, no. 2 (2010): 14-17.

Rouse, Margaret, et al. “data quality.” *TechTarget,* November 19, 2019. https://searchdatamanagement.techtarget.com/definition/data-quality

Sankaraguruswamy, Srinivasan. “Investor Sentiment and Stock Market Response to Earnings News.” *The Accounting Review* 87, no. 4 (2012): 1357 - 1384.

Stephanie. “Percent Error & Percent Difference: Definition & Examples.” *Statistics How To,* Nov 8, 2016. https://www.statisticshowto.com/percent-error-difference/

Sun, Yuan, et al. “How Mood Affects the Stock Market: Empirical Evidence from Microblogs.” *Information and Management*, July 26, 2019.

“P Values.” *StatsDirect Limited.* https://www.statsdirect.com/help/basics/p_values.htm

“Python Lists vs. Numpy Arrays - What is the Difference?” *University of Central Florida.* https://webcourses.ucf.edu/courses/1249560/pages/python-lists-vs-numpy-arrays-what-is-the-difference

"Testing the Significance of the Correlation Coefficient." *Lumen Learning.* https://courses.lumenlearning.com/introstats1/chapter/testing-the-significance-of-the-correlation-coefficient/

"Tutorial 1: Downloading End of Day Price Data for S&P 500 Stocks." *MTSM Bloomberg Lab Wordpress Blog*, June 8, 2017. https://mtsmbloomberglab.wordpress.com/2017/06/08/sp500_example/