Economy Data Science

Submitted by:
Britney Fernandez
Nolan O'Donnell
Gregory Miller
Kevin Tu

California State University, Fullerton
CPSC 491
Fall 2022
Advisor: Marc Velasco
Department of Computer Science

1. Proposal/Abstract

This project is a data science project. In which the project takes economic data, such as inflation, wages, stocks and other economic factors and creates a model that can predict new data. The result of the data will provide a narrative to what the data is trying to portray. This can be something about the data, such as why it might be the way it is or what should be done given this new found data. The data will do this specific action, because we will select datasets that are linked to the economy. In order for the data to tell a narrative, we will need to go through the data analyzing and data processing process first usually known as data wrangling. This process includes gathering, sorting and cleaning of the data, so that the data are usable in terms of our specific needs. We created different types of models providing perspective into seeing how well they perform with the given data. Creating new data from the model that we created. The data will provide us with models that can provide visual representations of what the data is trying to convey. We will be leveraging machine learning to train and predict future stocks predictions and to provide data within the economy. As a result, we shall develop a website that will be used as our poster board. Our website will contain all of our models that we have created. This project is designed to provide data and display models on the U.S. economy by analyzing through the process of running different algorithms. Our target users are any economic analysis, data analysis, investors, researcher and so forth. Private and public industries and governments, who are looking for insights into the economy. Our project can help others in their prospect of learning data sciences. And to be able to reference our projects as a guide in their journey towards data sciences.

2. Table of Contents

3. Table of Figures

4. Introduction

This project is a data science research project. That will collect data and make models that will try to accurately predict and tell the current state of the U.S. economy. First, data has to be collected from credible sources, so that the data can be accurate. Some economic factors that we will be using in our models would be factors such as CPI, GDP, Interest Rates and Stocks data. These will be used to provide an understanding of the economy's state such as growth or decline. Once the data is gathered, the data will need to be formatted together so we can begin modeling. This process is known as data wrangling. This might include removing NA values, removing limiting data (that being data that limits how much available data we can use), converting values. We will only combine datasets that fit and are usable within the modeling algorithm that we are planning to create. In modeling, we will be using different algorithms to create different types of models. Single variable and Multivariable linear regression will be used as a starting model to get familiar with the data. From there we will expand to other models such as ElasticNet and any other model that we believe in providing value. We will use tensorflow to create models that will attempt to predict future data. And we hope to improve it and use it to make better models for better predictions. Once all models have been made, we will verify our models accuracy. We will also test how similar our results are to other institutions, to see if we are off or close to a professional prediction. Then we will post our models and explain our models on our website. We will have our models displayed on the website as our posterboard, so that people can read our findings. And learn about our data science process and we hope our research benefits others.

5. Project Plan

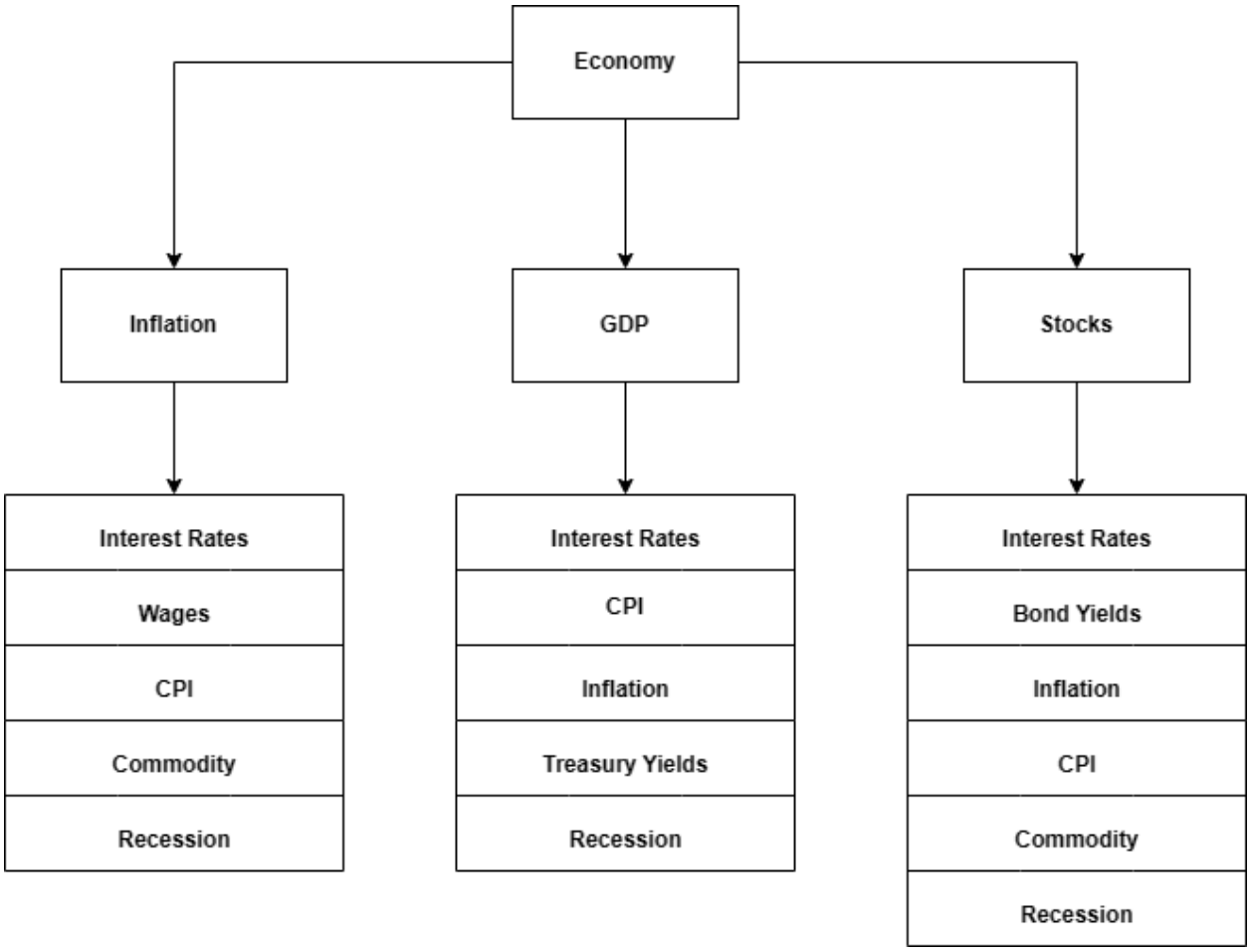


Figure 5.1 Economy Model Outline - This figure depicts an outline of economy models and their corresponding potential dataset.

6. Metrics

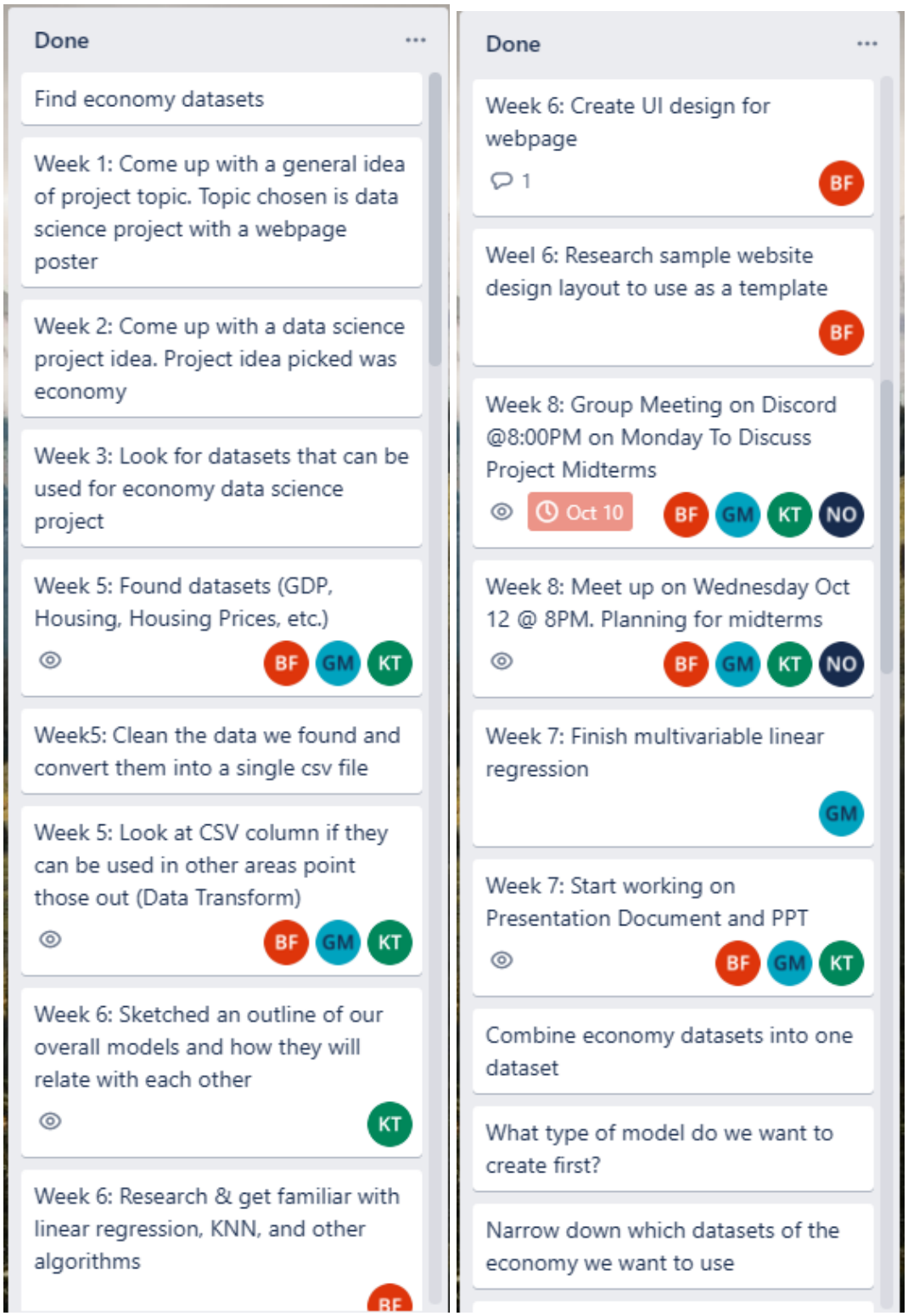


Figure 6.1 Trello Mid Sprint Done Column - This figure depicts the first half of completed task within the project as a metrics of progres.

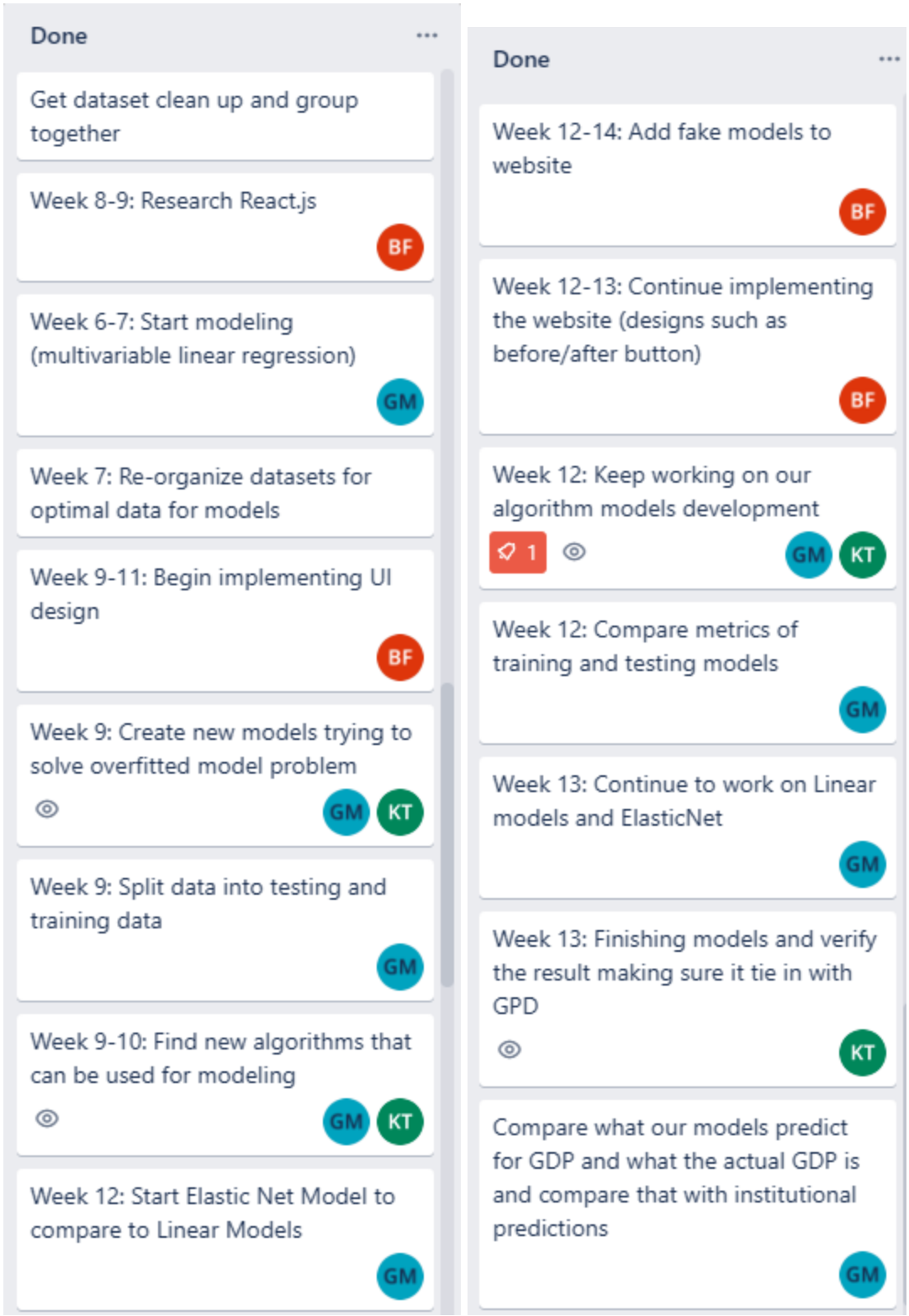


Figure 6.2 Trello Final Sprint Done Column - This figure depicts the last set of completed task within the project as a metrics of progres.

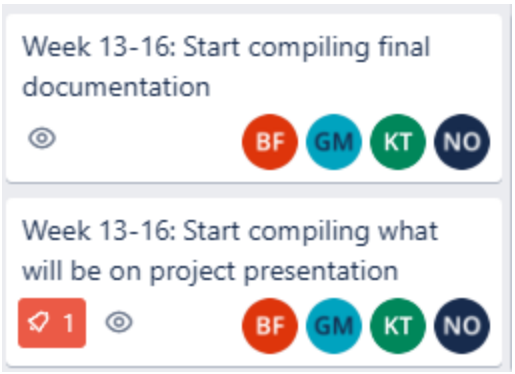


Figure 6.3 Trello Completed Sprint Done Column - This figure depicts the complete set of completed tasks within the project as a metrics of progres.

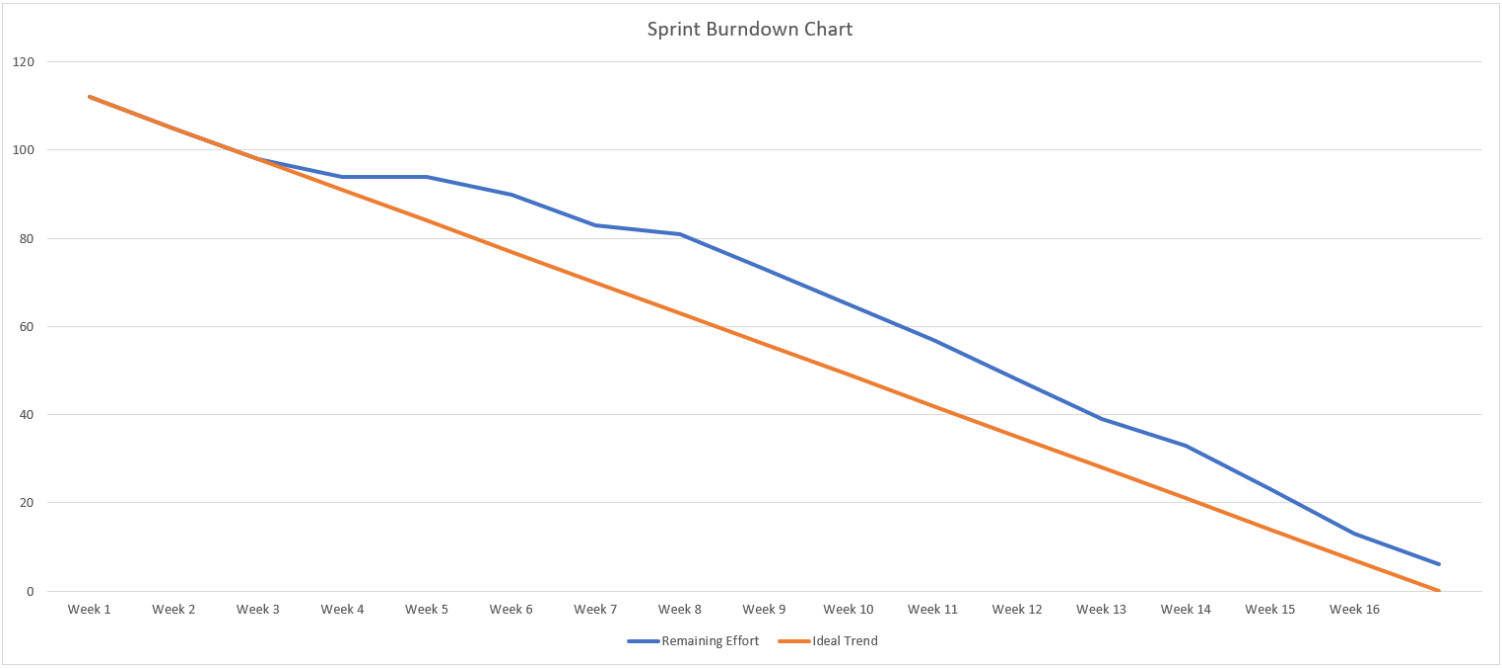


Figure 6.4 Sprint Burndown Chart - This figure depicts the project final sprint progress at week 16.

7. Detailed Technical

7.1 Requirements, Specification, Design or Algorithms

Yearly Models:

Data:

The first requirements for this data science project was to find credible economic data for the United States. This would include gdp, bond yields, cpi, unemployment, labor force, population as well as a few others. The data we found was a yearly collection of data, we would after the midterm switch to using quarterly data. This yearly dataset was found on datahub.io (see **reference 1**), which compiled a dataset for us so we could begin modeling. The source of datahub.io's economic indicators was the Bureau of Economics Analysis, which is a very reliable source of data. As was mentioned earlier, models were created using yearly and quarterly data and we will show those comparisons later.

Another step in the beginning of data science should be understanding more about the data. The correlation matrix below in **Figure 7.1.1** shows the correlation between variables in the dataset. Variables that are dark blue and dark red mean that those variables are either positively correlated (blue) or negatively correlated (red). The target variable was levelchained (GDP chained to 2009 inflation) and we can see that it is very positively correlated with many of our variables and only a few variables are not so correlated, such as unemployed_percent, cpi, and bond yields. For example when we look at population we see that it is very positively correlated with labor force as well as the other blue boxes.

Figure 7.1.1 Yearly Correlation Matrix - This Figure shows the correlation and how strong that correlation is between variables used in the yearly model

Data Wrangling:

Once the data was found, the data wrangling part of the project started. Most of the datasets had to go through similar wrangling before they could be combined. These steps include changing value types, removing NA values, removing unneeded columns, removing columns with limited data, and renaming columns. These steps are very important so that we can combine the datasets into one dataset that data science can be done on. A mistake that we made early on when combining data that should not have been combined due to one of the datasets having much fewer years of data. This took our dataset to having hundreds of data points down to only a few dozen. We resolved this by only combining datasets that have a certain amount of data together, that way our master data set would have as much data as possible. We now have multiple master datasets, one with yearly data (limited data) and ones with monthly and quarterly data which is much better data for modeling.

Linear regression:

```
lm1 <- lm(formula = GDP~unemployed_percent + population, data = temp_data)
```

Figure 7.1.2 Linear Model Formula - This figure shows an example formula for linear regression

The linear regression models were tested by comparing values that can determine the fit of a model. Those values were residual standard error and adjusted r-squared value, p-value. These values can tell how well a model is performing. Residual standard error can tell how well the regression model fits the dataset; a higher number means it does not correctly predict new data. While adjusted R squared can tell us the goodness-of-fit of the model; a higher number means the model fits the current data well.

```
Call:
lm(formula = levelchained ~ unemployed_percent + population +
    bondY + cpi + labor_force + changecurr, data = econ_data)

Residuals:
    Min      1Q  Median      3Q     Max
-434.15 -164.59  -29.69  177.91  420.27

Coefficients: (1 not defined because of singularities)
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       -8.136e+03  3.507e+02 -23.199  < 2e-16 ***
unemployed_percent -1.752e+02  2.485e+01  -7.048 4.15e-09 ***
population         1.593e-01  1.380e-02  11.542 5.84e-16 ***
bondY             -1.712e+02  1.460e+01 -11.720 3.28e-16 ***
cpi                       NA         NA      NA       NA
labor_force       -8.129e-02  1.844e-02  -4.408 5.26e-05 ***
changecurr        -3.697e+01  1.281e+01  -2.885  0.00568 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 231.3 on 52 degrees of freedom
Multiple R-squared:  0.9968,    Adjusted R-squared:  0.9965
F-statistic:  3283 on 5 and 52 DF,  p-value: < 2.2e-16
```

Figure 7.1.3 Yearly Model Results- This figure shows the results about a given model

Once the data is combined into a dataset, modeling can start. The first models that were created used linear regression. The majority of the modeling was done using multivariable linear regression using r-studios in built function lm(). This function takes in a formula such as ‘GDP~CPI + Population’ (See **Figure 7.1.2**) which will create a linear model that is predicting GDP given CPI and population. This model has a summary report that can be printed that shows many of the model's values that can be used to determine the fit of the model. The main values used for determining the fit of that model is residual standard error and adjusted r-squared , these values can be used to see each model's fit (see **Figure 7.1.3**). How we create different models is changing that formula shown above. We can add as many variables to the model in determining GDP as we want. All features in the data can be used. These models will give us different

residual standard error and adjusted r-squared values which we can use to determine the best model.

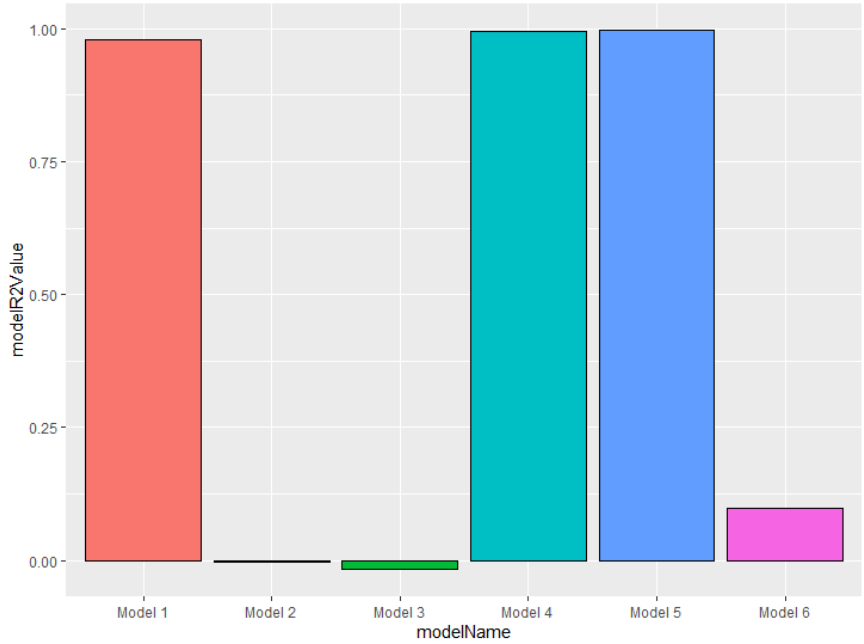


Figure 7.1.4 Yearly Model R2 Values - This figure shows the r-squared values of some yearly models

Conclusion:

Many more models were created than the six shown **Figure 7.1.4** but only wanted to show some. In **Figure 7.1.4** Model R2 Values, we can see that some models that were created have very low R2 values, indicating the model is not very good. However, we can see that some models have very high R2 values. When looking at those models that have high R2 values we can see that the majority of the R2 values only comes from a few variables. Model 5 for instance has six features used in predicting GDP, while model 1 only used two features. This indicates that the feature 'population' has a very high determining factor in determining the GDP. Another conclusion we can make from that, is that features such as CPI, Bond Yields, and other economic factors that were collected did not have a large impact on the creation of the models. Generally the models that were created shared high residual standard error, showing the model does not do well on prediction new data. However, for some models that have high R2 values meaning the linear model fit the data well. These two indicators show that the models seem to be overfitted and will suffer in predicting new data even though it fit its current data very well. The current reason as to why this is happening is that many of the features used in modeling are extremely correlated. Principal Component Analysis was done to try and see which variables were highly correlated, See **Figure 7.1.1** which depending on the color of the variable will determine which other variables it is correlated with. The lm() function will remove data in the model that is highly correlated which is why in some models we see that some features were removed. The

goal of creating a good model in predicting GDP will be finding the model with the highest R2 value while at the same time, trying to reduce the residual standard error.

Quarterly Models

Data:

After modeling using the yearly data, we decided that we wanted to have more data for our models. We would again go out and find more data, preferably data that was at least quarterly instead of yearly. Instead of using datahub.io's collection of variables, we would find our own collection. We would use FRED (Federal Reserve Economic Data, see **reference 2**) which was created and is maintained by the Research Department at the Federal Reserve Bank of St. Louis. This is one of the most credible sources of economic data available to the public. The next step was deciding what variables to include into the dataset. We would go with some of the same variables such as CPI, GDP(target variable), bond yields but changed others given that we learned more about some economic variables that might be good to use. Some of these new variables would be unemployment insurance claims, average hours worked, housing permits issued. These indicators are leading indicators which means that these are some of the first things that will move in the economy. For example when workers are laid off they go to get unemployment insurance, or when an employer needs to cut labor costs they reduce the hours worked for their employees.

Similar to the yearly section we would again try to better understand our data. The correlation matrix below in **Figure 7.1.5** shows the correlation between variables in the dataset. Variables that are dark blue and dark red mean that those variables are either positively correlated (blue) or negatively correlated (red). The target variable was GDP and we can see that unlike the yearly dataset it is not correlated with very many variables and the ones that are correlated, we see a negative correlation. If we look at the correlation below we can see that GDP is not very correlated to interest rates, since the int_rate box is basically white meaning zero correlation. If we look at bond yields, we see that it is very red, meaning that bond_yield is very negatively correlated with GDP. This means that when GDP goes up we expect bond yields to go down and vice versa. As we see in the correlation matrix there is a blue diagonal line through the matrix, this shows that each variable is one hundred percent correlated with itself. If we compare this correlation with the yearly one in **Figure 7.1.1**, we can see that most of the variables are not correlated with one another, which was one of the goals of the new dataset.

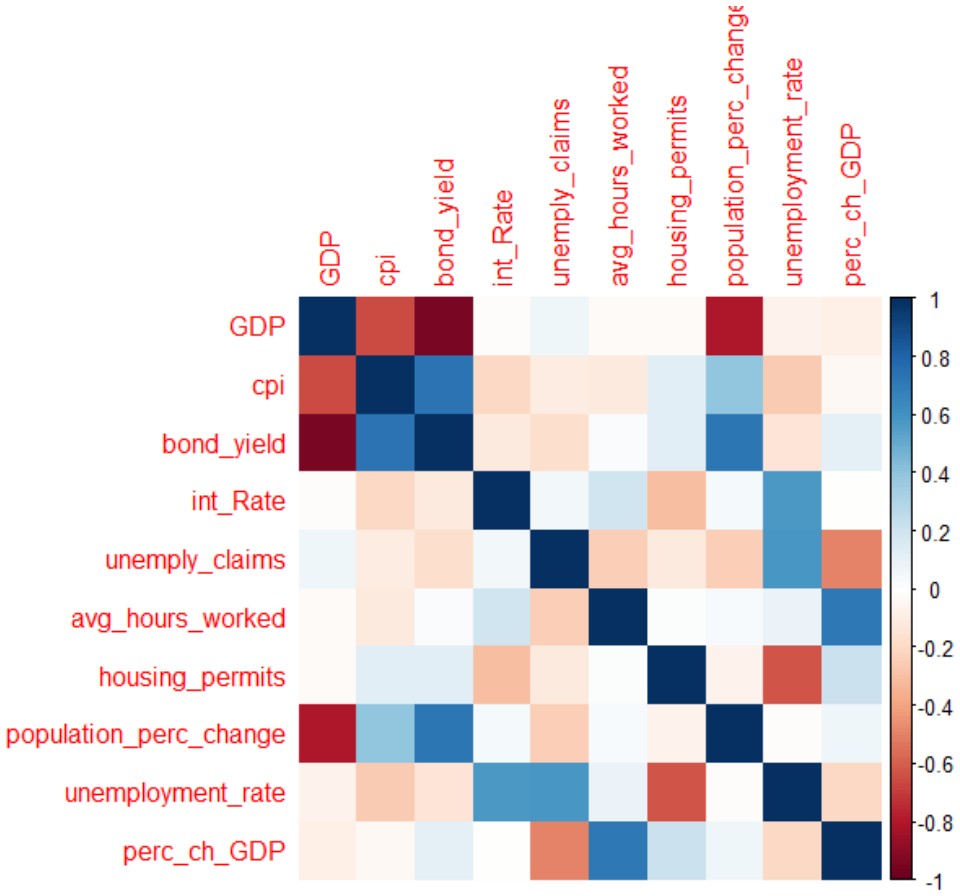


Figure 7.1.5 Quarterly Correlation Matrix - This Figure shows the correlation and how strong that correlation is between variables used in the quarterly model

Data Wrangling:

Like the yearly sections above, we will again wrangle the data into a format that can be used for modeling. This would include joining the different data into one dataset. This step was much easier than the yearly data since we got each variable from FRED, which would provide us a csv file that would have only the date in one column and the variable such as GDP, CPI in the other column. The only real cleaning that needed to be done was changing the column names that were giving joining the tables some issues. Unlike the yearly data which had 20 rows, after joining all the tables by date, we had over one hundred rows of data points, which is a significant increase in data. Another data wrangling step that was done for quarterly data, but not yearly data, was splitting the data into training and testing datasets. This split would be used to better evaluate the models by testing the model on data that it has never seen before. The test/train split was seventy percent training and thirty percent testing. Another step that was done was making sure that the testing data had the most recent quarterly data, this would include Q1, Q2 of 2022. This was done so that we can see how well our model does in a recent time frame, making it more relevant to today. The last data wrangling step that was done was normalizing the data,

however, this did not have much of an affect on the models so we won't be discussing it further and we didn't end up sticking with keeping our data normalized.

Linear regression:

Just like the yearly model described above we compared the models values such as residual standard error and adjusted r-squared value, p-value to compare the models to one another. These values can tell how well a model is performing. Residual standard error can tell how well the regression model fits the dataset; a higher number means it does not correctly predict new data. While adjusted R-squared can tell us the goodness-of-fit of the model; a higher number means the model fits the current data well. All of the models below will all have p-values that are considered significant, meaning the p-value is less than 0.05, so we will not discuss the p-values in lower sections.

```
Call:
lm(formula = perc_ch_GDP ~ cpi + bond_yield + int_Rate + unemply_claims +
    avg_hours_worked + housing_permits + population_perc_change +
    unemployment_rate, data = qrt_train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-1.02019 -0.26935 -0.05886  0.24719  1.41203

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)              6.855e-01  5.895e-01   1.163  0.24805
cpi                     -1.853e-01  8.979e-02  -2.064  0.04194 *
bond_yield               1.454e-01  5.156e-02   2.820  0.00593 **
int_Rate                -4.394e-03  6.069e-02  -0.072  0.94245
unemply_claims          -3.258e-06  3.449e-07  -9.447 4.92e-15 ***
avg_hours_worked         8.086e-02  2.512e-02   3.218  0.00181 **
housing_permits          5.404e-04  1.949e-04   2.773  0.00678 **
population_perc_change -2.887e+00  1.075e+00  -2.687  0.00863 **
unemployment_rate        1.644e-01  7.625e-02   2.156  0.03382 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4827 on 88 degrees of freedom
Multiple R-squared:  0.808,     Adjusted R-squared:  0.7905
F-statistic: 46.28 on 8 and 88 DF,  p-value: < 2.2e-16
```

Figure 7.1.6 Quarterly Model 1 Summary - This figure shows the summary of quarterly model 1

Once the quarterly data was collected and compiled into a usable dataset, modeling can begin. The first model that we will look at uses all the variables that are in our dataset. The formula can be seen at the top of **Figure 7.1.6** above. This summary also gives us coefficients which tell us how to create the line that the model creates along with the intercept. For example we can say that for every increase in GDP along the x-axis, cpi will decrease by -1.853e-01 and so on for the other variables. The next statistic to look at is the adjusted R-squared value which can show us how well our model can explain the variance in the dependent variable (GDP) by the movement of the independent variables (CPI, unemployment claims, etc). The adjusted R-squared value tries to adjust to the multiple variables that we use, since the more variables we

have we expect to be able to account for more variance. Both the adjusted and normal R-squared values are around 0.8 or 80%, which is very good. The model seems to be performing very well along with the fact that we have a low residual standard error (0.4827). A low residual standard is what you want for a model since the smaller value means that the model does better at predictions than a model with a higher value.

Comparing this first quarterly model to the yearly model, we can see that the adjusted R-squared value is much smaller in this model than that of the yearly one. However, in the conclusion of the yearly model, we said that we believed that the model was overfitted to the training data which gave us a high R-squared value, but when we looked at the residual standard error (231.3, see **Figure 7.1.3**) it is much higher than this model (0.4827). This shows that this model is not overfitted, unlike the yearly model. We can see below in **Figure 7.1.7** when we test our model on data that it has never seen before it does seem to get the general trend of the data, although it does seem to not account for all the variance, which is why for this model the adjusted R-squared value is in the 80% and not higher. See below in the our models vs others section for seeing what this and the other models below predicted percent gdp growth for quarter 1, 2 of 2022.

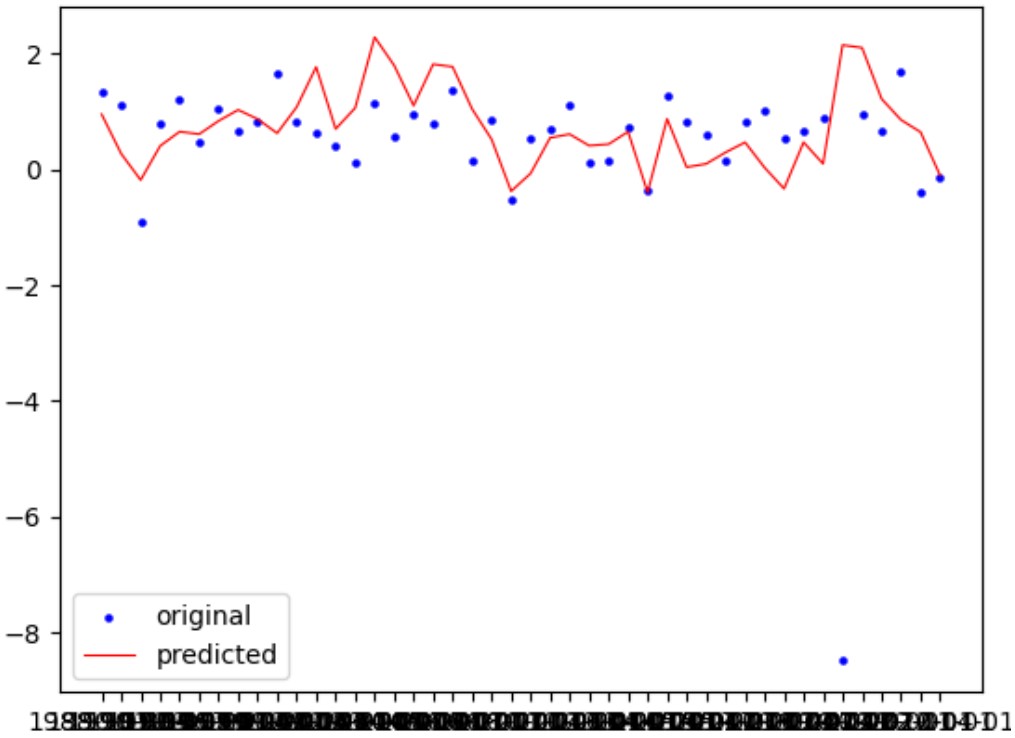


Figure 7.1.7 Model 1 Original Vs Prediction - This figure shows a graph with the original data and the predicted values of quarterly model 1

```
Call:
lm(formula = perc_ch_GDP ~ cpi + bond_yield + population_perc_change,
    data = qrt_train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-8.3989 -0.2186  0.0404  0.3591  2.0089

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)            0.57431    0.44120   1.302   0.1962
cpi                   -0.39374    0.17308  -2.275   0.0252 *
bond_yield             0.22224    0.09915   2.241   0.0274 *
population_perc_change 0.53316    2.14201   0.249   0.8040
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.015 on 93 degrees of freedom
Multiple R-squared:  0.1031,    Adjusted R-squared:  0.07415
F-statistic: 3.563 on 3 and 93 DF,  p-value: 0.01717
```

Figure 7.1.8 Quarterly Model 2 Summary - This figure shows the summary of quarterly model 2

This second model we wanted to see what would happen if we just used the highly correlated variable to predict GDP. The formula can be seen at the top of **Figure 7.1.8** above. This summary also gives us coefficients which tell us how to create the line that the model creates along with the intercept. For example we can say that for every increase in GDP along the x-axis, cpi will decrease by -0.39374 and so on for the other variables. The next statistic to look at is the adjusted R-squared value which is horrible. It means that we cannot account for variance in GDP given our independent variables. This is not something I would think would happen, since these variables are already highly correlated with the dependent variable. Even though the model has a low R-squared value the model has a low residual standard error (1.015). If we recall, a low residual standard is what you want for a model since the smaller value means that the model does better at predictions than a model with a higher value. Given that the adjusted R-squared value is so low, we would not expect the model to predict very well, however, when we looked at the predicted values this model gave us, it was surprisingly accurate at predicting the quarter 1 and 2 of 2022.

When we compare this model to the yearly and quarterly model 1, this model does not look impressive, in fact it looks pretty horrible. It is hard to say if this model is better than the yearly model, since that model is overfitted and has a very high residual standard error, but this model's R-squared value is extremely low. The details of this model honestly is not the interesting part, it's the fact that given the model is so bad, how did it predict so well, which will be discussed below. Looking at **Figure 7.1.9** below we can see that the model does seem to get some of the variance but it hardly performs and it is very noticeable to the eye.

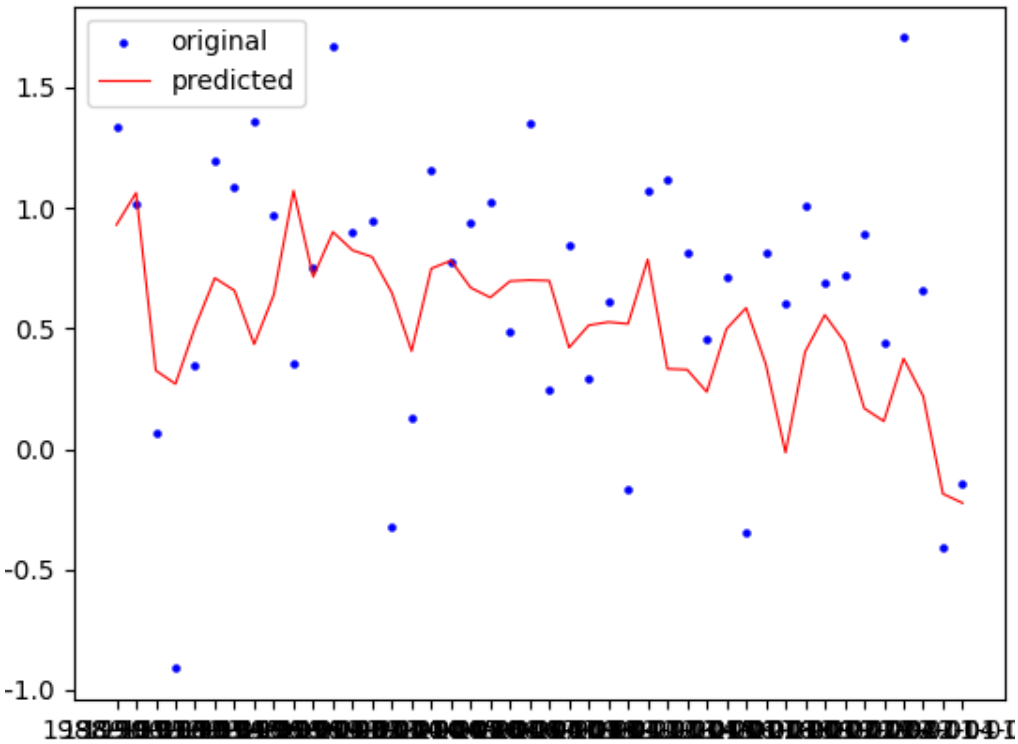


Figure 7.1.9 Model 2 Original data Vs Prediction - This figure shows a graph with the original data and the predicted values of quarterly model 2

```
Call:
lm(formula = perc_ch_GDP ~ int_Rate + unemply_claims + avg_hours_worked +
    housing_permits + unemployment_rate, data = qrt_train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-1.03963 -0.31165 -0.03244  0.25191  1.61215

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       -2.300e-01  4.988e-01  -0.461 0.645765
int_Rate          -4.529e-02  5.967e-02  -0.759 0.449744
unemply_claims    -3.298e-06  3.320e-07  -9.933 3.52e-16 ***
avg_hours_worked   9.158e-02  2.571e-02   3.562 0.000588 ***
housing_permits    6.961e-04  1.888e-04   3.686 0.000387 ***
unemployment_rate  2.111e-01  7.262e-02   2.907 0.004581 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4995 on 91 degrees of freedom
Multiple R-squared:  0.7874,    Adjusted R-squared:  0.7757
F-statistic:  67.4 on 5 and 91 DF,  p-value: < 2.2e-16
```

Figure 7.1.10 Quarterly Model 3 Summary - This figure shows the summary of quarterly model 3

The third quarterly model that we are showing is one where we wanted to use the least amount of variables, while still performing well, that being having a high R-squared and a low residual standard error. The variables that were removed were the highly correlated variables (darker boxes in **Figure 7.1.1**), or the variables that we used in quarterly model 2 (**Figure 7.1.8**).

If we compare this model to quarterly model 1 (**Figure 7.1.6**) we see that they are very similar. The residual standard error is very low and a very similar value to that of this model. This means that like the first model, this one should predict value well. The R-squared value is also almost the same, meaning that we account for almost the same variance of GDP given less variables used. This is impressive since we went from 8 variables to 5 variables and are still getting models roughly the same.

When we look at what the model predicts in **Figure 7.1.11**, we can see visually that the model does seem to get the general trends like what we see in quarterly model 1. This model may even look better to the eye. Although it does seem to overstep or understep the original value as we see that it peaks above and below some values. However, the general trend seems good. We will compare the quarter 1, 2 of 2022 predictions below.

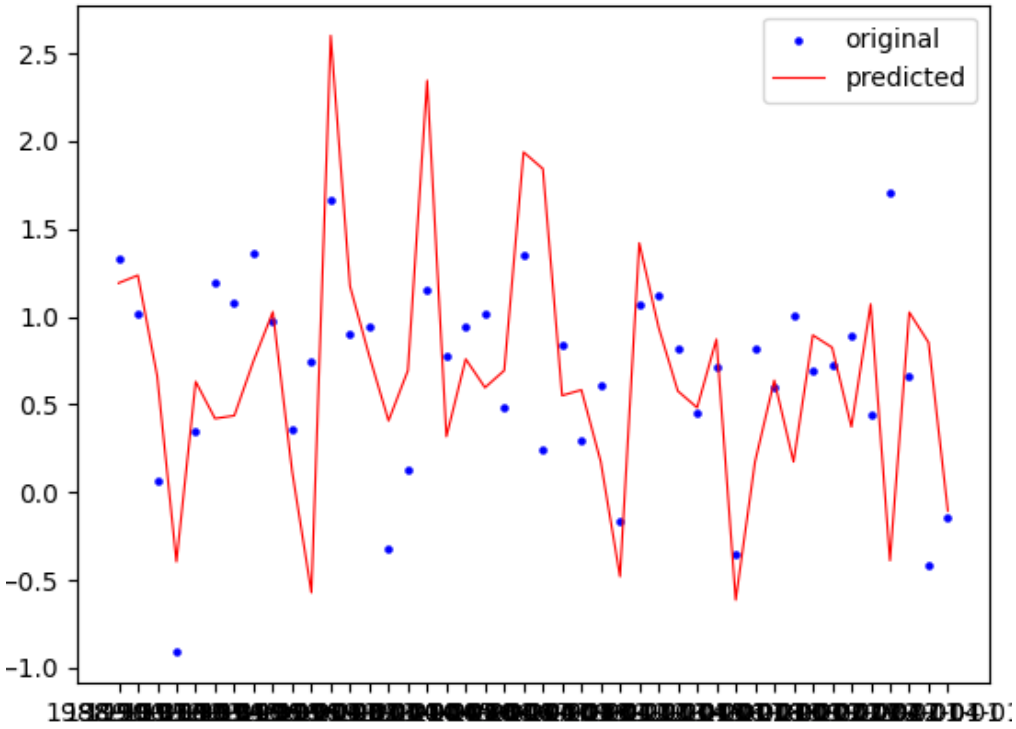


Figure 7.1.11 Model 3 Original data Vs Prediction - This figure shows a graph with the original data and the predicted values of quarterly model 3

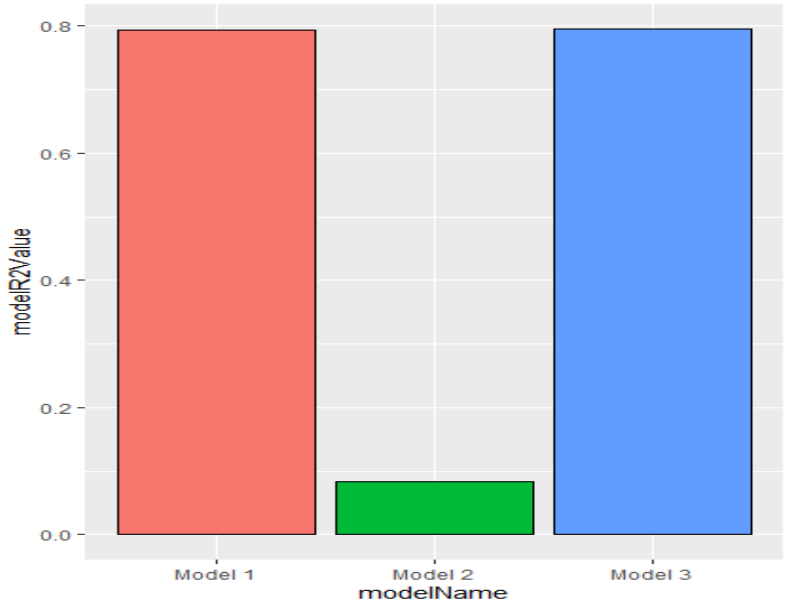


Figure 7.1.12 Quarterly Model R2 Values - This figure shows the r-squared values of the three quarterly models above

Quarterly Model Conclusion:

The models created using the quarterly data, as well as splitting the data into testing/training data, seemed to be a success when trying to make better models that are not overfitted. When we look at **Figure 7.1.12** we see that model 1 and model 3 are very similar in terms of R-squared values. When discussing these models above, we also said that the residual standard error for model 1 and model 3 were also very similar. The main difference between these two models (model 1 and model 3) is that model 3 has far fewer independent variables used to create the model. However, this seems to not have changed the outcome of the model. Using the model summaries we can see that the best model for R-squared and residual standard error is model 1, with model 3 being closely behind and model 2 being the worse. It seems that the addition of more data (quarterly vs yearly) and the splitting of training and testing data seemed to make these models perform better. Below we will compare what the models would predict for quarter 1 and 2 for 2022. This might give us a little more insight into the model's predictive powers.

Elastic-Net Regression:

Elastic net was chosen as the second algorithm to try and compare to the multivariable linear regression models. This was done using Python instead of R, which is why we don't have a nice summary to look at. Although the most important parts are that the Elastic-new model had a very high R-squared value (0.9427 or 94%), indicating that the model can explain the variance in the dependent variable given the independent variables. The RMSE (root mean square error is

rather large of 768, indicating that the model seems to be relatively off when predicting values, and we will take a look at the predicted values for quarter 2 of 2022 since quarter 1 could not be gotten. Like the quarterly models above, the data for modeling was split into training and testing, that way when we test, the model has never seen these values before.

When looking at the graph below (**Figure 7.1.13**) we can see that the elastic-net model did very well in showing the variance in the data. When we look at the graph, when the GDP is low the model predicts low and when the GDP is high the model accurately predicts a high GDP. This can be seen very well visually through the graph below. However, how well will this model do compared to our other models and to what others might predict.

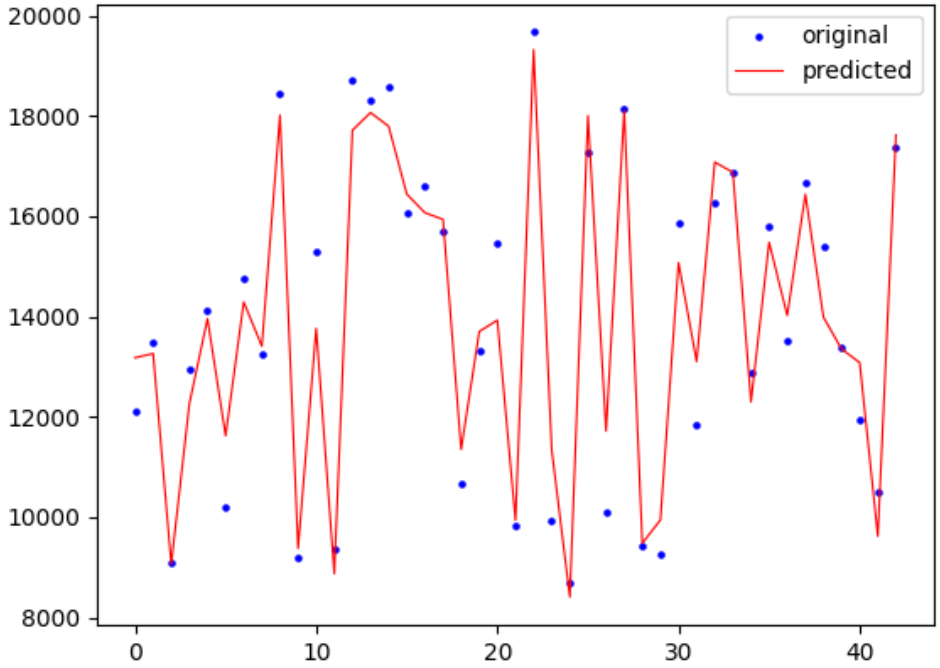


Figure 7.1.13 Elastic-Net Original data Vs Prediction - This figure shows a graph with the original data and the predicted values of quarterly model 1 of elastic net

Elastic-Net Quarterly Model Conclusion:

This model seems to have some similar measurements as the yearly models, in that we think it is overfitted. The reason behind that conclusion is that the R-squared value is very high (.94) and the RMSE is also very high (700+) which is the root mean squared error which tells us how off our predicted value is from the original. So, while the dependent variable's variance is highly accounted for by the independent variables and error in the predictions is higher than it should be. We are not sure where to go from here since we didn't have a lot of time to continue working on the Elastic-Net model. Despite that, the model actually did still predict the change in GDP fairly well and while it was the most off, it was close enough (See section below)

Our Models Vs. Others

This part was one of the more fun parts of the project, since we have models and what they show and we want to compare those results to other institutions. The two graphs below were gotten from **reference 3**, but they got it from The Atlanta Fed **reference 2**. A quick description of what the graphs below are (**Figures 7.1.14**, **7.1.15**), they are showing The Atlanta Fed GDPNow estimate which is their prediction for where the current GDP is for the United States (green line). The blue line is the top 10 and bottom 10 of the Blue chip consensus, which is the average of some of the top 50 private sector forecasters and the GDP that they are predicting over that quarter. The first **Figure 7.1.14** is the first quarter of 2022 and the second **Figure 7.1.15** is the second quarter of 2022. We also have our own predictions for those two quarters that we will share below. The actual final value for the change in GDP for quarter 1, 2022 was -0.41%. The actual final value for the change in GDP for quarter 2, 2022 was -0.15%.

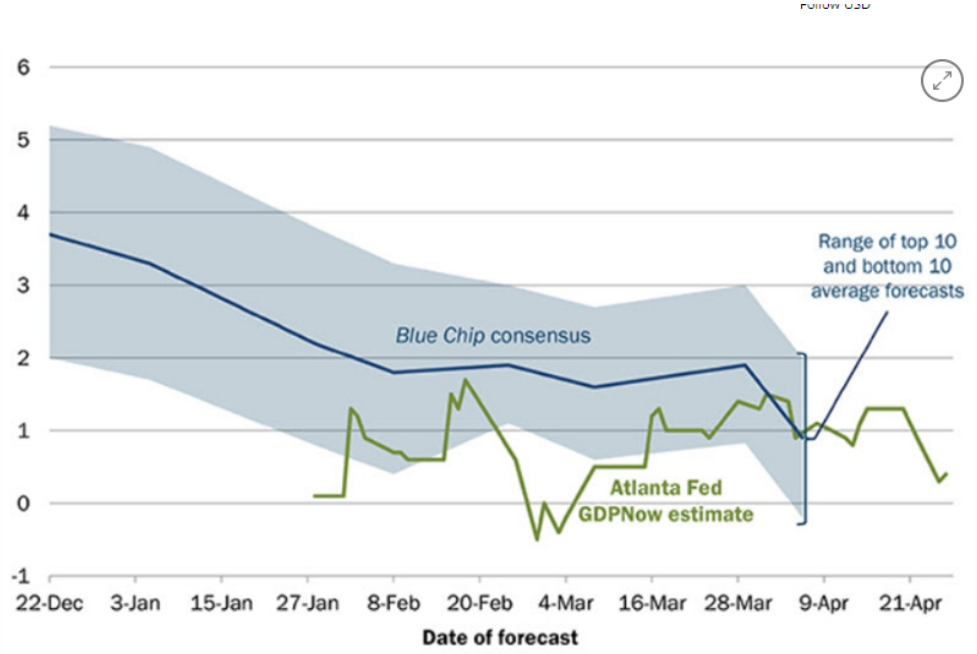


Figure 7.1.14 Others Predictions Q1 2022 - This figure shows a graph of the Atlanta Fed GDPNow estimate for Q1 2022

Figure 7.1.14 which is quarter 1 of 2022 shows that the Blue Chip consensus was ranging GDP growth to be anywhere between negative 0.1% and positive 2.0% with the average being about a 1.0% increase. The Atlanta Fed seemed to be about a 1.2% increase to about a 0.2% increase, although we see the trend going downward towards the end of the quarter.

Our first quarterly model predicted that the change in GDP would be 0.64% change. When we first saw this number we originally thought that we were very off from the actual percent change in GDP but compared to others, our model seemed to do pretty well.

Our second quarterly model predicted that the change in GDP would be -0.19% change. When we saw this, it was confusing since the second model had a very low R-squared value but it did have a low residual standard error, which is a measure on how well a model can predict. We are still surprised this model performed so well.

Our third quarterly model predicted that the change in GDP would be 0.85% which is definitely higher than that of model 1 but is still well within range of the Blue chip and Atlanta Fed.

We could not get the percent change in GDP for quarter 1, 2022 from our elastic-net model but we will discuss it for elastic-net quarter 2, 2022.

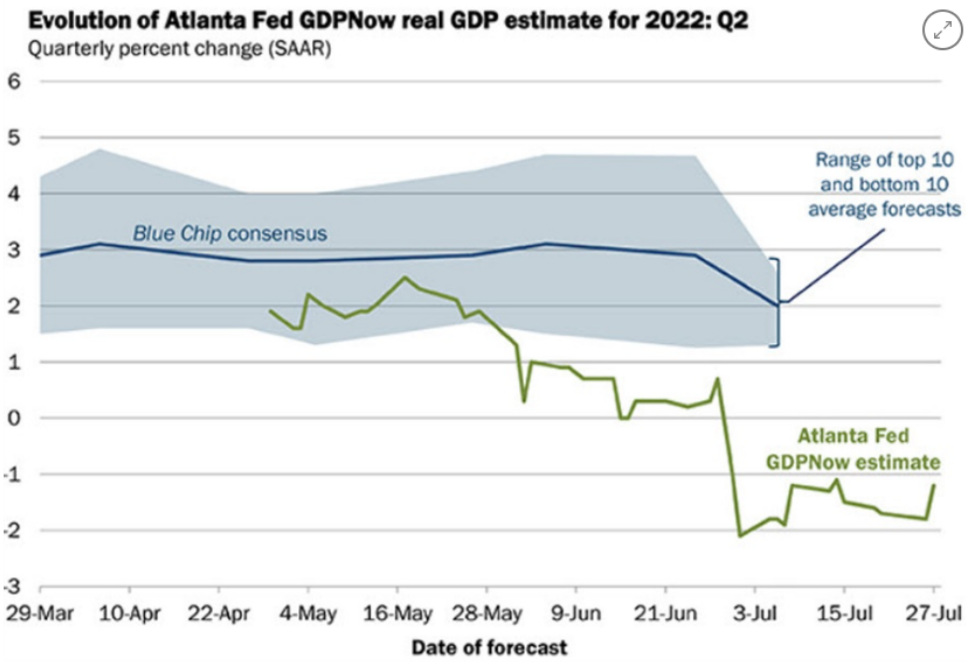


Figure 7.1.15 Others Predictions Q2 2022 - This figure shows a graph of the Atlanta Fed GDPNow estimate for Q2 2022

Figure 7.1.15 which is quarter 2 of 2022 shows that the Blue Chip consensus was ranging GDP growth to be anywhere between positive 1.3% and positive 2.9% with the average being about a 2% increase, and converging on 2%. The Atlanta Fed seemed to be about a -1% to about a -2% decrease, although we see the trend going upwards towards the end of the quarter.

Our first quarterly model predicted that the change in GDP would be -0.11% change. When we saw this we were very happy that the model predicted the GDP change very well, in fact it did better than Blue chip and Atlanta Fed.

Our second quarterly model predicted that the change in GDP would be -0.22% change. Just like for quarter 1 above, we do not understand why this model predicted both quarters so well.

Our third quarterly model predicted that the change in GDP would be -0.1% which is still really good compared to others.

The elastic-net model predicted that the change in GDP would be -5.8% which is very off, however, it is in a similar range to that of the others.

Logistic Regression:

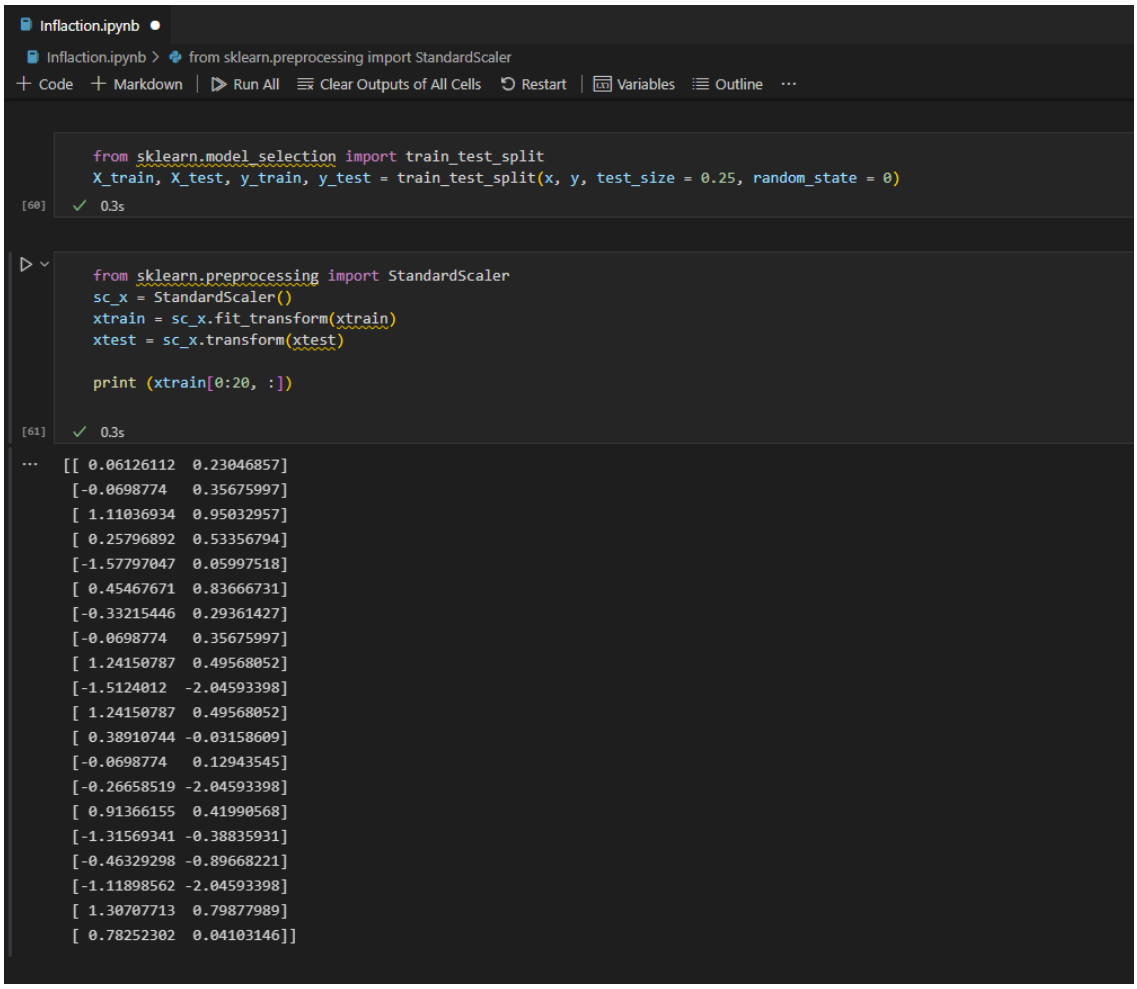


Figure 7.1.16 Logistic Regression - This figure shows the result output of the model Logistic Regression.

For **Figure 7.1.16,** the Logistic Regression model is model based on the dataset containing wages and CPI rates. The goal of this model is to calculate whether or not inflation does indeed exist. Logistic regression will output the result in a form ranging between 0 and 1. Where 0 is false, in that the odds of occurrence are not likely. While 1 is true, where the odds of occurrence are highly likely to occur. The current state of the model does not provide much clarity as to whether or not inflation will occur or not. This is because of the model error and needs of optimizations.

Evaluating Economic Data with Machine Learning:

As another implementation to evaluate economic data, the implementation of machine learning tools was utilized in order to show a distinct perspective of data science in the evaluation of economic data. Time series forecasting is the best way to utilize machine learning in a predictive model setting. Through relevant stock data, a model is able to be created that trains off a subset of this data in order to create accurate predictions for future stock prices.

Wrangling Data

The data used for the machine learning portion of the research project was done through Yahoo Finance. In order to gather and wrangle this data, a query was made to the Yahoo Finance API in order to pull the csv. The csv was then transferred to R Studio and wrangled and fixed to be easily modeled data. TensorFlow requires that the Date column is converted to a datetimens64 variable rather than an object in python. With Pandas, the date column was able to be converted to it's appropriate value in order to be modeled.

Initial Setup

For the initial set up, the data set is verified and plotted. The wrangled data set is then utilized as a base model for what our model in training is tested against.

```
data = pand.read_csv('./data/GSPC_data.csv')

# Verify our data is imported correctly
print(data.shape)
print(data.sample(5))

# Verify data set columns and values
data.info()

# TENSORFLOW - Convert Date from Yahoo Finance records to 'dateTime' for Pandas
data['Date'] = pand.to_datetime(data['Date'])

# Verify date column has changed
data.info()

# Initial plot of our data set
# Plot is Date/Close as Close shows the value of the stock on a particular day at close time
plt.plot(data['Date'], data['Close'])
plt.xlabel("Date")
plt.ylabel("Close")
plt.title("S&P 500 Prices 2022")
plt.show()
```

Figure 7.1.17 Initial Plot - This figure shows the logic behind the initial plotting of the model the machine learning model will train against.

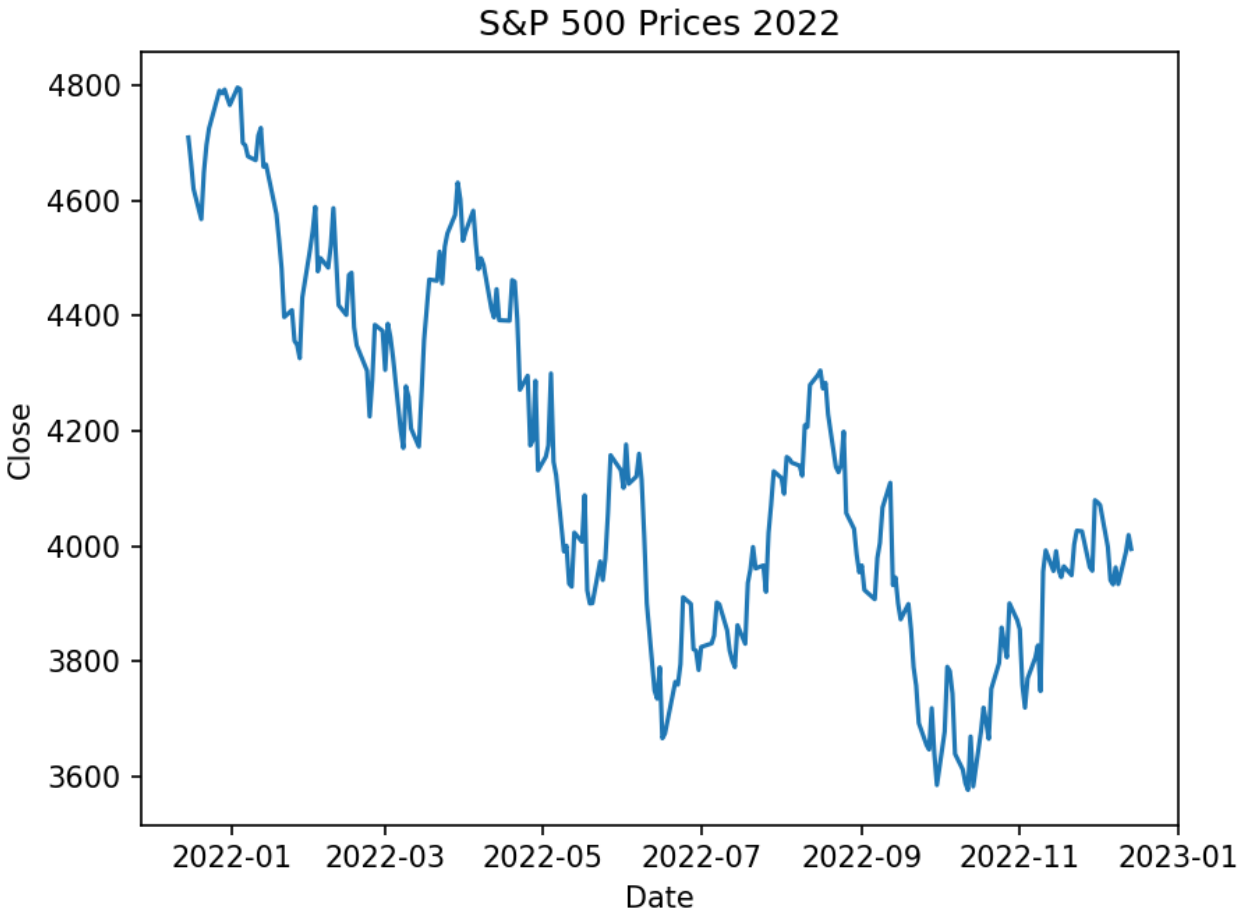


Figure 7.1.18 Initial Plot Display - This is the visual representation of the model the machine learning model is being tested against. The subset that is being taken is represented at the end.

Data Set Preparation

In order to create an accurate prediction model, the model must take a subset of data from the initial model that it can identify and garner trends from. This subset is mentioned in Figure 7.1.18, where the tail of the original model is taken and used to train our machine learning model. The length of this model is output to the console and this is used as our future 'Date' variable for the latter renditions of the predictive model.

```
# Prepare training set
close_data = data.filter(['Close'])
dataset = close_data.values
training = int(np.ceil(len(dataset) * .95))

# Output the length of our training set
print(training)


scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(dataset)

train_data = scaled_data[0:int(training), :]
# prepare feature and labels
x_train = []
y_train = []

for i in range(60, len(train_data)):
    x_train.append(train_data[i-60:i, 0])
    y_train.append(train_data[i, 0])

x_train, y_train = np.array(x_train), np.array(y_train)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
```

Figure 7.1.19 Logic For Training Set - Training set that is instantiated for the predictive model to train against.

LSTM Model Creation

The creation of the LSTM model is then applied, where the model is instantiated for the predictive machine learning model. Keras, with Tensorflow built on top, is utilized to deploy a simple sequential model with LSTM in tandem in order to enable the ability for time series

forecasting.

```
# Prepare our LSTM Model (sequential model base)
model = keras.models.Sequential()
model.add(keras.layers.LSTM(units=64,
                            return_sequences=True,
                            input_shape=(x_train.shape[1], 1)))
model.add(keras.layers.LSTM(units=64))
model.add(keras.layers.Dense(32))
model.add(keras.layers.Dropout(0.5))
model.add(keras.layers.Dense(1))

# Output for the model - verifies all layers added are true
model.summary

model.compile(optimizer='adam',
              loss='mean_squared_error')
```

Figure 7.1.20 Model Logic - Logic found behind the creation of the predictive model

Training, Epochs, and Predictions

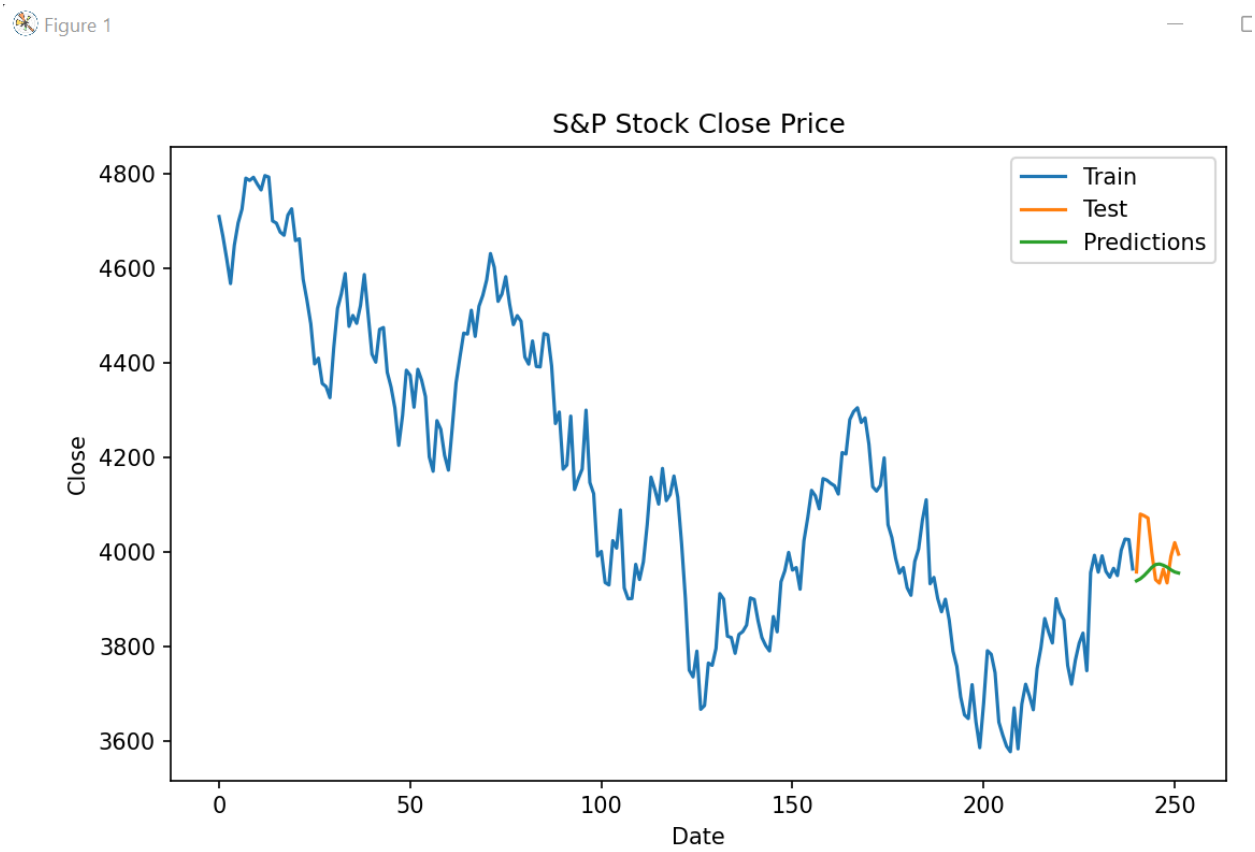
The last portion of the machine learning model is the iteration of training. Training of the machine learning model is done in sections called epochs. Training involves training against a training set that originates from the original model. Each iteration of training is done with a prediction and analysis is done at the end to check how closely this predictive model matches the initial model. The closeness of the model is measured in MSE and RMSE. Loss is output with each iteration and denotes the closeness of the predicting model to the original model. Any value of loss closer to 0 denotes a closeness to the original model we are training against.

```
# Creating our Testing subset
test_data = scaled_data[training - 60:, :]
x_test = []
y_test = dataset[training:, :]
for i in range(60, len(test_data)):
    x_test.append(test_data[i-60:i, 0])

x_test = np.array(x_test)
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))

# Predicting testing data
predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)

# MSE/RMSE - Our evaluation metrics (aside from loss) on how well our model fits
mse = np.mean(((predictions - y_test) ** 2))
print("MSE", mse)
print("RMSE", np.sqrt(mse))
```

Figure 7.1.21 Training and Evaluation Metrics - Training with each iteration is denoted here as well as the evaluation metrics being output for the closeness of fit to our model.

Visualization of Time Series Predictions

The last step of the time series predictive machine learning process is the visualization of the predictive model produced by the training. This model represents the most accurate representation of what the machine learning model believes the next values in the subset to be. As can be seen with the values of loss and the number of epochs iterated, a less accurate model is produced when a lower number of epochs are iterated through. Likewise, a more accurate model is presented when more iterations of training occur. It was found that roughly 500 iterations (epochs) were needed to produce the most accurate model possible with the provided data set.




```
Epoch 4/10
6/6 [==============================] - 0s 9ms/step - loss: 0.0298
Epoch 5/10
6/6 [==============================] - 0s 10ms/step - loss: 0.0275
Epoch 6/10
6/6 [==============================] - 0s 9ms/step - loss: 0.0233
Epoch 7/10
6/6 [==============================] - 0s 10ms/step - loss: 0.0201
Epoch 8/10
6/6 [==============================] - 0s 9ms/step - loss: 0.0202
Epoch 9/10
6/6 [==============================] - 0s 9ms/step - loss: 0.0163
Epoch 10/10
6/6 [==============================] - 0s 9ms/step - loss: 0.0182
MSE 4925.570836059499
RMSE 70.18241115877609
```

Figure 7.1.22 - Model with 10 Epochs - A less accurate model is represented here based on the number of iterations of training the predictive model was able to go through.

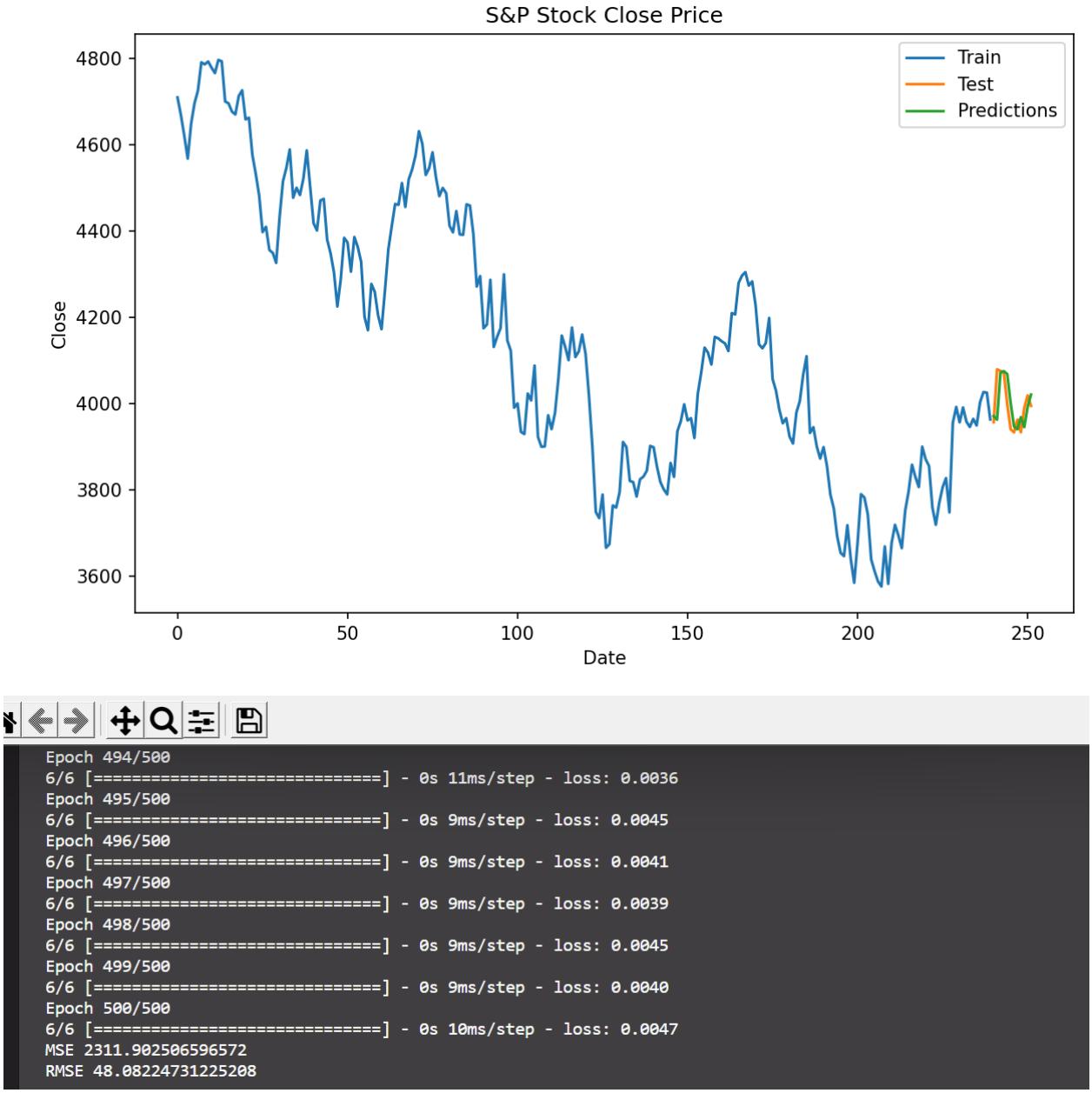


Figure 7.1.23 Model with 500 Epochs - A much more accurate model is represented and produced when the predictive model is given enough iterations of our training subset. It can be seen that the number of iterations greatly affects the prediction accuracy.

Conclusion

The machine learning model approach to economic data analysis works best when given a large enough data set with ample training done on the predictive model. This allows the model to follow the various trends in the market. Comparing a predictive model with a very high loss value (MSE and RMSE closer to 0) and a predictive model with a very low loss value (MSE and

RMSE > 0) displays that a more well-trained model will match the desired visualization of the training set.

7.2 Architecture and Designs

Figures 7.2.1 through 7.2.7 show the original layout of the user interface. Figures 7.2.8 through 7.2.14 show the final layout. We decided to remove some models that didn't perform well and Tensorflow. We wanted to keep our website clean by only providing models that the user could benefit from. Therefore, we decided on a simple website that is easy to access by users.

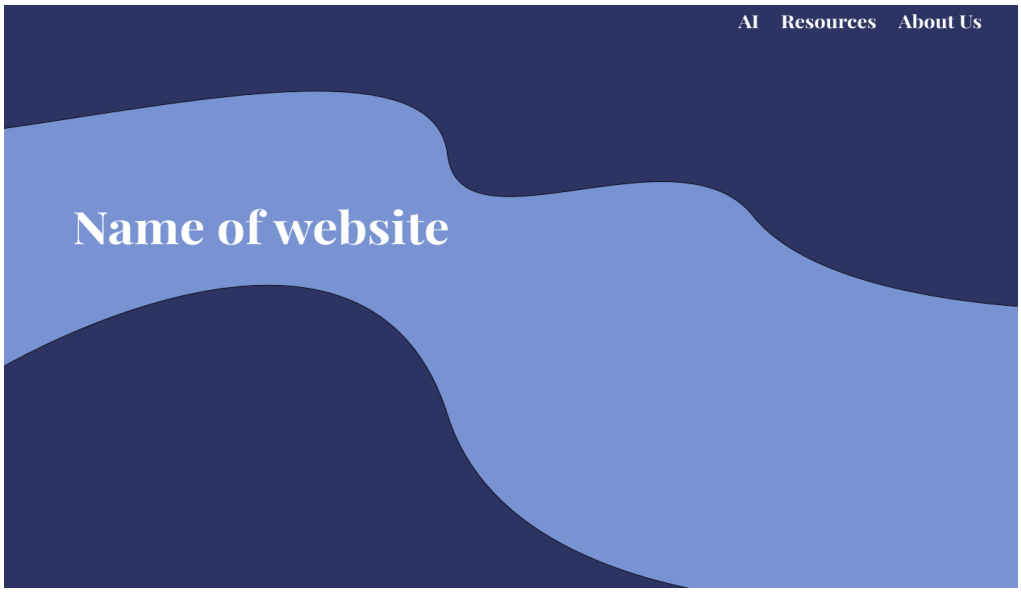


Figure 7.2.1 Homepage - This figure shows the layout for the homepage of our website

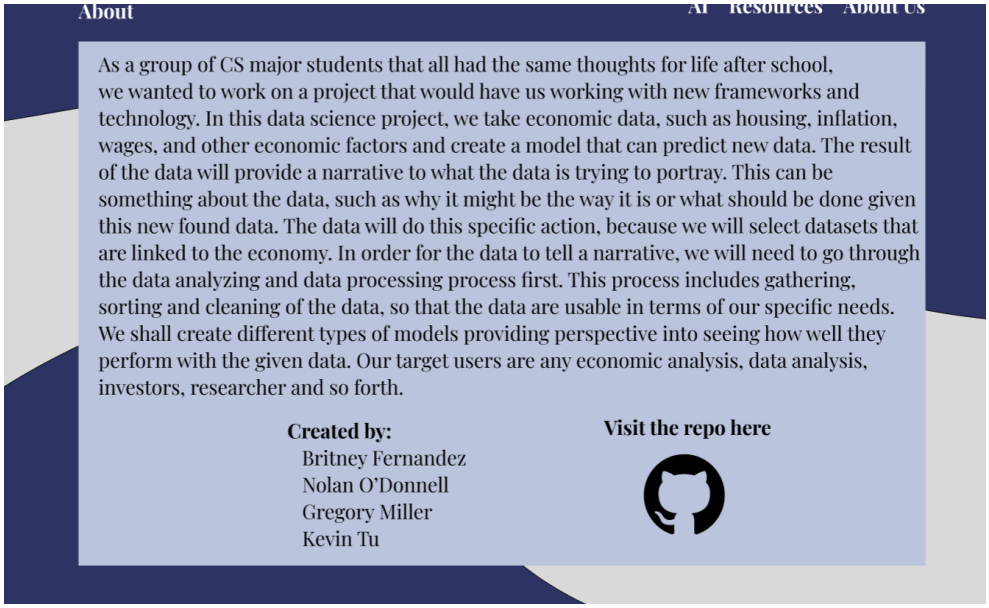


Figure 7.2.2 About Us- This figure shows the layout for the About Us page

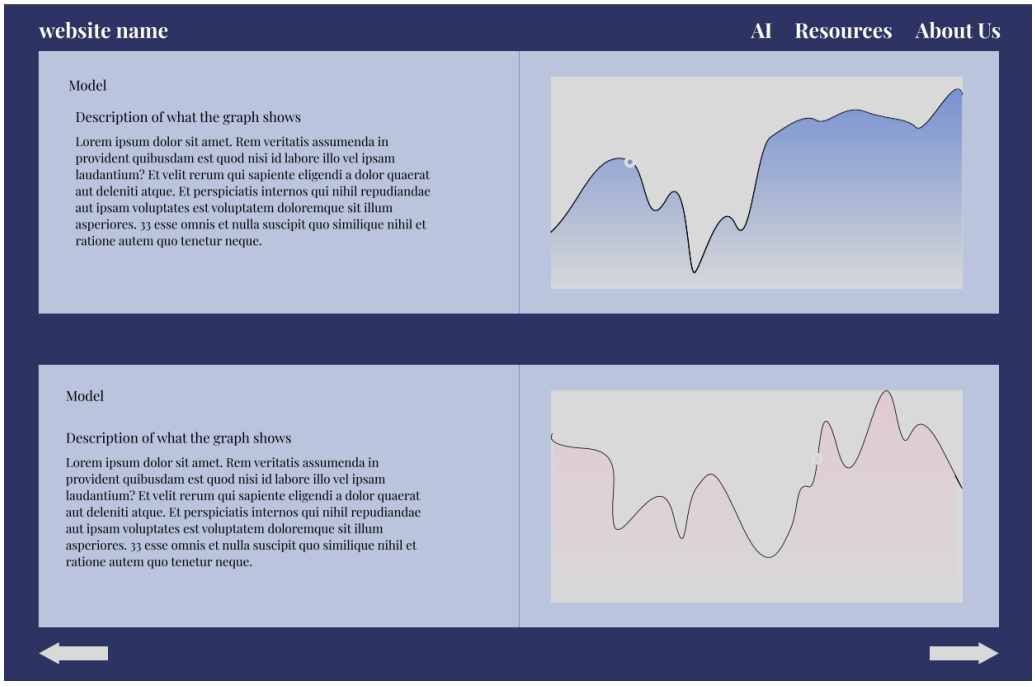


Figure 7.2.3 Linear Graph Layout- This figure shows the layout for the various models we want to display

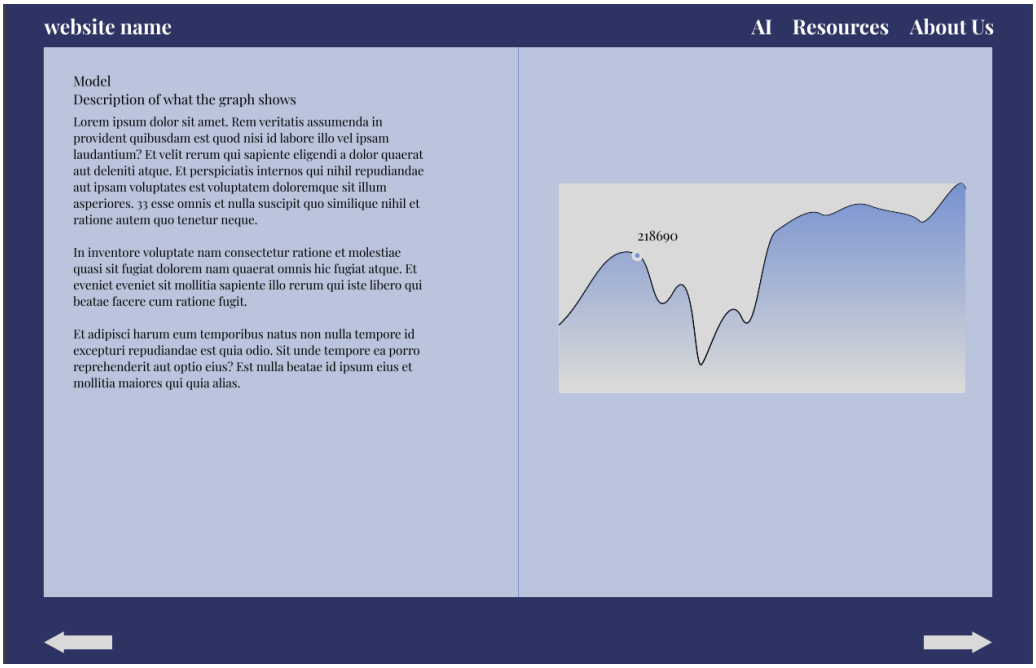


Figure 7.2.4 Description of Model- This figure shows a different layout for displaying our models

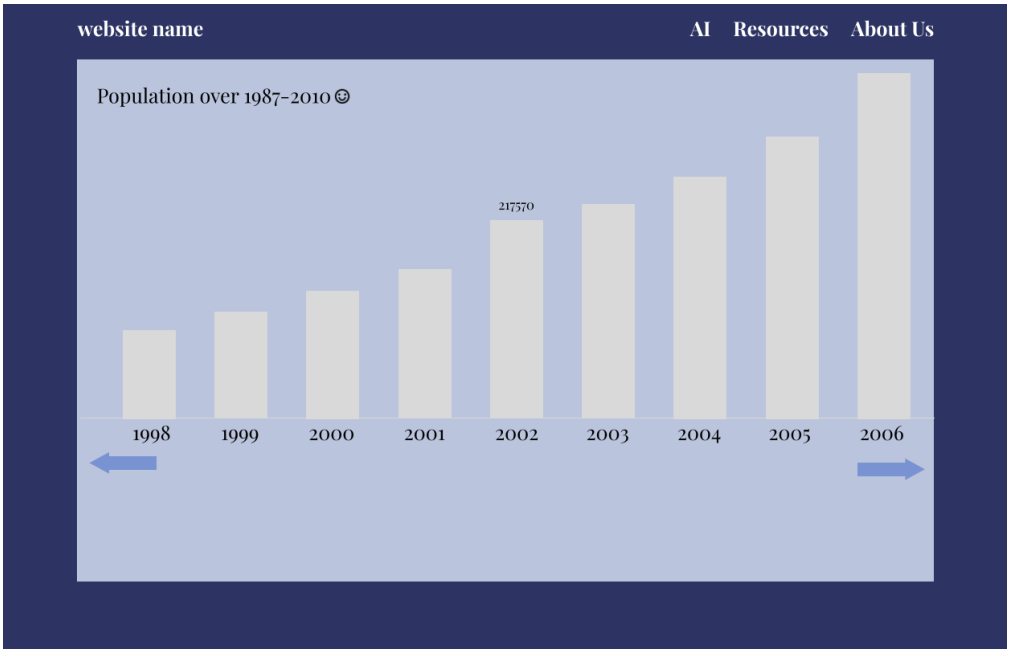


Figure 7.2.5 Bar Graph Layout- This figure shows another potential layout for displaying models

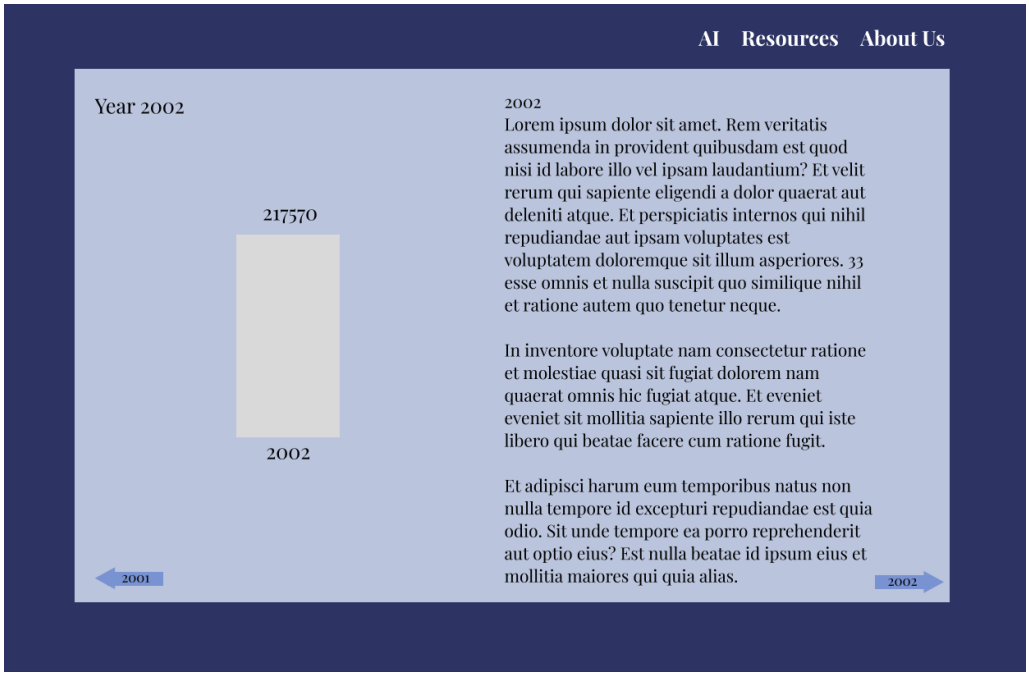


Figure 7.2.6 Individual Bars- This figure shows the layout for one bar if the user wants to learn more about it

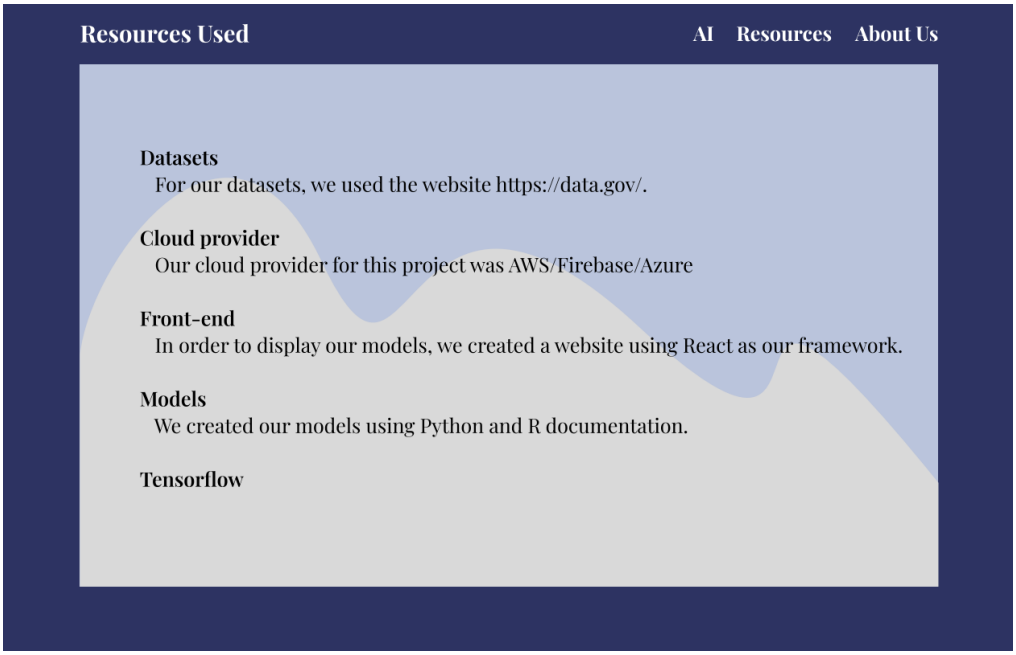


Figure 7.2.7 Resources- This figure shows the layout for the resources used page

Figure 7.2.8 Final About Us- This figure shows the final layout for the About page

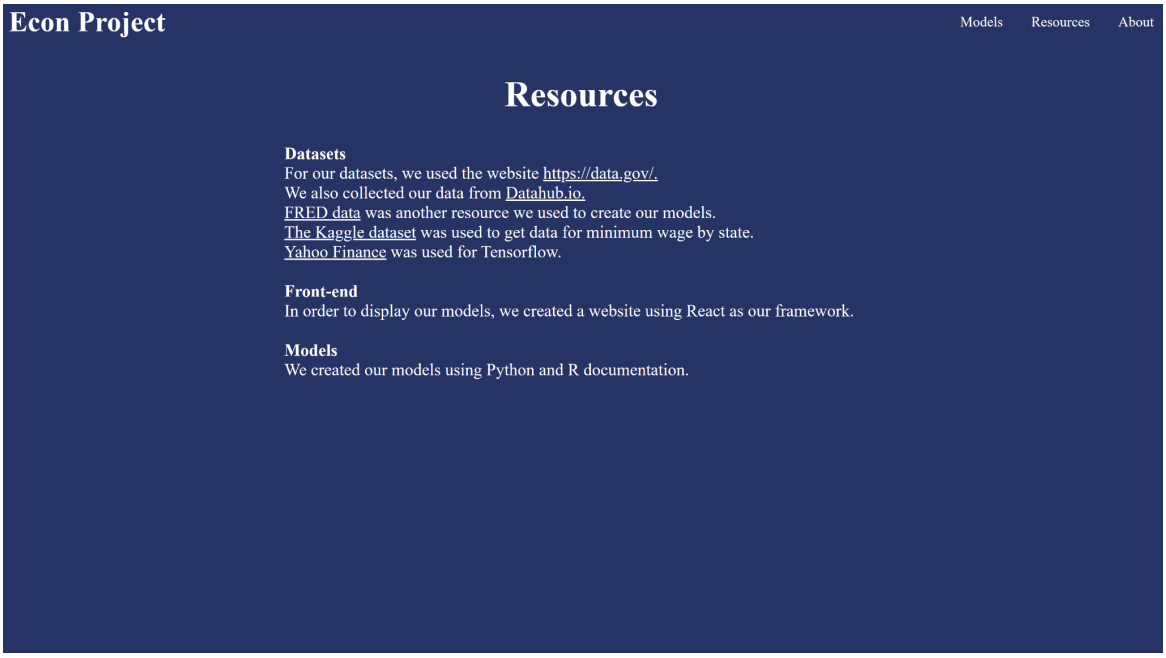


Figure 7.2.9 Final Resources- This figure shows the layout for the resources page

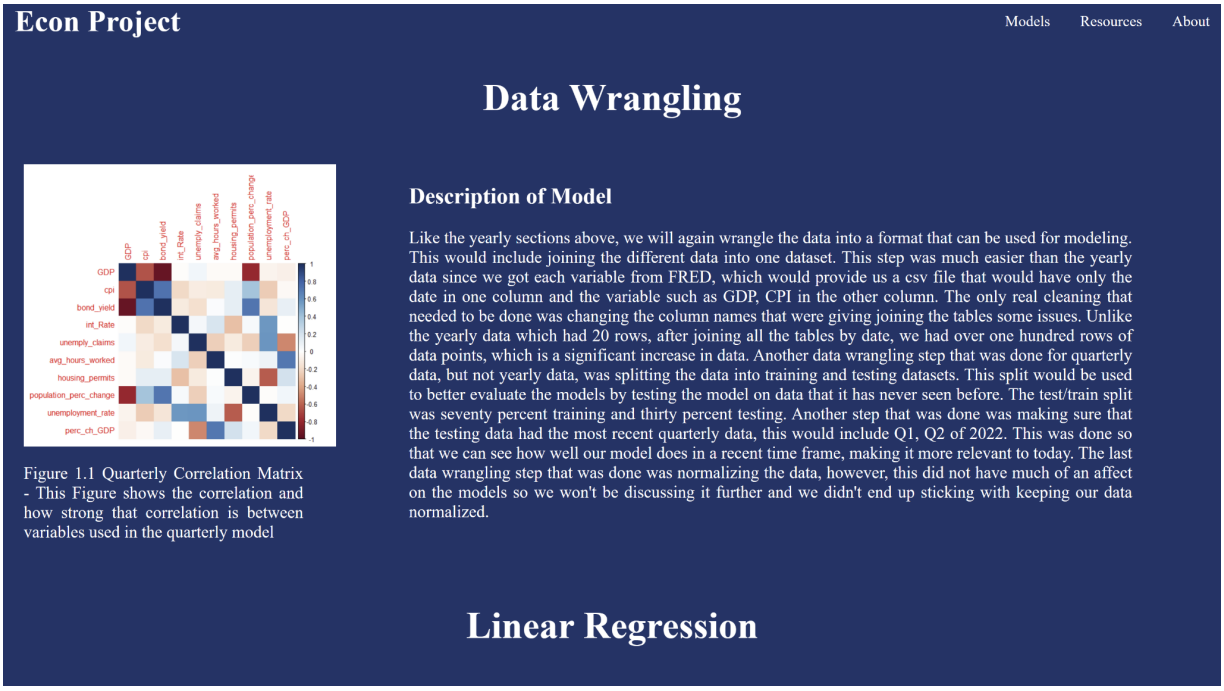


Figure 1.1 Quarterly Correlation Matrix - This Figure shows the correlation and how strong that correlation is between variables used in the quarterly model

Figure 7.2.10 Final Models - This figure shows the final layout for the models page

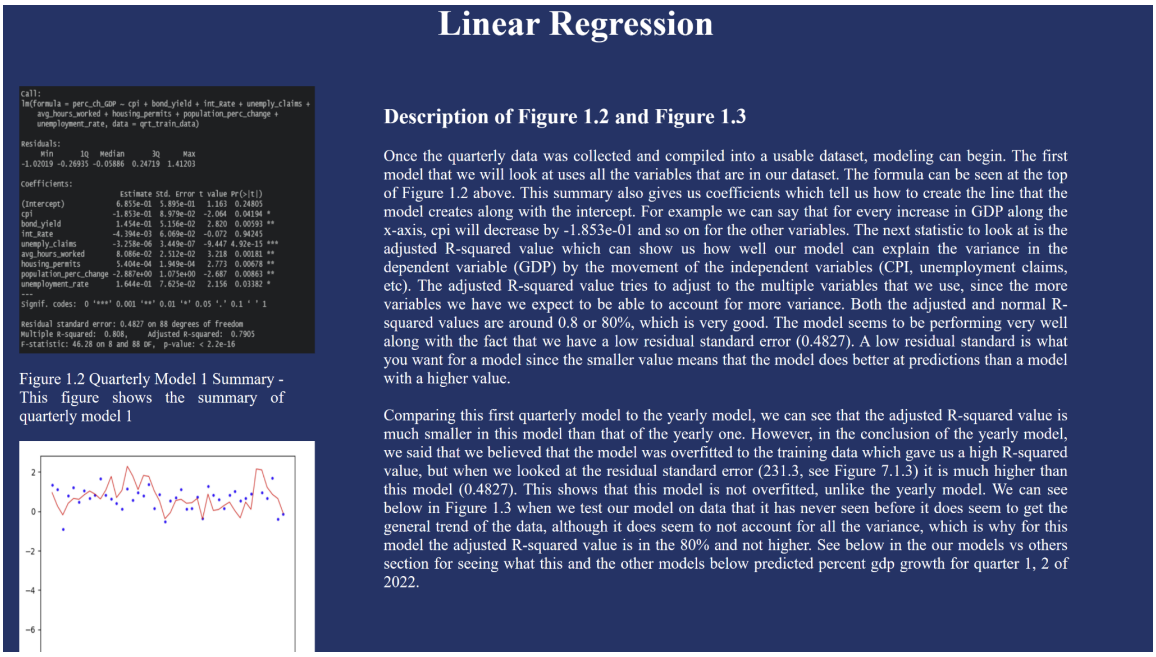
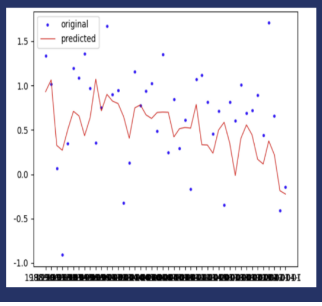


Figure 1.2 Quarterly Model 1 Summary - This figure shows the summary of quarterly model 1

Figure 7.2.11 Final Models- This figure shows the final layout for the models page

```
Call:
lm(formula = perc_ch_GDP ~ cpi + bond_yield + population_perc_change,
    data = qrt_train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-8.3989 -0.2186  0.0404  0.3591  2.0089

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)           0.57431    0.44120   1.302   0.1962
cpi                  -0.39374    0.17308  -2.275   0.0252 *
bond_yield            0.22224    0.09915   2.241   0.0274 *
population_perc_change  0.53316    2.14201   0.249   0.8040
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.015 on 93 degrees of freedom
Multiple R-squared:  0.1031,    Adjusted R-squared:  0.07415
F-statistic: 3.563 on 3 and 93 DF,  p-value: 0.01717
```

Figure 1.4 Quarterly Model 2 Summary - This figure shows the summary of quarterly model 2



Description of Figures 1.4 and Figure 1.5

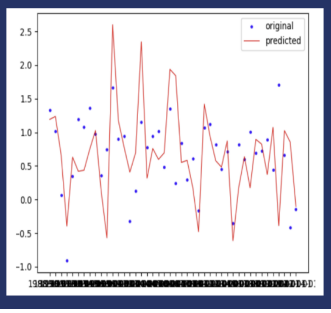
This second model we wanted to see what would happen if we just used the highly correlated variable to predict GDP. The formula can be seen at the top of Figure 1.4 above. This summary also gives us coefficients which tell us how to create the line that the model creates along with the intercept. For example we can say that for every increase in GDP along the x-axis, cpi will decrease by -0.39374 and so on for the other variables. The next statistic to look at is the adjusted R-squared value which is horrible. It means that we cannot account for variance in GDP given our independent variables. This is not something I would think would happen, since these variables are already highly correlated with the dependent variable. Even though the model has a low R-squared value the model has a low residual standard error (1.015). If we recall, a low residual standard is what you want for a model since the smaller value means that the model does better at predictions than a model with a higher value. Given that the adjusted R-squared value is so low, we would not expect the model to predict very well, however, when we looked at the predicted values this model gave us, it was surprisingly accurate at predicting the quarter 1 and 2 of 2022.

When we compare this model to the yearly and quarterly model 1, this model does not look impressive, in fact it looks pretty horrible. It is hard to say if this model is better than the yearly model, since that model is overfitted and has a very high residual standard error, but this model's R-squared value is extremely low. The details of this model honestly is not the interesting part, it's the fact that given the model is so bad, how did it predict so well, which will be discussed below. Looking at Figure 1.5 below we can see that the model does seem to get some of the variance but it hardly performs and it is very noticeable to the eye.

Figure 7.2.12 Final Models- This figure shows the final layout for the models page

```
Call:
lm(formula = perc_ch_GDP ~ int_Rate + unemply_claims + avg_hours_worked +
    housing_permits + unemployment_rate, data = qrt_train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-1.03963 -0.31165 -0.03244  0.25191  1.61215

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)      -2.300e-01  4.988e-01  -0.461 0.645765
int_Rate         -4.529e-02  5.967e-02  -0.759 0.449744
unemply_claims   -3.298e-06  3.320e-07  -9.933 3.52e-16 ***
avg_hours_worked  9.158e-02  2.571e-02   3.562 0.000588 ***
housing_permits   6.961e-04  1.888e-04   3.686 0.000387 ***
unemployment_rate 2.111e-01  7.282e-02   2.907 0.004581 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4995 on 91 degrees of freedom
Multiple R-squared:  0.7874,    Adjusted R-squared:  0.7757
F-statistic:  67.4 on 5 and 91 DF,  p-value: < 2.2e-16
```

Figure 1.6 Quarterly Model 3 Summary - This figure shows the summary of quarterly model 3



Description of Figures 1.6 and Figure 1.7

The third quarterly model that we are showing is one where we wanted to use the least amount of variables, while still performing well, that being having a high R-squared and a low residual standard error. The variables that were removed were the highly correlated variables (darker boxes in Figure 7.1.1), or the variables that we used in quarterly model 2 (Figure 7.4). If we compare this model to quarterly model 1 (Figure 7.2) we see that they are very similar. The residual standard error is very low and a very similar value to that of this model. This means that like the first model, this one should predict value well. The R-squared value is also almost the same, meaning that we account for almost the same variance of GDP given less variables used. This is impressive since we went from 8 variables to 5 variables and are still getting models roughly the same.

When we look at what the model predicts in Figure 1.7, we can see visually that the model does seem to get the general trends like what we see in quarterly model 1. This model may even look better to the eye. We will compare the quarter 1, 2 of 2022 predictions below.

Figure 7.2.13 Final Models- This figure shows the final layout for the models page

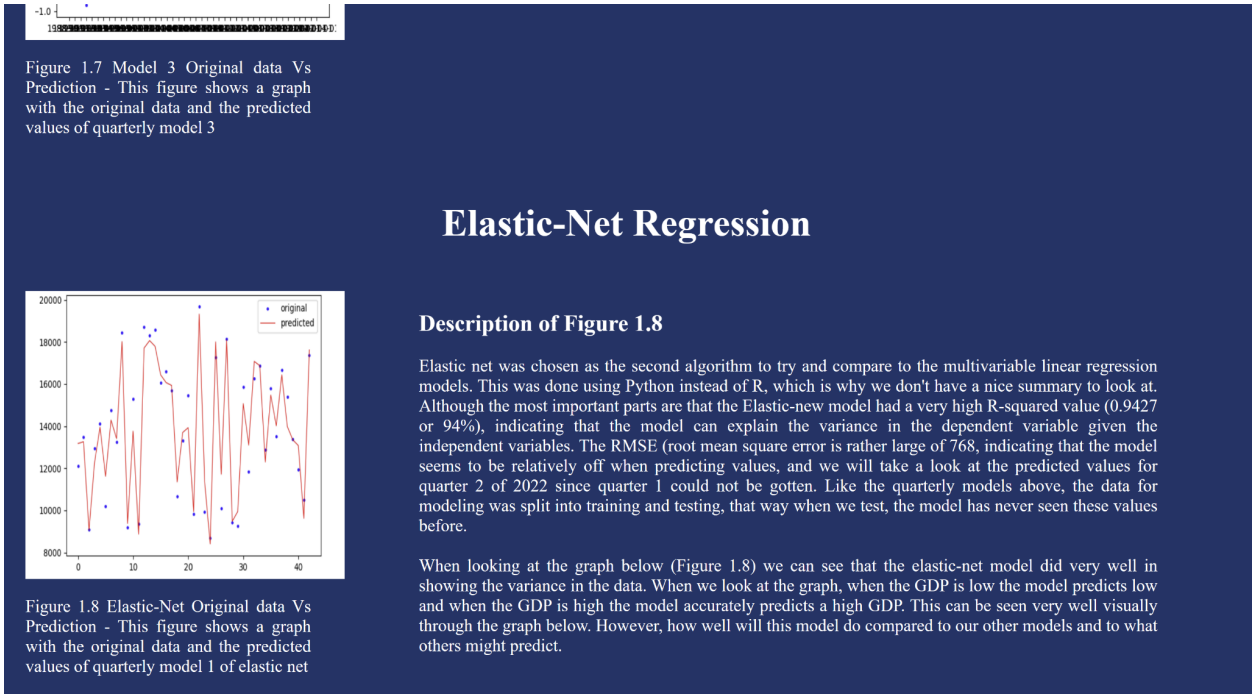


Figure 7.2.14 Final Models- This figure shows the final layout for the models page

7.3 Open Source and 3rd Party Components Used

Anaconda - Used in Python, R data science and machine learning for performing data science. Contain thousands of open-source packages and libraries.

Dplyr - A library in R-studio used for mathematics across columns in a dataset.

GGplot2 - A library in R-studio used for graphing data.

Keras - a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation.

Matplotlib - A library used for data visualization in creating static, animated and interactive charts within the Python programming language.

NumPy - A library used for working with arrays within the Python programming language.

Pandas - A library used in data analysis for data manipulation within the Python programming language.

React.js - An open source JavaScript framework for building interactive user interfaces and web applications.

R-studio - An open source IDE for R. Allows easy data science through its library manager.

Sklearn - An open source Python library used for machine learning and has many other algorithms in the library such as linear regression, Elastic-Net, and many more.

Tensorflow - an open-source platform for machine learning using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them. This flexible architecture allows machine learning algorithms to be described as a graph of connected operations.

Tidyverse - A library in R-studio used for wrangling data as well as many other features that are not used in the project.

Corrplot - A library in R-studio used for creating a correlation matrix

7.4 Implementation Methods

The models that have been created were using Multivariable Linear regression or Elastic-net algorithms. The yearly and quarterly linear regression models were created using R and R-Studio's libraries. The Elastic-Net model was created in Python using Sklearn's library. Residual standard error and Adjusted R squared values are used to determine the fit of the models.

7.5 Test Plans and Results

The part of this section that was here during the midterm, was moved to section 7.1 along with the other results for other models since it just made more sense to keep everything together, but added additional conclusions in section 8 conclusions.

7.6 Installation or Setup Guides

Anaconda - Install Anaconda on Windows 11, 64 bit, through the "Anaconda3-2022.05-Windows-x86_64.exe". Open Anaconda command prompt and run the

following command: `conda create -n myenv python=3.9 pandas jupyter seaborn scikit-learn keras tensorflow` to create an environment named **myenv**. Using Visual Studio Code as the notebook. Select **Kernel** at the top right of the notebook in Visual Studio Code. Choose the Python environment you created above this will be "**myenv**" in which to run your kernel.

Python - Python 3.10.8 installation on Windows 11, 64 bit, through the "python-3.10.8-amd64.exe".

Keras & Tensorflow - Both, Keras and Tensorflow are installed at the same time when installing the Tensorflow 2.0 package with Python. In Python, check to see that your version is up to date: `python3 --version` then ensure the latest version of pip is installed: `pip install --upgrade pip`. Following this, run the following command: `pip install tensorflow` and both packages for Keras and Tensorflow will be installed.

React - Download and install `Node`. Then run the command '`npx create-react-app my-app`' in the folder you want. Then go to your app by running '`cd my-app`'. Once you're in the right directory, you can run '`npm start`'.

R - Install R 3.3.0 by going to cran.rstudio.com and downloading for your OS

R-Studio - Install Rstudio Desktop for a free open source IDE at Rstudio.com

Visual Studio Code - Install Visual Studio Code, through the "VSCodeUserSetup-x64-1.72.1.exe". Go to the extension and install the following: **Python extension** , **Pylance extension**, **Jupyter extension.**

8. Conclusion

For the yearly linear regression section, the conclusions are the same as the midterm. That being that the models are overfitted. When looking at the above conclusions we can see that the high R-squared value and high residual standard error are very high, indicating an overfitted model. We aimed to solve this with our quarterly models, where we gave the model more data to train on, and we split the data into training and testing. This allowed us to get a better understanding on how well the model was doing. From the conclusion section for the quarterly linear regression section, we see that it seemed to work. The r-squared value is still very high, although not as high and the residual standard error is very low. So, we can conclude that the increase in data, as well as splitting the data into training and testing seemed to create models that were less overfitted. Another conclusion for our models is that the new quarterly models did very well when predicting quarter 1 and quarter 2 of 2022. While our linear regression models seemed to outperform our Elastic-Net model, the Elastic-Net model was still relatively accurate and accurately predicted a negative GDP growth. We showed how well our models did compared to other institutions in the section ‘our models vs others’ which again shows that our model did very well. (See conclusion sections in yearly models, quarterly models, and Elastic-Net model for more details on conclusions) Another conclusion that we can make from the research so far, is that there is always more research to be done, we can still continue the work to better predict GDP. However, we don't have infinite time so our current conclusions will have to do for now.

9. References

1. Datopian. (n.d.). *Economic Data*. DataHub. Retrieved December 13, 2022, from https://datahub.io/collections/economic-data
2. Federal Reserve Bank of St. Louis. (n.d.). *Federal Reserve Economic Data: Fred: St. louis fed*. FRED. Retrieved December 13, 2022, from https://fred.stlouisfed.org/
3. *Forex News, Technical Analysis & Trading Tools*. Forexlive. (n.d.). Retrieved December 13, 2022, from https://www.forexlive.com/
4. *Let's build from here*. GitHub. (n.d.). Retrieved December 13, 2022, from https://github.com/
5. Lislejoem. (2020, December 31). *US minimum wage by state from 1968 to 2020*. Kaggle. Retrieved December 13, 2022, from https://www.kaggle.com/datasets/lislejoem/us-minimum-wage-by-state-from-1968-to-2017
6. Microsoft. (2021, November 3). *Visual studio code - code editing. redefined*. RSS. Retrieved December 13, 2022, from https://code.visualstudio.com/
7. *Pandas*. pandas. (n.d.). Retrieved December 13, 2022, from https://pandas.pydata.org/
8. Posit. (2022, November 14). Retrieved December 13, 2022, from https://www.rstudio.com/
9. *React – a JavaScript library for building user interfaces*. – A JavaScript library for building user interfaces. (n.d.). Retrieved December 13, 2022, from https://reactjs.org/
10. *Tensorflow*. TensorFlow. (n.d.). Retrieved December 13, 2022, from https://www.tensorflow.org/
11. *Trello brings all your tasks, teammates, and tools together*. Trello. (n.d.). Retrieved December 13, 2022, from https://trello.com/
12. U.S. Bureau of Labor Statistics. (n.d.). *Databases, tables & calculators by subject*. U.S. Bureau of Labor Statistics. Retrieved December 14, 2022, from https://www.bls.gov/data/#employment
13. U.S. Bureau of Labor Statistics. (n.d.). *Databases, tables & calculators by subject*. U.S. Bureau of Labor Statistics. Retrieved December 14, 2022, from https://www.bls.gov/data/#prices

14. Usdatagov @usdatagov, & Usdatagov. (2020, September 14). Data.gov. Retrieved December 13, 2022, from https://data.gov/
15. *Visualization with python*. Matplotlib. (n.d.). Retrieved December 13, 2022, from https://matplotlib.org/
16. *Welcome to Python.org*. Python.org. (n.d.). Retrieved December 11, 2022, from https://www.python.org/
17. *The world's most popular data science platform*. Anaconda. (n.d.). Retrieved December 13, 2022, from https://www.anaconda.com/
18. Yahoo! (2022, December 15). *S&P 500 (^GSPC) historical data*. Yahoo! Finance. Retrieved December 14, 2022, from https://finance.yahoo.com/quote/%5EGSPC/history?period1=1639543446&period2=1671079446&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true&guccounter=1