ECONOMICS OF COVID-19 LOCKDOWNS:

Optimizing the Lockdown Health-Economy Tradeoff

Team Woahhhh
David Fan, Isabelle Tao, Lucy Cao, Emma Lu

WHARTON HACKATHON
COVID AND THE ECONOMY

DAVID FAN • Manila, Philippines

David is studying towards both a MSE in Data Science and a BS in Economics. As an aspiring data scientist, David has interned at Facebook and Grab in Data Science roles over the past summers. He also heads the Analytics department at the Daily Pennsylvanian. David enjoys Chinese rap music, sitcoms, and NBA statistics.

ISABELLE TAO • Singapore, Singapore

Isabelle is a senior concentrating in Business Analytics at the University of Pennsylvania. She interned at McKinsey & Company as a Business Analyst in the digital practice the past summer, where she enjoyed working on digital transformation for large companies in the consumer electronics industry. She also enjoys brewing Kombucha and watercoloring.

LUCY CAO • Shanghai, China

Lucy is a senior at the University of Pennsylvania majoring in Cognitive Science and Computer Science. This past summer, Lucy worked at McKinsey & Company as a Summer Analytics Fellow transforming businesses' decision-making with advanced analytics. In her free time, she likes to make song covers and visit escape the rooms.

EMMA LU • Vancouver, Canada

Emma is a senior in the Roy & Diana Vagelos Life Sciences and Management program at the University of Pennsylvania. She is majoring in Computational Biology, Finance, and Statistics. Over the summer, Emma worked at Bain & Company as a Associate Consultant Intern advising clients on investment decisions. Her hobbies include reading biographies, hiking, and gardening.

BACKGROUND

The Covid-19 pandemic has forced many countries to use lockdowns as a public health measure to prevent further spread of the disease, often at the expense of slowed economic activities. The lockdown-induced trade-off between economic and health outcomes has underscored the importance to evaluate the **effectiveness of lockdowns**.

We focus on the US economy given its leading world count in Covid-19 cases and its economy's influence on the global economy. In addition, US states have experienced varying levels of lockdown success, allowing for further investigation. We evaluated the health and economic outcomes of different US state lockdown policies that vary in **duration and stringency**, adjusted based on the states' characteristics, to determine the optimal lockdown policies that would maximize both health and economic outcomes.

METHODOLOGY

To understand the effects of lockdown, we first created created an index that tracks multiple health and economic indicators for each of the 50 states when the lockdown policies were imposed. This was done by first standardizing these metrics and feeding them through a Principal Component Analysis (PCA).

Then we selected control variables (Population Density, Population Size, Political Leaning, and Share of Population above 65 years old) that might also play a part on lockdown outcomes independent of government intervention. Finally, lever variables (Duration and Stringency of lockdown) were selected for their direct relationship to the characteristic of the lockdown.

We further analyzed the relative variable importance between the nuisance and lever variables, generated Partial Dependence Plots to understand each lever's marginal effects, and conducted a case study to understand the synergies between potential government interventions.

KEY OBSERVATIONS

1. **Lockdown length is more important than lockdown stringency to contain the virus.**
   Our analysis shows that the longer the length of the lockdown, the more effective the lockdown is. Stringency on the other hand, has an inverse effect on the health index. This means that states with more stringent lockdowns actually promotes more rebellious behavior which causes more deaths, hospitalizations and spikes in cases.

2. **Lockdown length and stringency are both not strongly correlated with decline in GDP and increase in unemployment.**
   While it is common to assume that the longer the lockdown, the worse the length of the state of the economy, our analysis shows that that is not the case. Given alternative consumption methods (online shopping) and alternative working options (work from home), consumption and productivity can still be sustained. This is aligned with results globally: the actual or expected drop in GDP, across OECD countries is not as strongly correlated with lockdown lengths or stringency. (McKinsey Analytics)

3. **The most effective lockdown duration is between 55 and 60 days.**
   We found that there is a golden period where lockdowns are the most effective. When the lockdown is below 55 days, it's insufficient to cause a decline in cases. When the lockdown is above 60 days, there is essentially no effect for both health index and economic index.

NEXT STEPS

First, we hope to incorporate more **granular county level data**, so we can add in more control variables and obtain results that are of higher statistical significance. Second, we hope to **extend these results globally** to check if our observations apply to global situations.

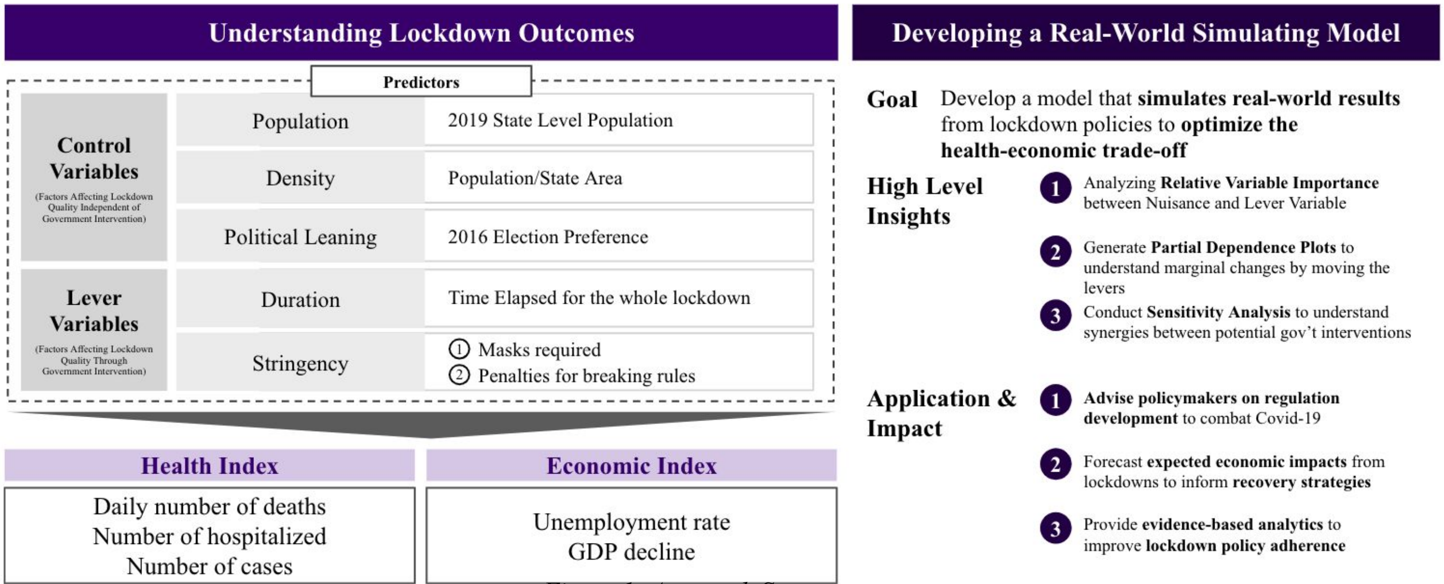


Figure 1: Approach Summary

Introduction

Covid-19 lockdowns have been implemented around the world for public health reasons. While lockdowns are undoubtedly an **effective public health measure**, they also **limit economic activities** and **negatively affect economic growth**. For example, the US, leading the world charts with over 7 million Covid-19 cases, had a GDP fall of 32.9% (annualized rate) in Q2 2020, the lowest since 1947.

Policymakers are faced with the challenge of **balancing the health and economic trade-off**. If a lockdown is lifted too quickly, it could cause a re-surge in cases, resulting in more lockdowns. Alternatively, a lockdown that is too long could cause detriments to the economy that will take years to recover. Our knowledge of the lockdown's effectiveness is limited and there are few historical data references. With over six months of current data and different states employing different policies, it is possible to **empirically assess the outcomes** from these lockdowns to derive additional understandings of the **optimal trade-off**.

We aim to **create a model that stimulates real-world reactions at the state-level** towards different lockdown policies. The model will allow policymakers to **forecast lockdown effectiveness and economic impacts**. Our model can also be used as an **evidence-based argument to improve policy adherence**.

Analytical Approach

While Covid-19 lockdowns have garnered much interest from health and economic experts, there still remain many gaps in the literature of assessing the effect of lockdown measure that we aim to investigate.

Index Construction

First, we needed a metric that will allow us to measure the **health and economic outcomes of lockdowns**. Since outcomes can be measured using many indicators, we created **two indices that combined relevant variables** to track the health and economic outcomes of a state's lockdown policy.

These indicators were created through **factor analysis** where we utilized the top Principal Component across individual indicators. This served 2 main purposes; first, we want to be able to **isolate the underlying latent state of either health or economics that causes the observables** as opposed to relying on the observed metrics themselves. This is because metrics observed (death, cases in the health cases, or GDP and unemployment in the econ cases) are subjected to some degree of randomness and may therefore individually exhibit variation that would add noise to our data. Secondly, the creation of indexes lessens additional model we need to run in order to incorporate various health outcomes, drastically simplifying the process.

Health Index

The Health Index is created to access the coronavirus cases in states during the lockdown. The higher the absolute number of the health index, the worse the performance of the lockdown. Given that this measure is a gradient, we opted to focus on the percentage decline of 3 key health attributes: **daily number of deaths, number of hospitalized, and number of cases.**

We obtained the decline rate of each of these 3 health attributes during the lockdown using the following method. First, we obtained the **highest number for each of these 3 metrics** during the lockdown. Next, we extracted these **3 metrics on the last day** of the lockdown. Lastly, we divided the final day metric by the maximum metric to get the gradient change during the period.

$$Gradient = \frac{Final\ Day\ Number}{Maximum\ Number}$$

In simple terms, the **higher the gradient change ratio, the less effective the lockdown is,** because the lockdown did not improve the health metric as expected. If the ratio is low, it indicates that the lockdown is effective in lowering the cases from the peak.

As indicated earlier, we wanted to combine these 3 high level metrics into one overall indicator. **This was done by first standardizing these metrics and feeding them through a Principal Component Analysis (PCA)**. As expected, the leading principal component was able to explain **52% of the variation** in these metrics, making it a fair representation of the underlying health traits. The loading score of the aforementioned indicator are all around 0.5, indicating that a one unit increase in health index correspond to half a standard deviation increase gradient change, pointing to a less effective lockdown.

Economic Index

Similar to the health index, the economic index is created to gauge the overall decline in state economic condition.

This measure was done with the use of gradient change for 2 metrics: GDP decline from 2019 Q4 to 2020 Q1, and unemployment rate increase from February 2020 to April 2020.

Since both metrics were already in their natural percentage format, rescaling is no longer necessary. We simply performed our factor analysis using **PCA on both these gradients.**

The leading component using the PCA was able to explain 71% of the total variation in the gradient once again, making a viable candidate to represent the underlying economic drivers. The loading vectors for GDP change is -0.7 and 0.7 for unemployment change. This means as one unit of economic index increases, we would expect the **GDP to decrease by 0.7%** while **unemployment rate to increase by 0.7%.**

Control Variables and Lever Variables

After constructing the indexes needed for our target variable, we now move on to create the left hand side of our equation, or the x-variables.

When considering our x-variables, we looked at **variables that may affect our aforementioned indexes independent of any kind of intervention that the government attempts.** We refer to these variables as our **control variables.** While it may be ideal to include as many control variables as possible to create impartial results, since we are using state level dataset with limited amount of observations (50 states at most), to avoid the curse of dimensionality problem, **we opted to only include 4 main control variables: Population Density, Population, Political Leaning and Share of Population above 65 years old.** These variables are selected due to how they may directly affect the indexes at hand without the Gov't intervention.

For our lever variables, we selected two main characteristic related lockdown: **the length of the lockdown and the relative stringency of lockdown** which is anchored on two characteristics: 1) Whether the state required masks and 2) whether the state implemented a penalty for violating the rules

Initial Model Performance
We performed an initial linear regression model to assess the relationship between lockdown length and the health index. We found that the linear regression model gave us an **r-squared of 0.279** and we also found **the lockdown length variable to be statistically significant** with a p-value of 0.035. Similarly, we fitted a second linear regression model to **evaluate the relationship between lockdown length and the economic index.** We found that this linear regression model had an r-squared of 0.262. In this model, we found that **the republican feature is statistically significant with a p-value of 0.020.**

Improved Model Performance
We then performed a **random forest model to account for potentially non-linear relationships between the variables and the indices** as well as increase predictive power of our modeling. Moreover, through partial dependence plots we can better understand the marginal effect of the length and stringency of lockdown on economic and health outcomes. We first performed a 20% split on the data set, with 80% of the data in the training set and 20% of the data in the test set.

This model gave us a **better predictive ability overall for our dataset.** This includes a 0.9 r-squared for our training dataset and a 0.4 r-squared for our test dataset when predicting the health indexes and 0.75 r-squared for our training dataset and a 033 r-squared for our test dataset when predicting the econ indexes. We then set out to draw additional inference from the model.

Variable Relative Importance
First we aim to analyze the variable importance information within our two models. Starting off with the health index model.

Weight	Feature
0.5357 ± 0.4113	Lockdown Length
0.0115 ± 0.2518	Population 2019
0.0106 ± 0.0108	Added Levels
0.0096 ± 0.0161	Republican
-0.0015 ± 0.0055	Democrat
-0.0565 ± 0.1245	Density
-0.1444 ± 0.1540	Share_65

Table 1: Feature Importance of Health Index

It is apparent from the variable importance plot that **the lockdown length is by far the most important variable** in our dataset superseding even the control variables that we have included. **This generally is in line with our hypothesis that the length of lockdown will very likely benefit the states in terms of containing the spread of the virus.**

Stringency of lockdowns, on the other hand, represented by the added levels variable, **ranks third in importance,** indicating that it does somewhat still have an effect on the indexes but just not as apparent as the length itself. This can be an artifact of the majority perception of lockdown such that most individuals are likely to abide by the rules regardless of stringency

For the econ model on the other hand,

Weight	Feature
-0.0026 ± 0.0118	Added Levels
-0.0053 ± 0.0516	Republican
-0.0567 ± 0.1146	Population 2019
-0.0960 ± 0.5125	Density
-0.1043 ± 0.0286	Democrat
-0.2388 ± 0.3275	Share_65
-0.4528 ± 1.1405	Lockdown Length

Table 2: Feature Importance for Economic Index

the variable importance is rather interesting. No individual variable stood out too strongly in terms of how irreplaceable it is in our model. It is especially quite interesting to see that the Lockdown Length actually did not seem to impact the econ index at all from a variable importance point of view. These results regardless should be taken with a grain of salt given the huge variation around the weight of these variables. Nonetheless, an insight from this point of view is that the economic downfall during COVID may not necessarily be as related to the lockdown given the rise of alternative consumption methods and alternative work opportunities.

Partial Dependence Plots
We now want to take a deep-dive into the health index model and examine the two lever variables of interest that we have identified: Lockdown length and stringency. This is specifically done for the health index case as it is in there that both lockdown length and added levels were most significant

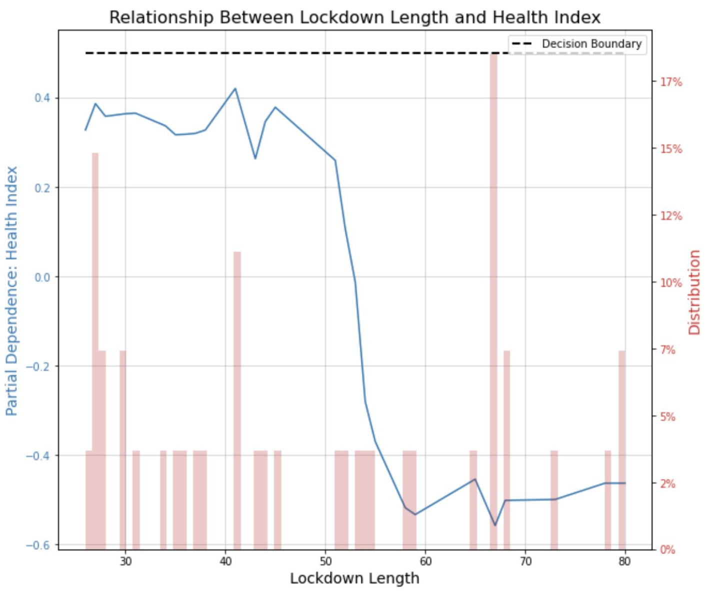


Figure 2: Partial dependence (lockdown length)

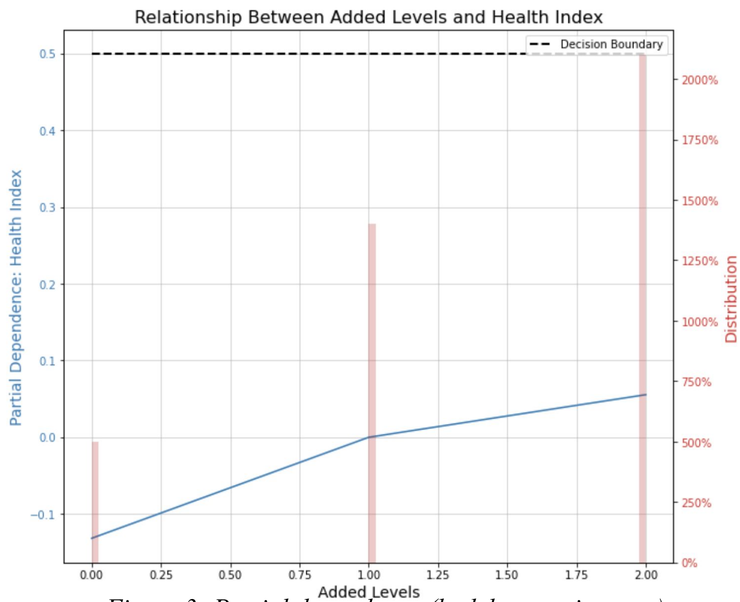


Figure 3: Partial dependence (lockdown stringency)

The deep dive into the partial dependence plots shed light on something extremely interesting. As expected, the p**artial dependence plots for the lockdown periods follows a negative relationship with the health index** (i.e., as lockdown length increases, we see a greater reduction in cases from the peak). The effect is actually not fully continuous but t**here is a sharp increase in effectiveness of lockdown at around 55-60 days and later remains flat.** While not conclusive, this gives us an idea to the ideal lockdown period.

Another interesting insight that emerged is that **lockdown stringency actually may trigger an inverse reaction** that governments do not expect. Specifically, we saw that as stringency increases, the health index actually rose gradually, indicating a less effective lockdown. This is likely due to individuals feeling too suppressed and constrained by the lockdown and end up not abiding by the lockdown rules altogether.

Case Studies

In order to better understand the effect of lockdown length on the health and economic outcomes, we decided to look more closely at how states health and economic indices change when lockdown lengths are altered. For Texas, with an original 30-day lockdown, we predicted the health index to be 0.82. However, when we extend this lockdown period to 60 days, the health index decreases to -0.61 and when we further extend this lockdown period to 90 days, the health index decreases to -0.84. **This result is consistent with our finding that longer lockdowns lead to better containment of the disease and better health outcomes.** We also took a look at the economic index and found that adjusting the lockdown period does not drastically affect the economic index. For the state of Alabama, we notice that the economic index with the original 26-day lockdown is predicted to be -1.43 while an extension of the lockdown to 40 days gives us an economic index of -0.21 and an extension to 60 days gives us a new prediction of -0.05.

Conclusion & Next Steps

This study from a theoretical level showed that **lockdown length, stringency and efficiency is not a purely additive function**. Lockdown length and stringency does not have a positive linear function with improved health outcomes. Instead, the the best approach to achieve an efficient lockdown is often a **combination the right length with a lesser emphasis on stringency.** Furthermore we also explored and realized that the lockdown length and stringency does not drastically affect the economic status of states due to the rise of other opportunities.

In the future, we wish to extend this study to a county but also a global level in order to incorporate more control variables but also allow us to create statistical models with more confidence from more observations.

Data Sets:

1. **Population.csv:** Population of each state (https://www.census.gov/data/datasets/time-series/demo/popest/2010s-state-total.html)
2. **Area.csv:** Area of each state (https://www.kaggle.com/giodev11/usstates-dataset?select=state-areas.csv)
3. **Deaths.csv:** Number of deaths during the lockdown (https://covidtracking.com/data)
4. **Hospitalized.csv:** Number of hospitalizations during the lockdown (https://covidtracking.com/data)
5. **Cases.csv:** Total number of cases during the lockdown (https://covidtracking.com/data)
6. **Unemployment.csv:** Unemployment rate of each state from March 2020 - July 2020 (https://carsey.unh.edu/COVID-19-Economic-Impact-By-State)
7. **GDP.csv:** GDP change from each state from Q4 2019 to Q1 2020 (https://www.bea.gov/data/gdp/gdp-state)

References:

McKinsey & Company (2020), *More stringent lockdowns aren't necessarily worse for GDP* https://www.mckinsey.com/industries/healthcare-systems-and-services/our-insights/covid-19-saving-thousands-of-lives-and-trillions-in-livelihoods

Code:

1. **Python** and **Jupyter Notebook** were primarily used for data wrangling, EDA, and preliminary visualization. We used standard libraries such as **pandas, numpy, matplotlib, seaborn, scipy,** etc.

Appendix 1: Model Specifications

Dep. Variable:	y	**R-squared:**	0.291
Model:	OLS	**Adj. R-squared:**	0.163
Method:	Least Squares	**F-statistic:**	2.263
Date:	Fri, 25 Sep 2020	**Prob (F-statistic):**	0.0614
Time:	16:13:37	**Log-Likelihood:**	-59.017
No. Observations:	40	**AIC:**	132.0
Df Residuals:	33	**BIC:**	143.9
Df Model:	6		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Population 2019	3.654e-08	3.23e-08	1.130	0.266	-2.92e-08	1.02e-07
Lockdown Length	-0.0331	0.015	-2.167	0.038	-0.064	-0.002
Density	-0.0001	0.000	-0.818	0.419	-0.000	0.000
Democrat	2.3640	1.616	1.463	0.153	-0.924	5.652
Republican	2.3586	1.499	1.574	0.125	-0.691	5.408
Added Levels	0.2993	0.302	0.992	0.328	-0.314	0.913
Share_65	-8.4979	9.139	-0.930	0.359	-27.091	10.095

Omnibus:	0.684	**Durbin-Watson:**	1.989
Prob(Omnibus):	0.710	**Jarque-Bera (JB):**	0.706
Skew:	0.010	**Prob(JB):**	0.703
Kurtosis:	2.350	**Cond. No.**	4.48e+08

Figure 4: Linear Regression Health Index

Dep. Variable:	y	**R-squared:**	0.262
Model:	OLS	**Adj. R-squared:**	0.127
Method:	Least Squares	**F-statistic:**	1.949
Date:	Fri, 25 Sep 2020	**Prob (F-statistic):**	0.102
Time:	14:40:10	**Log-Likelihood:**	-72.455
No. Observations:	40	**AIC:**	158.9
Df Residuals:	33	**BIC:**	170.7
Df Model:	6		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
Population 2019	-3.932e-09	4.52e-08	-0.087	0.931	-9.59e-08	8.81e-08
Lockdown Length	0.0223	0.021	1.043	0.304	-0.021	0.066
Density	-0.0001	0.000	-0.635	0.530	-0.001	0.000
Democrat	-4.5396	2.261	-2.008	0.053	-9.140	0.061
Republican	-5.1084	2.097	-2.436	0.020	-9.375	-0.841
Added Levels	-0.1435	0.422	-0.340	0.736	-1.002	0.715
Share_65	26.7997	12.788	2.096	0.044	0.783	52.816

Omnibus:	13.454	**Durbin-Watson:**	1.864
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	23.573
Skew:	0.788	**Prob(JB):**	7.61e-06
Kurtosis:	6.415	**Cond. No.**	4.48e+08

Figure 5: Linear Regression Econ Index

Appendix 2:

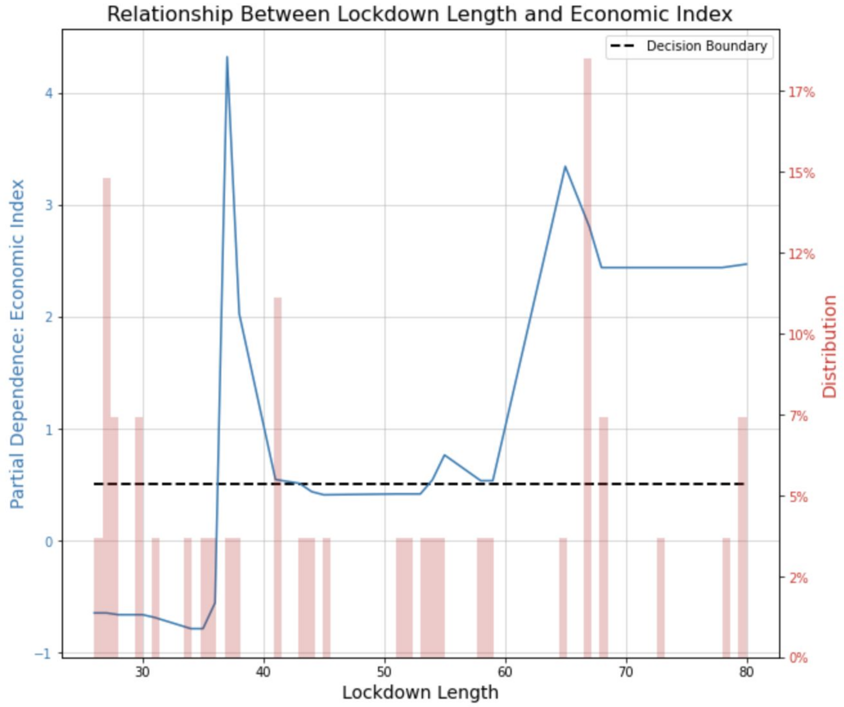


Figure 6

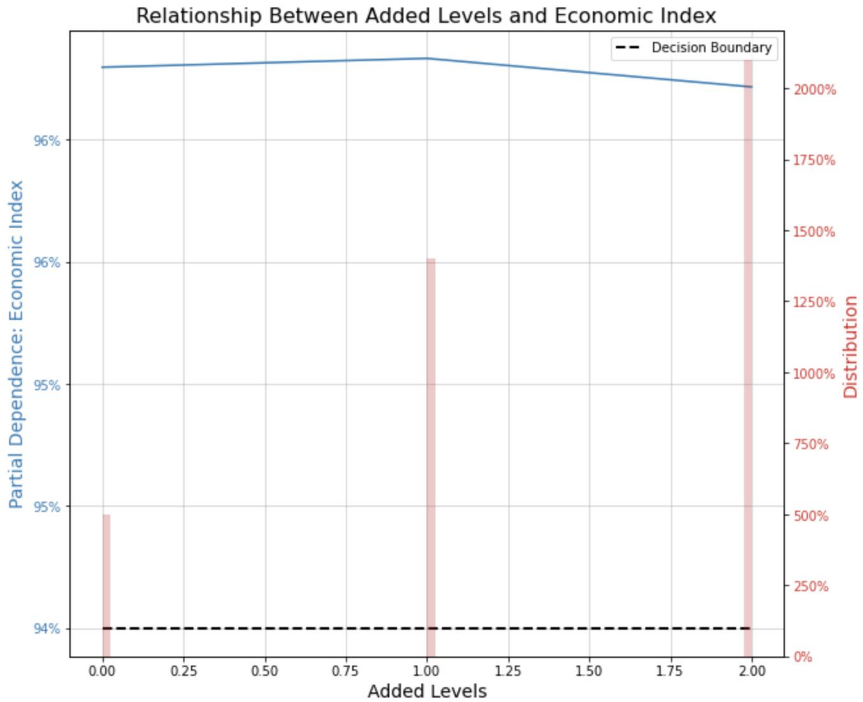


Figure 7

In [1]:
```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np

from bs4 import BeautifulSoup
import requests
from selenium.webdriver import Chrome
from selenium.webdriver.support.select import Select
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support.expected_conditions import visibility_o
f_element_located, element_to_be_clickable
import os

#Insert chrome driver directory here
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager

driver = webdriver.Chrome(ChromeDriverManager().install())
```

In [6]:
```
driver.get('https://infogram.com/reopening-chart-1h7j4dmw0wqx4nr')
```

In [23]:
```
ds = pd.DataFrame(np.reshape([i.text for i in driver.find_elements_by_t
ag_name('td')],(51,5)).tolist())
```

In [25]:
```
ds
```

Out[25]:

	0	1	2	3	4
0	Alabama	April 4 - April 30; Penalties not mentioned.	Alabama has reopened retail stores, restaurant...	Yes – required for anyone older than age 6 on ...	There are no statewide restrictions.

	0	1	2	3	4
1	Alaska	March 28 - April 24: A business or organizatio...	Alaska has reopened retail stores, dining, bar...	No	All non-residents entering the state must prov...
2	Arizona	March 31 - May 15; Prior to any enforcement ac...	Arizona has reopened retail stores, restaurant...	No	There are no statewide restrictions.
3	Arkansas	No stay at home order.	Arkansas never issued a stay-at-home order and...	Yes – required for anyone age 10 or older in p...	There are no statewide restrictions.
4	California	March 19 until lifted; Any person that refuses...	Most counties have reopened restaurants and pe...	Yes – required for anyone age 2 or older in pu...	There are no statewide restrictions.
5	Colorado	March 26 - April 26: Local authorities have di...	Colorado has reopened retail stores, restauran...	Yes – required for anyone age 10 in public ind...	There are no statewide restrictions.
6	Connecticut	March 23 - May 20: Penalties not mentioned	Connecticut has reopened retail stores, malls,...	Yes – required for anyone age 2 or older in pu...	Travelers from a state with a current daily po...
7	Delaware	March 24 - May 31: Failure to comply is a crim...	Delaware has reopened retail stores, malls, fa...	Yes – required for anyone over the age of 12 w...	There are no statewide restrictions.
8	District of Columbia	April 1 - May 15: Any individual or entity tha...	Washington, DC has reopened restaurants with o...	Yes – Required for anyone over the age of 2 wh...	Visitors who have been to a high-risk states i...
9	Florida	April 3 - April 30: Extended to June 12 for th...	Florida has reopened retail stores, restaurant...	No	There are no statewide restrictions.
10	Georgia	April 3 - April 30 (extended to June 12 for th...	Georgia has reopened gyms, personal care servi...	No	There are no statewide restrictions.
11	Hawaii	March 25 - May 31: Any person who intentionall...	Hawaii has reopened beaches, piers, docks, sta...	Yes - required to enter a business or public s...	Travelers and residents arriving from out of s...

	0	1	2	3	4
12	Idaho	March 25 - April 30: Violation of or failure t...	Idaho has reopened retail stores, restaurant d...	No	The state is encouraging those traveling from ...
13	Illinois	March 25 - May 31: May be enforced by state an...	Illinois has reopened retail stores, restauran...	Yes – required for anyone over the age of 2 in...	The state is encouraging travelers from a coun...
14	Indiana	March 24 - May 1: May be enforced by state and...	Indiana has reopened retail stores, restaurant...	Yes – required for anyone age 8 or older when ...	There are no statewide restrictions.
15	Iowa	No stay at home order.	Iowa never issued a stay-at-home order but ins...	No	There are no statewide restrictions.
16	Kansas	March 30 - May 3: Penalties not mentioned.	Kansas has reopened gyms, personal care servic...	Yes – required for anyone over the age of 5 in...	Those who are entering the state who have trav...
17	Kentucky	In effect for the duration of the state emerge...	Kentucky has reopened retail stores, restauran...	Yes – required for anyone older than age of 5 ...	There are no statewide restrictions.
18	Louisiana	March 22 - May 15: The governor's Office of Ho...	Louisiana has opened retail stores, malls, per...	Yes – required for anyone age 8 or older in pu...	There are no statewide restrictions.
19	Maine	April 2 - May 31. The order will be enforced b...	Maine has reopened retail stores, restaurants,...	Yes – required for anyone over the age of 2 in...	Travelers from all states must self-quarantine...
20	Maryland	Until termination of the state of emergency an...	Maryland has reopened retail stores, malls, ou...	Yes – required for anyone over the age of 5 in...	The state strongly discourages travel to or fr...
21	Massachusetts	March 24 - May 18: Penalties not mentioned.	Massachusetts has reopened outdoor recreation,...	Yes – required for anyone over the age of 2 in...	Travelers from all states (except CT, CO, DE, ...
22	Michigan	March 24 - May 28: Lifted May 18 for the Upper...	Michigan has reopened retail stores, restauran...	Yes – required for anyone over age 4 in all in...	There are no statewide restrictions.

	0	1	2	3	4
23	Minnesota	March 27 - May 17: A person who willfully viol...	Minnesota has reopened industrial and manufact...	Yes – required for anyone over age 5 in indoor...	There are no statewide restrictions.
24	Mississippi	March 31 - May 11: May be enforced by all stat...	Mississippi has reopened retail stores, restau...	Yes – required in schools and at public gather...	There are no statewide restrictions.
25	Missouri	April 6 - May 3: Penalties not mentioned.	Missouri citizens may return to economic and s...	No	There are no statewide restrictions.
26	Montana	March 29 - April 26: Enforceable by the Attorn...	Montana has reopened main street and retail bu...	Yes – required for anyone age 5 or older in in...	There are no statewide restrictions.
27	Nebraska	No stay at home order.	Nebraska never issued a stay-at-home order and...	No	There are no statewide restrictions.
28	Nevada	April 2 - May 9: Local governments responsible...	Nevada has reopened retail stores, malls, rest...	Yes – required for anyone over age 9 in public...	There are no statewide restrictions.
29	New Hampshire	March 27 - June 15; The Division of Public Hea...	New Hampshire has reopened retail stores, rest...	Yes - required for gatherings of more than 100...	Travelers from all states outside of New Engla...
30	New Jersey	March 21 - June 9:; Penalties for violations o...	New Jersey has reopened retail stores, outdoor...	Yes – required for anyone over age 2 in indoor...	Travelers from a state with either more than 1...
31	New Mexico	March 24 - May 31; Penalties not mentioned.	New Mexico has reopened retail stores, malls, ...	Yes – required in public spaces.	All travelers must self-quarantine for 14 days...
32	New York	March 22 - May 28: Penalties not mentioned.	New York has reopened retail stores, outdoor d...	Yes – required for anyone over age 2 in public...	Travelers from a state with either more than 1...
33	North Carolina	March 30 - May 22: Violation is punishable as ...	North Carolina has reopened retail stores, res...	Yes – required for people over age 2 in public...	There are no statewide restrictions.
34	North Dakota	No stay at home order.	North Dakota never issued a stay-at-home order...	No	There are no statewide restrictions.

	0	1	2	3	4
35	Ohio	March 23 - May 29: Enforced by state and local...	Ohio has reopened retail stores, restaurant di...	Yes – required for people age 10 and older whe...	The state encourages travelers from states rep...
36	Oklahoma	March 24 - May 6: Penalties not mentioned.	Oklahoma reopened retail stores, restaurant di...	No	There are no statewide restrictions.
37	Oregon	March 23 until further notice: Any person foun...	Oregon has reopened retail stores, restaurant ...	Yes – required in public spaces for people age...	There are no statewide restrictions.
38	Pennsylvania	March 23 - June 4: Penalties not mentioned.	Pennsylvania has reopened retail stores, house...	Yes – required for anyone age 2 or older in pu...	Travelers from a state deemed at risk are reco...
39	Rhode Island	March 28 - May 8: Penalties not mentioned.	Rhode Island has reopened retail stores, resta...	Yes – required in all public spaces.	Travelers from states with a positivity rate o...
40	South Carolina	April 6 - May 4: All law enforcement officers ...	South Carolina has reopened retail stores, res...	No	The state is encouraging out-of-state traveler...
41	South Dakota	No stay at home order.	The governor never issued a stay-at-home order...	No	There are no statewide restrictions.
42	Tennessee	March 31 - April 30: Penalties not mentioned.	Tennessee has reopened restaurants and retail ...	No	There are no statewide restrictions.
43	Texas	March 31 - April 30: Failure to comply with an...	Texas has reopened retail stores, restaurants,...	Yes – required in all counties with more than ...	There are no statewide restrictions.
44	Utah	March 27 - May 1: Penalties not mentioned.	Utah has reopened restaurants, personal servic...	No	There are no statewide restrictions.
45	Vermont	March 24 - May 15: Penalties not mentioned.	Vermont has reopened retail stores, restaurant...	Yes – required for anyone age 2 or older when ...	Travelers driving must either quarantine for 1...

	0	1	2	3	4
46	Virginia	March 24 - June 10: Class 1 misdemeanor: jail ...	Virginia has reopened retail stores, restauran...	Yes – required in public places for anyone ove...	There are no statewide restrictions.
47	Washington	March 25 - May 31: Criminal penalties pursuant...	Washington has reopened retail stores, restaur...	Yes – required for anyone age 5 or older in an...	There are no statewide restrictions.
48	West Virginia	March 24 - May 4: The order may be enforced by...	West Virginia has reopened retail stores, mall...	Yes – required for anyone age 9 or older in al...	There are no statewide restrictions.
49	Wisconsin	March 25 – May 13: Order may be enforced by an...	The governor's stay-at-home order was to be in...	Yes – required for anyone age 5 or older in pu...	The state encourages travelers to check themse...
50	Wyoming	No stay at home order.	Wyoming never issued a stay-at-home order and ...	No	There are no statewide restrictions.

In [29]:
```
ds.columns = ['state','time','reopen','requirement','add.restrictions']
```

In [44]:
```
def clean_func(x):
    if "-" in x:
        return(x.split(';')[0].split(':')[0].split('.')[0].split('(')[0
])

ds['time_range'] = ds.time.apply(clean_func)
```

In [46]:
```
ds.to_csv('ds.csv')
```

Part 2

In [42]:
```
lever_variables = pd.read_csv('https://docs.google.com/spreadsheets/u/
1/d/1rHxjvo7rZHRo08x3RUTl4g8eSf0yumngTUIIsyhHUqw/export?format=csv&id=1
rHxjvo7rZHRo08x3RUTl4g8eSf0yumngTUIIsyhHUqw&gid=98237079')
```

```
lever_variables['lockdown_len'] = lever_variables['lockdown_len'].apply
(lambda x: str(x).split(" ")[0])
lever_variables['added_levels'] = lever_variables.penalties + lever_var
iables.masks_required
states = lever_variables.copy()
```

In [141]:
```
states['share_65'] = states['Total number, adults age 65 and older']/st
ates['Population 2019']
density = (states.iloc[:,10].astype(float)/states.iloc[:,9].astype(floa
t)).reset_index()[[0]]
density.columns = ['density']

states = pd.concat([states,pd.get_dummies(states.iloc[:,13]), density],
axis = 1)
```

In [143]:
```
pruned_states = states[['State','Abbreviation','Population 2019','lockd
own_len','density','Democrat','Republican','added_levels', 'share_65']]
```

In [144]:
```
pruned_states
```

Out[144]:

	State	Abbreviation	Population 2019	lockdown_len	density	density	Democrat
0	Alabama	AL	4903185	26	93.531179	93.531179	0
1	Alaska	AK	731545	27	1.114438	1.114438	0
2	Arizona	AZ	7278717	45	63.845034	63.845034	0
3	Arkansas	AR	3017804	nan	56.744838	56.744838	0
4	California	CA	39512223	nan	241.359398	241.359398	1
5	Colorado	CO	5758736	31	55.319270	55.319270	1
6	Connecticut	CT	3565287	58	643.089286	643.089286	1
7	Delaware	DE	973764	68	498.343910	498.343910	1
8	District of Columbia	DC	705749	44	10.732519	10.732519	1

	State	Abbreviation	Population 2019	lockdown_len	density	density	Democrat
9	Florida	FL	21477737	27	361.328662	361.328662	0
10	Georgia	GA	10617423	27	971.224204	971.224204	0
11	Hawaii	HI	1415872	67	16.941537	16.941537	1
12	Idaho	ID	1787065	36	30.855088	30.855088	0
13	Illinois	IL	12671821	67	347.935777	347.935777	1
14	Indiana	IN	6732219	38	119.628598	119.628598	0
15	Iowa	IA	3155070	nan	38.344595	38.344595	0
16	Kansas	KS	2913314	34	72.092104	72.092104	0
17	Kentucky	KY	4467673	nan	86.176977	86.176977	0
18	Louisiana	LA	4648794	54	131.370108	131.370108	0
19	Maine	ME	1344212	59	108.343032	108.343032	1
20	Maryland	MD	6045680	nan	572.778778	572.778778	1
21	Massachusetts	MA	6892503	55	71.196188	71.196188	1
22	Michigan	MI	9986857	65	114.866717	114.866717	0
23	Minnesota	MN	5639632	51	116.439526	116.439526	1
24	Mississippi	MS	2976149	41	42.693899	42.693899	0
25	Missouri	MO	6137428	27	41.738150	41.738150	0
26	Montana	MT	1068778	28	13.815998	13.815998	0
27	Nebraska	NE	1934408	nan	17.495347	17.495347	0
28	Nevada	NV	3080156	37	329.393220	329.393220	1
29	New Hampshire	NH	1359711	80	155.894405	155.894405	1
30	New Jersey	NJ	8882190	80	73.048531	73.048531	1
31	New Mexico	NM	2096829	68	38.491583	38.491583	1
32	New York	NY	19453561	67	361.449267	361.449267	1

	State	Abbreviation	Population 2019	lockdown_len	density	density	Democrat
33	North Carolina	NC	10488084	53	148.337916	148.337916	0
34	North Dakota	ND	762062	nan	16.999688	16.999688	0
35	Ohio	OH	11689100	67	167.218860	167.218860	0
36	Oklahoma	OK	3956971	43	40.218842	40.218842	0
37	Oregon	OR	4217737	nan	91.574471	91.574471	1
38	Pennsylvania	PA	12801989	73	8286.077023	8286.077023	0
39	Rhode Island	RI	1059361	41	33.097791	33.097791	1
40	South Carolina	SC	5148714	28	66.761505	66.761505	0
41	South Dakota	SD	884659	nan	20.990343	20.990343	0
42	Tennessee	TN	6829174	30	25.424976	25.424976	0
43	Texas	TX	28995881	30	341.513721	341.513721	0
44	Utah	UT	3205958	35	333.432969	333.432969	0
45	Vermont	VT	623989	52	14.589750	14.589750	1
46	Virginia	VA	8535519	78	119.707712	119.707712	1
47	Washington	WA	7614893	67	314.262432	314.262432	1
48	West Virginia	WV	1792147	41	27.359770	27.359770	0
49	Wisconsin	WI	5822434	nan	59.523135	59.523135	0
50	Wyoming	WY	578759	nan	8511.161765	8511.161765	0
51	Puerto Rico	NaN	3193694	nan	908.590043	908.590043	0

In [145]:
```
import datetime
```

In [154]:
```
bounds = lever_variables[['State','Abbreviation','lockdown_start','lock
down_end']]
def convert(x):
```

```
    try:
        return(datetime.datetime.strptime(str(x),'%Y-%m-%d'))
    except:
        return(datetime.datetime.strptime('1899-01-01','%Y-%m-%d'))

years_added = datetime.timedelta(days = 365 * 120)
bounds.lockdown_end = bounds.lockdown_end.apply(lambda x: convert(x) +
years_added)
bounds.lockdown_start = bounds.lockdown_start.apply(lambda x: convert(x
) + years_added)
bounds = bounds.dropna(axis = 0)
```

In [233]:

```
covid = pd.read_csv('https://covidtracking.com/data/download/all-states
-history.csv')
```

In [234]:

```
covid.dropna(subset = ['state']).columns
```

Out[234]:

```
Index(['date', 'state', 'dataQualityGrade', 'death', 'deathConfirmed',
       'deathIncrease', 'deathProbable', 'hospitalized',
       'hospitalizedCumulative', 'hospitalizedCurrently',
       'hospitalizedIncrease', 'inIcuCumulative', 'inIcuCurrently', 'ne
gative',
       'negativeIncrease', 'negativeTestsAntibody',
       'negativeTestsPeopleAntibody', 'negativeTestsViral',
       'onVentilatorCumulative', 'onVentilatorCurrently', 'pending',
       'positive', 'positiveCasesViral', 'positiveIncrease', 'positiveS
core',
       'positiveTestsAntibody', 'positiveTestsAntigen',
       'positiveTestsPeopleAntibody', 'positiveTestsPeopleAntigen',
       'positiveTestsViral', 'recovered', 'totalTestEncountersViral',
       'totalTestEncountersViralIncrease', 'totalTestResults',
       'totalTestResultsIncrease', 'totalTestsAntibody', 'totalTestsAnt
igen',
       'totalTestsPeopleAntibody', 'totalTestsPeopleAntigen',
       'totalTestsPeopleViral', 'totalTestsPeopleViralIncrease',
       'totalTestsViral', 'totalTestsViralIncrease'],
      dtype='object')
```

In [235]:

```
covid.date = covid.date.apply(lambda x: convert(x))
filtered_covid = pd.merge(covid,bounds[['Abbreviation','lockdown_start'
,'lockdown_end']], left_on='state', right_on = 'Abbreviation')
weeks2_added = datetime.timedelta(days = 14)
filtered_covid['lockdown_end_delayed'] = filtered_covid['lockdown_end']
 + weeks2_added
filtered_covid['lockdown_start_delayed'] = filtered_covid['lockdown_sta
rt'] + weeks2_added
filtered_covid = filtered_covid.query('(date<=lockdown_end_delayed) &
 (date>=lockdown_start_delayed)')
```

In [242]:

```
filtered_covid_peak = filtered_covid.groupby('state').agg({'deathIncrea
se':'max',
                                     'hospitalizedIncrease':'max',
                                     'positiveIncrease':'max'}).reset_i
ndex()
```

In [243]:

```
filtered_covid_end = filtered_covid.sort_values('date').reset_index().g
roupby('state').agg({'deathIncrease':'last',
                                     'hospitalizedIncrease':'last',
                                     'positiveIncrease':'last'}).reset_
index()
```

In [251]:

```
fil_cov = pd.merge(filtered_covid_peak,filtered_covid_end, on = 'state'
)
fil_cov['death_diminishing_rate'] = fil_cov['deathIncrease_y']/fil_cov[
'deathIncrease_x']
fil_cov['hospitalized_diminishing_rate'] = fil_cov['hospitalizedIncreas
e_y']/fil_cov['hospitalizedIncrease_x']
fil_cov['positive_diminishing_rate'] = fil_cov['positiveIncrease_y']/fi
l_cov['positiveIncrease_x']
```

In [255]:

```
fil_cov = fil_cov[['state','death_diminishing_rate',
         'hospitalized_diminishing_rate',
         'positive_diminishing_rate']].fillna(fil_cov.hospitalized_dimi
nishing_rate.mean(skipna = True))
```

In [256]:
```
filtered_covid_agg = fil_cov
```

In [266]:
```
from sklearn.preprocessing import StandardScaler
filtered_covid_agg_data = StandardScaler().fit_transform(filtered_covid
_agg[['death_diminishing_rate',

      'hospitalized_diminishing_rate',

      'positive_diminishing_rate']])
```

In [267]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
principalComponents = pca.fit(filtered_covid_agg_data)
```

In [268]:
```
principalComponents.explained_variance_ratio_
```

Out[268]:
```
array([0.52673492, 0.27307939, 0.20018569])
```

In [269]:
```
principalComponents.components_
```

Out[269]:
```
array([[ 0.59069292,  0.5009272 ,  0.63257713],
       [-0.51333867,  0.83814829, -0.18436607],
       [ 0.62254742,  0.21582257, -0.75223356]])
```

In [296]:
```
filtered_covid_agg_data
```

Out[296]:
```
array([[-0.08679487,  1.3729835 ,  0.19620723],
       [ 0.96462078, -0.4564843 , -0.6299453 ],
       [-0.01447528,  2.21565164,  0.99622318],
       [ 1.43191663,  2.60034796,  1.62884168],
       [-0.87404533, -1.08174544, -1.87315364],
       [ 1.43191663,  0.        , -0.16038737],
       [ 0.35988498,  0.        , -0.72621194],
       [ 1.12817433,  1.55495867, -0.16996099],
       [-0.48165987,  0.41774961,  0.2410829 ],
       [-1.60550638, -1.08174544, -1.94319354],
       [ 1.43191663, -0.63989423, -1.44723529],
       [ 0.4618496 ,  0.        ,  0.24863326],
```

```
       [ 1.04533552, -1.08174544,  0.78910924],
       [-0.50098892,  0.39109192,  0.93323987],
       [-0.56948613,  0.        , -1.39268913],
       [-0.03858181, -0.28920422,  0.34701358],
       [-1.60550638, -0.78116639, -0.12215598],
       [-0.72483344,  0.        ,  1.62884168],
       [ 1.43191663,  0.95515729,  1.3198421 ],
       [ 0.47273042,  0.        , -0.17641224],
       [ 0.99799906, -0.29055182,  1.4420686 ],
       [-1.60550638,  1.98666573, -0.83201989],
       [ 0.50381516,  0.        ,  1.12684918],
       [-0.64632017, -0.79850749,  0.37661305],
       [-1.26450671,  0.        , -1.01900844],
       [ 1.17879805, -1.08174544,  0.16357168],
       [ 0.49732494,  0.        , -0.14930323],
       [-0.95075168, -0.764509  , -1.44036016],
       [-0.57102173,  0.89706666, -0.67790741],
       [-1.03598957, -1.08174544, -0.21431375],
       [-0.95306353, -1.08174544, -0.79913798],
       [-0.18171434, -0.60340702,  1.43813897],
       [ 0.41944229,  0.37665234,  1.12447823],
       [ 0.56408148,  2.07433462, -0.19314237],
       [ 1.43191663,  0.        , -0.12823161],
       [-0.99802178, -0.67822835,  0.62827159],
       [ 0.40136239, -0.88674361,  1.62884168],
       [-1.60550638, -1.08174544, -1.67303962],
       [-1.27263811,  0.        , -0.23723042],
       [ 1.43191663, -1.08174544, -0.25282744]])
```

In [270]:
```python
filtered_covid_agg['health_index'] = [i[0] for i in principalComponents
.transform(filtered_covid_agg_data)]
```

In [303]:
```python
filtered_covid_agg[['state','health_index']]
```

Out[303]:

	state	health_index
0	AK	0.760612

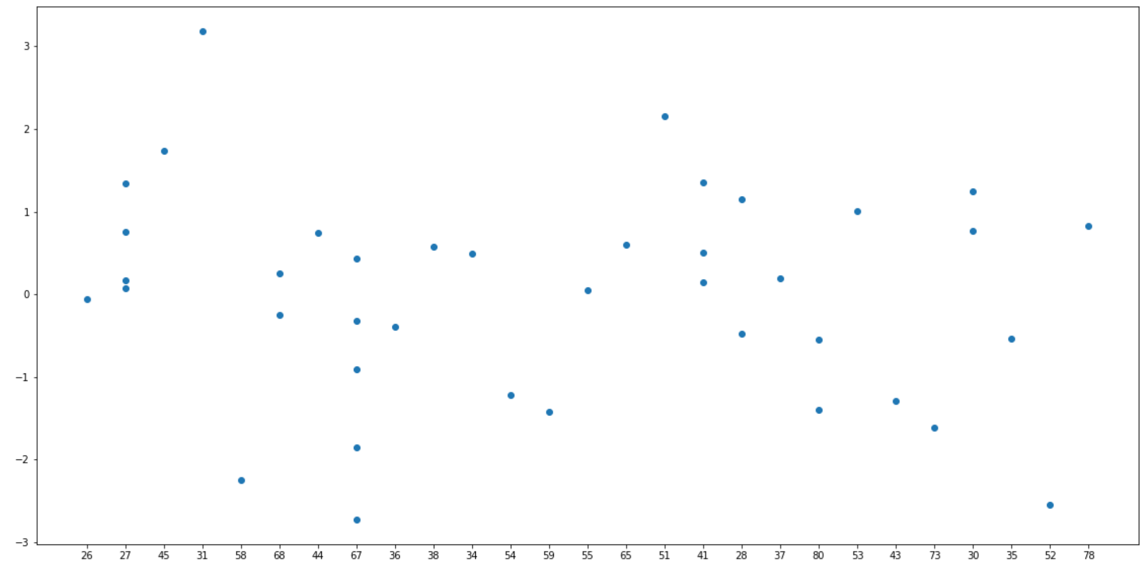
	state	health_index
1	AL	-0.057360
2	AZ	1.731518
3	CO	3.178776
4	CT	-2.243082
5	DC	0.744366
6	DE	-0.246804
7	FL	1.337812
8	GA	0.077253
9	HI	-2.719457
10	ID	-0.390205
11	IL	0.430091
12	IN	0.574769
13	KS	0.490324
14	LA	-1.217375
15	MA	0.051853
16	ME	-1.416942
17	MI	0.602214
18	MN	2.159189
19	MO	0.167644
20	MS	1.356185
21	MT	-0.479503
22	NC	1.010419
23	NH	-0.543534
24	NJ	-1.391537
25	NM	0.257904

	state	health_index
26	NV	0.199321
27	NY	-1.855705
28	OH	-0.316762
29	OK	-1.289397
30	PA	-1.610360
31	RI	0.500133
32	SC	1.147756
33	TN	1.250112
34	TX	0.764707
35	UT	-0.531837
36	VA	0.823256
37	VT	-2.548564
38	WA	-0.901805
39	WV	0.144014

In [271]:
```
health_i = pd.merge(pruned_states,filtered_covid_agg[['state','health_index']], left_on = 'Abbreviation', right_on = 'state')
```

In [279]:
```
plt.scatter(health_i.lockdown_len,health_i.health_index)
```

Out[279]: <matplotlib.collections.PathCollection at 0x1312789e8>



In [273]:
```
health_i.to_csv('health_index.csv')
```

In [274]:
```
econ = pd.read_csv('Downloads/gdp1.csv').iloc[:,1:7].dropna(axis = 0).d
rop(['Unemployment Feb20'], axis = 1)
```

In [275]:
```
econ.iloc[:,2] = econ.iloc[:,2].apply(lambda x: x.replace("%","")).appl
y(float)
econ.iloc[:,3] = econ.iloc[:,3].apply(lambda x: x.replace("%","")).appl
y(float)
econ.iloc[:,4] = econ.iloc[:,4].apply(lambda x: x.replace("%","")).appl
y(float)
```

In [280]:
```
econ['Unemployment_Rate'] = econ.iloc[:,4]/econ.iloc[:,2]
```

In [289]:
```
econ = econ.drop(['Unemployment March20','Unemployment April20','Unempl
oyment May20'], axis = 1)
```

In [290]:
```
from sklearn.decomposition import PCA
```

```
pca = PCA(n_components=2)
principalComponents2 = pca.fit(econ.iloc[:,1:])
```

In [291]:
```
principalComponents2.explained_variance_ratio_
```

Out[291]:
```
array([0.71729648, 0.28270352])
```

In [292]:
```
principalComponents2.components_
```

Out[292]:
```
array([[-0.71484831,  0.69927956],
       [ 0.69927956,  0.71484831]])
```

In [293]:
```
econ['indexes'] = [i[0] for i in principalComponents2.transform(econ.il
oc[:,1:])]
```

In [300]:
```
econ
```

Out[300]:

	State	GDP change 1st quarter	Unemployment_Rate	indexes
0	Alabama	-4.8	3.200000	-0.049782
1	Alaska	-4.0	2.442308	-1.151499
2	Arizona	-3.6	1.475410	-2.113570
3	Arkansas	-5.0	1.920000	-0.801890
4	California	-4.7	2.981818	-0.273837
5	Colorado	-4.1	1.961538	-1.416206
6	Connecticut	-4.6	2.823529	-0.456010
7	Delaware	-5.6	3.180000	0.508111
8	District of Columbia	-4.0	1.466667	-1.833745
9	Florida	-4.9	3.113636	-0.038689
10	Georgia	-4.7	2.043478	-0.929999
11	Hawaii	-8.1	9.791667	6.918635

	State	GDP change 1st quarter	Unemployment_Rate	indexes
12	Idaho	-4.1	3.600000	-0.270464
13	Illinois	-5.4	3.642857	0.688808
14	Indiana	-5.6	4.100000	1.151449
15	Iowa	-3.5	3.090909	-1.055370
16	Kansas	-3.1	3.571429	-1.005291
17	Kentucky	-5.8	2.096154	-0.106830
18	Louisiana	-6.6	2.119403	0.481306
19	Maine	-6.3	3.133333	0.975872
20	Maryland	-5.0	3.030303	-0.025478
21	Massachusetts	-5.1	5.928571	2.072707
22	Michigan	-6.8	4.953488	2.606093
23	Minnesota	-4.0	3.413793	-0.472159
24	Mississippi	-5.2	2.058824	-0.561844
25	Missouri	-4.7	2.589744	-0.548006
26	Montana	-5.4	2.500000	-0.110368
27	Nebraska	-1.3	1.325000	-3.862900
28	Nevada	-8.2	3.666667	2.707033
29	New Hampshire	-5.7	6.416667	2.842931
30	New Jersey	-5.5	4.162162	1.123432
31	New Mexico	-3.1	1.444444	-2.492648
32	New York	-8.2	3.536585	2.616070
33	North Carolina	-5.1	2.976744	0.008555
34	North Dakota	-2.6	4.550000	-0.678421
35	Ohio	-5.5	2.396552	-0.111223
36	Oklahoma	-4.0	4.344828	0.178894

	State	GDP change 1st quarter	Unemployment_Rate	indexes
37	Oregon	-4.4	4.085714	0.283641
38	Pennsylvania	-5.6	2.310345	-0.100021
39	Rhode Island	-6.2	3.489362	1.153351
40	South Carolina	-4.8	3.875000	0.422232
41	South Dakota	-2.2	3.032258	-2.025686
42	Tennessee	-6.2	3.333333	1.044243
43	Texas	-2.5	2.549020	-2.149150
44	Utah	-3.1	2.263158	-1.920138
45	Vermont	-6.1	4.129032	1.529174
46	Virginia	-3.8	2.727273	-1.095199
47	Washington	-5.0	2.960784	-0.074091
48	West Virginia	-5.0	2.150000	-0.641056
49	Wisconsin	-5.0	3.903226	0.584939
50	Wyoming	-3.6	2.315789	-1.525910

In [301]:
```
econ.iloc[:,[0,3]]
```

Out[301]:

	State	indexes
0	Alabama	-0.049782
1	Alaska	-1.151499
2	Arizona	-2.113570
3	Arkansas	-0.801890
4	California	-0.273837
5	Colorado	-1.416206
6	Connecticut	-0.456010

	State	indexes
7	Delaware	0.508111
8	District of Columbia	-1.833745
9	Florida	-0.038689
10	Georgia	-0.929999
11	Hawaii	6.918635
12	Idaho	-0.270464
13	Illinois	0.688808
14	Indiana	1.151449
15	Iowa	-1.055370
16	Kansas	-1.005291
17	Kentucky	-0.106830
18	Louisiana	0.481306
19	Maine	0.975872
20	Maryland	-0.025478
21	Massachusetts	2.072707
22	Michigan	2.606093
23	Minnesota	-0.472159
24	Mississippi	-0.561844
25	Missouri	-0.548006
26	Montana	-0.110368
27	Nebraska	-3.862900
28	Nevada	2.707033
29	New Hampshire	2.842931
30	New Jersey	1.123432
31	New Mexico	-2.492648

	State	indexes
32	New York	2.616070
33	North Carolina	0.008555
34	North Dakota	-0.678421
35	Ohio	-0.111223
36	Oklahoma	0.178894
37	Oregon	0.283641
38	Pennsylvania	-0.100021
39	Rhode Island	1.153351
40	South Carolina	0.422232
41	South Dakota	-2.025686
42	Tennessee	1.044243
43	Texas	-2.149150
44	Utah	-1.920138
45	Vermont	1.529174
46	Virginia	-1.095199
47	Washington	-0.074091
48	West Virginia	-0.641056
49	Wisconsin	0.584939
50	Wyoming	-1.525910

In [294]:
```
econ.to_csv('econ.csv')
```

In [ ]:
```
covid.dropna(subset = ['state'])[]
```

In [ ]:
```
```

In [6]:
```
import numpy as np
import pandas as pd

from datetime import datetime

import matplotlib.pyplot as plt
import matplotlib.cm as cm

import sklearn.model_selection as ms
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error
from sklearn.cluster import KMeans
from sklearn.manifold import TSNE
from sklearn.inspection import plot_partial_dependence
```

In [7]:
```
lockdown_data = pd.read_csv("~/Downloads/ds.csv")
```

In [8]:
```
lockdown_data["time_range"] = lockdown_data["time_range"][lockdown_data
['time_range'].notnull()].apply(lambda x: x.split(" - "))
```

In [9]:
```
lockdown_data["lockdown_start"] = lockdown_data["time_range"][lockdown_
data['time_range'].notnull()].apply(lambda x:x[0])
lockdown_data["lockdown_end"] = lockdown_data["time_range"][lockdown_da
ta['time_range'].notnull()].apply(lambda x:x[1])
```

In [10]:
```
lockdown_data["lockdown_end"][10] = "April 30"
lockdown_data["lockdown_start"] = lockdown_data["lockdown_start"][lockd
own_data['lockdown_start'].notnull()].apply(lambda x:datetime.strptime(
x, '%B %d'))
lockdown_data["lockdown_end"] = lockdown_data["lockdown_end"][lockdown_
data['lockdown_end'].notnull()].apply(lambda x:datetime.strptime(x, '%B
 %d'))
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:1: Setting
```

```
WithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """Entry point for launching an IPython kernel.
```

In [11]:
```
lockdown_data["lockdown_len"] = lockdown_data["lockdown_end"] - lockdow
n_data["lockdown_start"]
```

In [12]:
```
lockdown_data["penalties"] = lockdown_data["time"].apply(lambda x : not
("Penalties not mentioned." in x))
```

In [13]:
```
lockdown_data["masks_required"] = lockdown_data["requirement"].apply(la
mbda x : "Yes" in x)
```

In [14]:
```
lockdown_data["additional"] = lockdown_data["add.restrictions"].apply(l
ambda x : not("There are no statewide restrictions." in x))
```

In [15]:
```
state_data = pd.read_csv("~/Desktop/State Data  - Sheet1.csv")
```

In [16]:
```
new_header = state_data.iloc[0]
state_data = state_data[1:]
state_data.columns = new_header
```

In [17]:
```
lockdown_data_new = lockdown_data.iloc[:, [1, 7, 8, 9, 10, 11, 12]]
```

In [18]:
```
lockdown_data_new = lockdown_data_new.rename(columns = {"state":"State"
})
state_data = state_data.rename(columns = {'State ':"State"})
```

In [19]:
```
merged_data = pd.merge(lockdown_data_new, state_data, on='State', how=
'outer')
```

In [20]:
```
merged_data.to_csv(r'~/Desktop/state_lockdown_data.csv')
```

In [ ]:
```
#skeleton code

X_train, X_test, y_train, y_test = ms.train_test_split(X, y, test_size=
0.2, random_state = 0)
rfregressor = RandomForestRegressor(max_depth=100, random_state=0)
rfregressor.fit(X_train, y_train)

#partial dependency
my_plots = plot_partial_dependence(rfregressor,
                                   features=[0, 2], # column numbers of
 plots we want to show
                                   X=X,            # raw predictors dat
a.
                                   feature_names=['Distance', 'Landsiz
e', 'BuildingArea'], # labels on graphs
                                   grid_resolution=10)

#feature importance
importances = rfregressor.feature_importances_
std = np.std([tree.feature_importances_ for tree in rfregressor.estimat
ors_],
             axis=0)
indices = np.argsort(importances)[::-1]

# Print the feature ranking
print("Feature ranking:")

for f in range(X.shape[1]):
    print("%d. feature %d (%f)" % (f + 1, indices[f], importances[indic
es[f]]))

# Plot the impurity-based feature importances of the forest
plt.figure()
plt.title("Feature importances")
plt.bar(range(X.shape[1]), importances[indices],
        color="r", yerr=std[indices], align="center")
plt.xticks(range(X.shape[1]), indices)
```

```
plt.xlim([-1, X.shape[1]])
plt.show()
```

In [963]:

```
import numpy as np
import pandas as pd

import statsmodels.api as sm

from datetime import datetime

import matplotlib.pyplot as plt
import matplotlib.cm as cm

import sklearn.model_selection as ms
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error
from sklearn.cluster import KMeans
from sklearn.manifold import TSNE
from sklearn.inspection import plot_partial_dependence
from sklearn.inspection import permutation_importance
from sklearn.metrics import r2_score
from sklearn.inspection import partial_dependence


import eli5
from eli5.sklearn import PermutationImportance

import plotly
import pandas as pd
import numpy as np
import seaborn as sns
import plotly.express as px
import matplotlib
%matplotlib inline
```

In [964]:

```
health_index = pd.read_csv("~/Desktop/health_index.csv")
econ_index = pd.read_csv("~/Desktop/econ.csv")
```

In [965]:
```
health_index = health_index.rename(columns = {'lockdown_len':"Lockdown
 Length"})
health_index = health_index.rename(columns = {'health_index':"Health In
dex"})
health_index = health_index.rename(columns = {'added_levels':"Added Lev
els"})
health_index = health_index.rename(columns = {'density':"Density"})
health_index = health_index.rename(columns = {'share_65':"Share_65"})
X = health_index.iloc[:, [3, 4, 5, 7, 9, 11, 12]]
X = X.drop(X.index[12])
y = health_index.iloc[:, [14]]
y = y.drop(y.index[12])
y = y.values.ravel()
```

In [966]:
```
for i in range(len(y)):
    temp_X = X.drop(X.index[i])
    temp_y = y.drop(y.index[i])
    temp_y = temp_y.values.ravel()
    rfregressor = RandomForestRegressor(max_depth=4, random_state=5)
    rfregressor.fit(temp_X, temp_y)
    print(i, rfregressor.predict(health_index.iloc[i, [3, 4, 5, 7, 9, 1
1, 12]].values.reshape(1, -1))-health_index.iloc[i, [14]].values)
```

```
-----------------------------------------------------------------------
----
AttributeError                            Traceback (most recent call l
ast)
<ipython-input-966-4227c171821b> in <module>
      1 for i in range(len(y)):
      2     temp_X = X.drop(X.index[i])
----> 3     temp_y = y.drop(y.index[i])
      4     temp_y = temp_y.values.ravel()
      5     rfregressor = RandomForestRegressor(max_depth=4, random_sta
te=5)

AttributeError: 'numpy.ndarray' object has no attribute 'drop'
```

In [888]:
```
LR = sm.OLS(y, X).fit()
```

In [889]:
```
LR.summary()
```
Out[889]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.291
Model:	OLS	**Adj. R-squared:**	0.163
Method:	Least Squares	**F-statistic:**	2.263
Date:	Fri, 25 Sep 2020	**Prob (F-statistic):**	0.0614
Time:	16:13:37	**Log-Likelihood:**	-59.017
No. Observations:	40	**AIC:**	132.0
Df Residuals:	33	**BIC:**	143.9
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Population 2019	3.654e-08	3.23e-08	1.130	0.266	-2.92e-08	1.02e-07
Lockdown Length	-0.0331	0.015	-2.167	0.038	-0.064	-0.002
Density	-0.0001	0.000	-0.818	0.419	-0.000	0.000
Democrat	2.3640	1.616	1.463	0.153	-0.924	5.652
Republican	2.3586	1.499	1.574	0.125	-0.691	5.408
Added Levels	0.2993	0.302	0.992	0.328	-0.314	0.913
Share_65	-8.4979	9.139	-0.930	0.359	-27.091	10.095

Omnibus:	0.684	**Durbin-Watson:**	1.989
Prob(Omnibus):	0.710	**Jarque-Bera (JB):**	0.706
Skew:	0.010	**Prob(JB):**	0.703
Kurtosis:	2.350	**Cond. No.**	4.48e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.48e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

In [967]:
```
X_train, X_test, y_train, y_test = ms.train_test_split(X, y, test_size=
0.2, random_state = 5)
rfregressor = RandomForestRegressor(max_depth=4, random_state=5)
rfregressor.fit(X, y)
np.sqrt(np.mean((rfregressor.predict(X_test) - y_test)**2))
```

Out[967]:
```
0.6237966692301442
```

In [968]:
```
rfregressor.predict(health_index.iloc[12, [3, 4, 5, 7, 9, 11, 12]].valu
es.reshape(1, -1))
```

Out[968]:
```
array([0.91188192])
```

In [969]:
```
health_index.iloc[12, [3, 4, 5, 7, 9, 11, 12]].values.reshape(1, -1)
```

Out[969]:
```
array([[6732219, 38, 119.6285983367688, 0, 1, 2, 0.1508273869284407]],
      dtype=object)
```

In [974]:
```
rfregressor.predict(np.array([6732219, 90, 119.6285983367688, 0, 1, 2,
0.1508273869284407]).reshape(1, -1))
```

Out[974]:
```
array([-0.44019457])
```

In [894]:
```
def pred_ints(model, X, percentile=95):
    err_down = []
    err_up = []
    perc_50 = []
    for x in range(len(X)):
        preds = []
        for pred in model.estimators_:
```

```
            preds.append(pred.predict(X_test.iloc[x,].values.reshape(1,
 -1)))
        err_down.append(np.percentile(preds, (100 - percentile) / 2. ))
        perc_50.append(np.percentile(preds, 50))
        err_up.append(np.percentile(preds, 100 - (100 - percentile) /
2.))
    return err_down, err_up, perc_50
```

In [895]:

```
pred_ints(rfregressor, X_test, percentile=90)
```

Out[895]:

```
([-0.05961785228657985,
  -0.4542011945606605,
  -1.402898366266888,
  0.05793002951319271,
  -1.6335839801633703,
  -1.3699375805583438,
  -2.3959525587486623,
  -1.2957746038908695],
 [1.3964562451662454,
  2.159189199233614 5,
  1.013029230539649,
  3.1787760193110546,
  0.25891439842809855,
  2.125244968632568,
  0.346541445223477,
  1.1085055967097148],
 [0.5930460748795401,
  0.22284549804590875,
  -0.9974719242363628,
  3.1787760193110546,
  -0.5435340548379382,
  0.6676628911436999,
  -2.049393386451893,
  0.1440144428018571])
```

In [896]:

```
truth = y_test
correct = 0.
for i, val in enumerate(truth):
    if err_down[i] <= val <= err_up[i]:
```

```
        correct += 1
print(correct/len(truth))
```

```
0.5
```

In [897]:
```
np.std(y_test)
```

Out[897]: 1.5000959095201696

In [898]:
```
correlation_matrix = np.corrcoef(rfregressor.predict(X_test), y_test)
correlation = correlation_matrix[0,1]
r_squared = correlation**2
r_squared
```

Out[898]: 0.9253695974295977

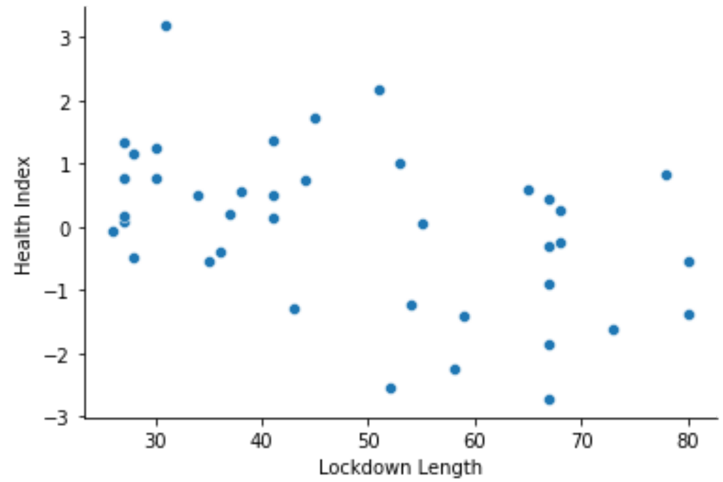
In [899]:
```
correlation_matrix = np.corrcoef(rfregressor.predict(X_train), y_train)
correlation_xy = correlation_matrix[0,1]
r_squared = correlation_xy**2
r_squared
```

Out[899]: 0.8688545829255999

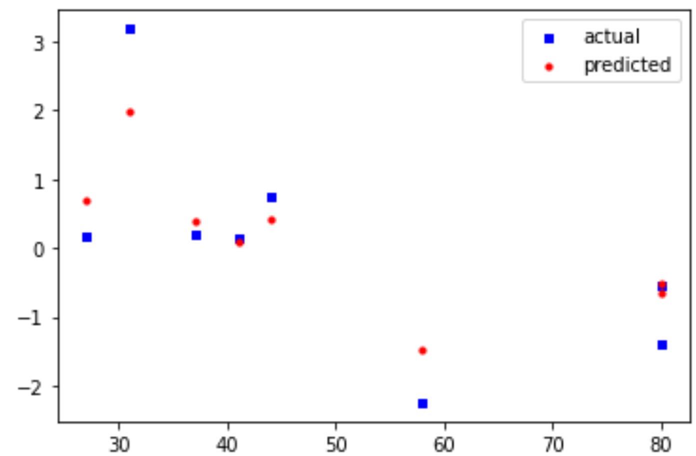
In [900]:
```
#r2_score(y_test, rfregressor.predict(X_test))
```

In [901]:
```
#r2_score(y_train, rfregressor.predict(X_train))
```

In [902]:
```
sns.scatterplot(
    x='Lockdown Length',
    y='Health Index',
    data=health_index
)
sns.despine()
```



In [903]:
```
fig = plt.figure()
ax1 = fig.add_subplot(111)

ax1.scatter(X_test.iloc[:,1], y_test, s=10, c='b', marker="s", label='a
ctual')
ax1.scatter(X_test.iloc[:,1], rfregressor.predict(X_test), s=10, c='r',
 marker="o", label='predicted')
plt.legend(loc='upper right')
plt.show()
```



In [904]:
```
#partial dependency - lockdown length
#my_plots = plot_partial_dependence(rfregressor, features=[1], X=X_trai
n, grid_resolution=10)
```

In [905]:
```
def plot_pdp(model, X, feature, target=False, return_pd=False, y_pct=Tr
ue, figsize=(10,9), norm_hist=True, dec=.5):
    # Get partial dependence
    pardep = partial_dependence(model, X, [feature])

    # Get min & max values
    xmin = pardep[1][0].min()
    xmax = pardep[1][0].max()
    ymin = pardep[0][0].min()
    ymax = pardep[0][0].max()

    # Create figure
    fig, ax1 = plt.subplots(figsize=figsize)
    ax1.grid(alpha=.5, linewidth=1)

    # Plot partial dependence
    color = 'tab:blue'
    ax1.plot(pardep[1][0], pardep[0][0], color=color)
```

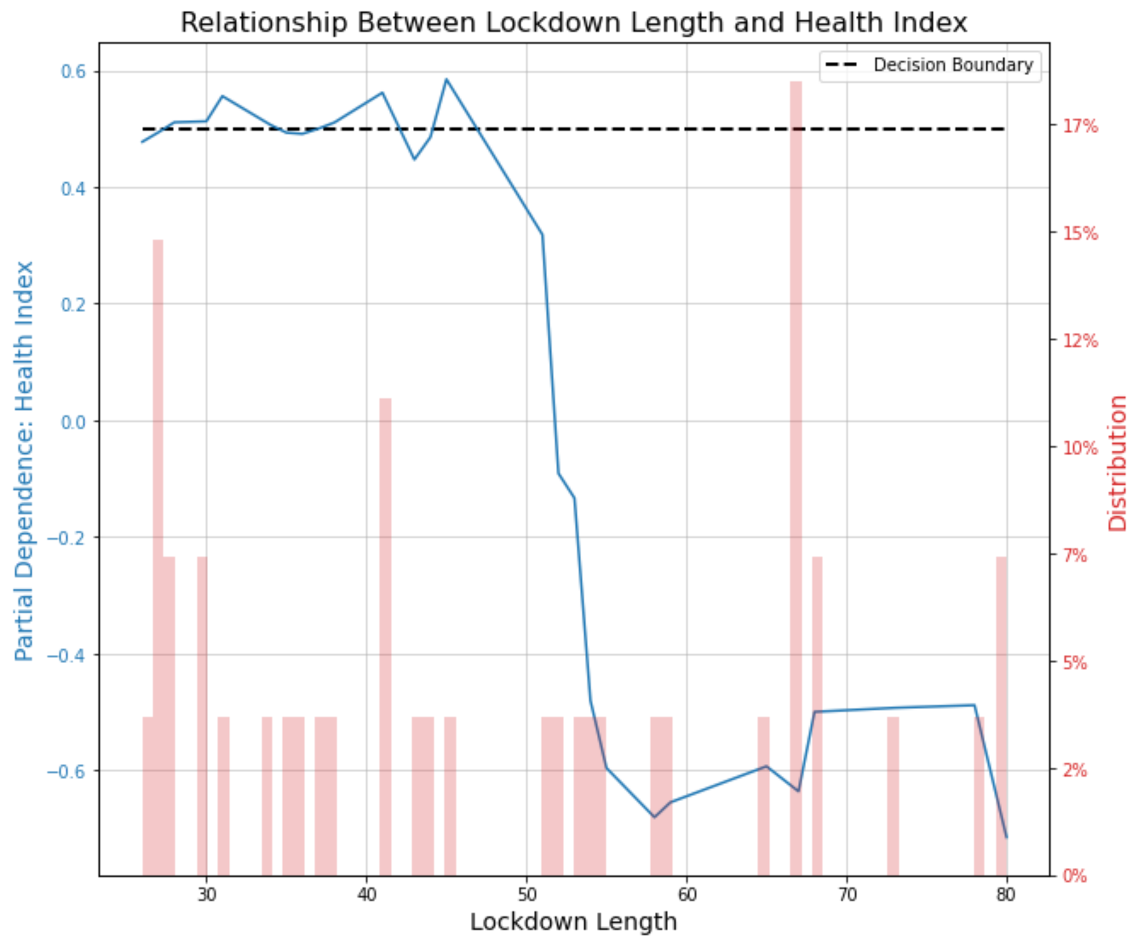
```
    ax1.tick_params(axis='y', labelcolor=color)
    ax1.set_xlabel(feature, fontsize=14)

    tar_ylabel = ': {}'.format(target) if target else ''
    ax1.set_ylabel('Partial Dependence{}'.format(tar_ylabel), color=col
or, fontsize=14)

    tar_title = target if target else 'Target Variable'
    ax1.set_title('Relationship Between {} and {}'.format(feature, tar_
title), fontsize=16)

    if y_pct and ymin>=0 and ymax<=1:
        # Display yticks on ax1 as percentages
        fig.canvas.draw()
        labels = [item.get_text() for item in ax1.get_yticklabels()]
        labels = [int(np.float(label)*100) for label in labels]
        labels = ['{}%'.format(label) for label in labels]
        ax1.set_yticklabels(labels)

    # Plot line for decision boundary
    ax1.hlines(dec, xmin=xmin, xmax=xmax, color='black', linewidth=2, l
inestyle='--', label='Decision Boundary')
    ax1.legend()

    ax2 = ax1.twinx()
    color = 'tab:red'
    ax2.hist(X[feature], bins=80, range=(xmin, xmax), alpha=.25, color=
color, density=norm_hist)
    ax2.tick_params(axis='y', labelcolor=color)
    ax2.set_ylabel('Distribution', color=color, fontsize=14)

    if y_pct and norm_hist:
        # Display yticks on ax2 as percentages
        fig.canvas.draw()
        labels = [item.get_text() for item in ax2.get_yticklabels()]
        labels = [int(np.float(label)*100) for label in labels]
        labels = ['{}%'.format(label) for label in labels]
        ax2.set_yticklabels(labels)
```

```
    plt.show()

    if return_pd:
        return pardep
```

In [906]:

```
plot_pdp(rfregressor, X, 'Lockdown Length', target='Health Index')
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:51: UserWa
rning:

FixedFormatter should only be used together with FixedLocator
```

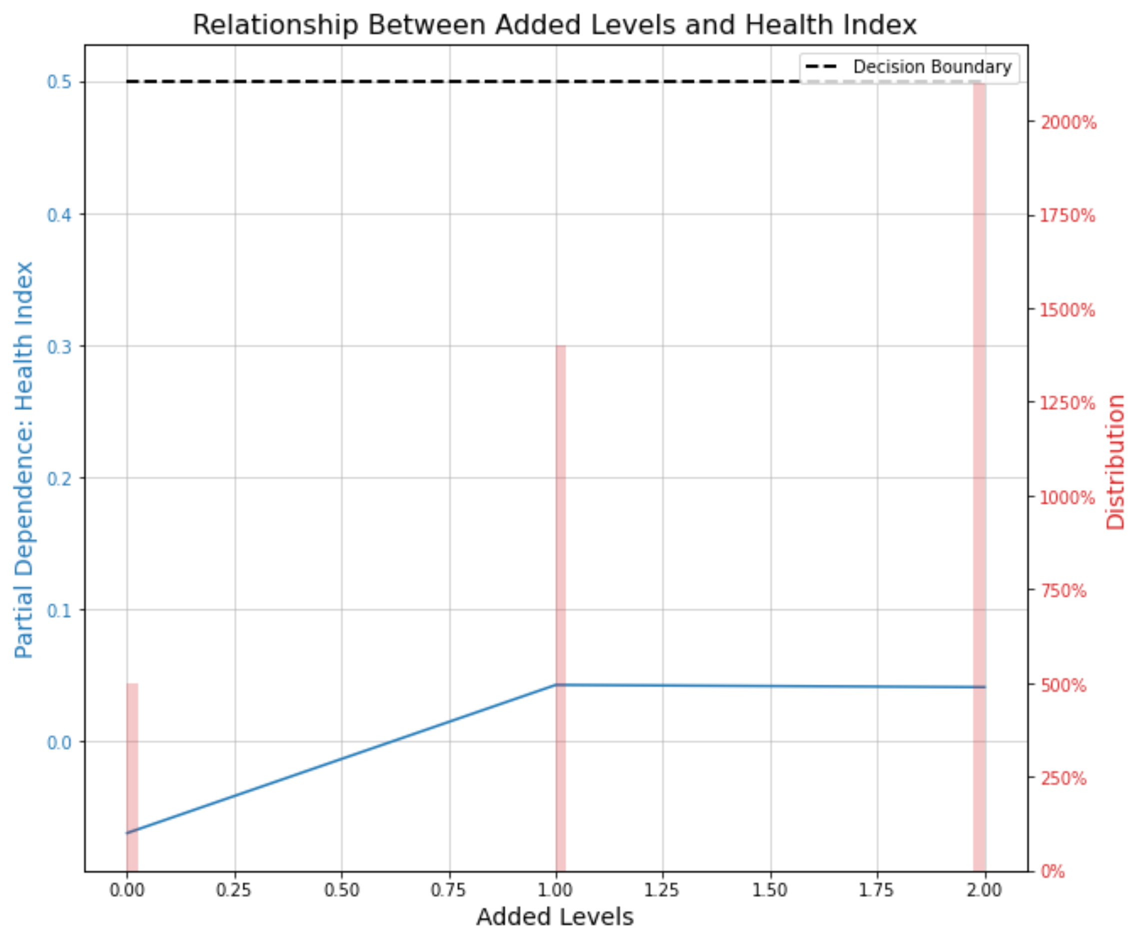


In [907]:
```
#partial dependency - added levels
#my_plots = plot_partial_dependence(rfregressor, features=[5], X=X_trai
n, grid_resolution=10)
```

In [908]:
```
plot_pdp(rfregressor, X, 'Added Levels', target='Health Index')
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:51: UserWa
```

```
rning:

FixedFormatter should only be used together with FixedLocator
```



In [909]:

```
perm = PermutationImportance(rfregressor, random_state=1).fit(X_test, y
_test)
eli5.show_weights(perm, feature_names = X_test.columns.tolist())
```

Out[909]:

Weight	Feature

Weight	Feature
1.0898 ± 0.7635	Lockdown Length
0.1605 ± 0.2849	Population 2019
0.0799 ± 0.1081	Democrat
0.0658 ± 0.0337	Density
0.0390 ± 0.0582	Republican
0.0223 ± 0.0577	Share_65
0.0041 ± 0.0090	Added Levels

In [1006]:

```
econ_index = econ_index.rename(columns = {'State ':"State"})
econ_index = econ_index.rename(columns = {'indexes':"Economic Index"})
econ_index["State"] = econ_index["State"].apply(lambda x: x.strip())
econ = pd.merge(health_index, econ_index, on='State')
econ
```

Out[1006]:

	Unnamed: 0_x	State	Abbreviation	Population 2019	Lockdown Length	Density	density.1	De
0	0	Alabama	AL	4903185	26	93.531179	93.531179	
1	1	Alaska	AK	731545	27	1.114438	1.114438	
2	2	Arizona	AZ	7278717	45	63.845034	63.845034	
3	3	Colorado	CO	5758736	31	55.319270	55.319270	
4	4	Connecticut	CT	3565287	58	643.089286	643.089286	
5	5	Delaware	DE	973764	68	498.343910	498.343910	
6	6	District of Columbia	DC	705749	44	10.732519	10.732519	
7	7	Florida	FL	21477737	27	361.328662	361.328662	
8	8	Georgia	GA	10617423	27	971.224204	971.224204	
9	9	Hawaii	HI	1415872	67	16.941537	16.941537	
10	10	Idaho	ID	1787065	36	30.855088	30.855088	
11	11	Illinois	IL	12671821	67	347.935777	347.935777	
12	12	Indiana	IN	6732219	38	119.628598	119.628598	

	Unnamed: 0_x	State	Abbreviation	Population 2019	Lockdown Length	Density	density.1	De
13	13	Kansas	KS	2913314	34	72.092104	72.092104	
14	14	Louisiana	LA	4648794	54	131.370108	131.370108	
15	15	Maine	ME	1344212	59	108.343032	108.343032	
16	16	Massachusetts	MA	6892503	55	71.196188	71.196188	
17	17	Michigan	MI	9986857	65	114.866717	114.866717	
18	18	Minnesota	MN	5639632	51	116.439526	116.439526	
19	19	Mississippi	MS	2976149	41	42.693899	42.693899	
20	20	Missouri	MO	6137428	27	41.738150	41.738150	
21	21	Montana	MT	1068778	28	13.815998	13.815998	
22	22	Nevada	NV	3080156	37	329.393220	329.393220	
23	23	New Hampshire	NH	1359711	80	155.894405	155.894405	
24	24	New Jersey	NJ	8882190	80	73.048531	73.048531	
25	25	New Mexico	NM	2096829	68	38.491583	38.491583	
26	26	New York	NY	19453561	67	361.449267	361.449267	
27	27	North Carolina	NC	10488084	53	148.337916	148.337916	
28	28	Ohio	OH	11689100	67	167.218860	167.218860	
29	29	Oklahoma	OK	3956971	43	40.218842	40.218842	
30	30	Pennsylvania	PA	12801989	73	8286.077023	8286.077023	
31	31	Rhode Island	RI	1059361	41	33.097791	33.097791	
32	32	South Carolina	SC	5148714	28	66.761505	66.761505	
33	33	Tennessee	TN	6829174	30	25.424976	25.424976	
34	34	Texas	TX	28995881	30	341.513721	341.513721	

	Unnamed: 0_x	State	Abbreviation	Population 2019	Lockdown Length	Density	density.1	De
35	35	Utah	UT	3205958	35	333.432969	333.432969	
36	36	Vermont	VT	623989	52	14.589750	14.589750	
37	37	Virginia	VA	8535519	78	119.707712	119.707712	
38	38	Washington	WA	7614893	67	314.262432	314.262432	
39	39	West Virginia	WV	1792147	41	27.359770	27.359770	

40 rows × 21 columns

In [1007]:
```
X = econ.iloc[:, [3, 4, 5, 7, 9, 11, 12]]
#X = X.drop(X.index[12])
y = econ.iloc[:, [20]]
#y = y.drop(y.index[12])
y = y.values.ravel()
```

In [924]:
```
for i in range(len(y)):
    temp_X = X.drop(X.index[i])
    temp_y = y.drop(y.index[i])
    temp_y = temp_y.values.ravel()
    rfregressor = RandomForestRegressor(max_depth=4, random_state=5)
    rfregressor.fit(temp_X, temp_y)
    print(i, rfregressor.predict(econ.iloc[i, [3, 4, 5, 7, 9, 11, 12]].
values.reshape(1, -1))-econ.iloc[i, [20]].values)
```

```
0 [0.8514288687687763]
1 [-1.5788217729574068]
2 [3.714555804725369]
3 [1.2160093484207768]
4 [10.591709298909816]
5 [1.5197627420769493]
6 [6.111596366163483]
7 [-0.6391143263823982]
```

```
8 [3.5506208604422946]
9 [-15.548246228110244]
10 [5.204449459171856]
11 [-1.8903128147384116]
12 [0.6793355036391016]
13 [2.5618318810547454]
14 [-4.033065277731541]
15 [6.780683667455127]
16 [-5.743404905351756]
17 [-13.120489014626042]
18 [7.11796592831533]
19 [-0.18102724415151528]
20 [3.7858993358168456]
21 [5.611580732228454]
22 [-23.839003437913473]
23 [-2.659716004869784]
24 [-2.412545524712864]
25 [11.176238851611473]
26 [0.8675843208921528]
27 [3.9584705093218755]
28 [1.1465740464271912]
29 [-1.7293615539129694]
30 [1.7773994092388787]
31 [-7.105433471472004]
32 [-1.4351703673626746]
33 [-1.5698588400372309]
34 [0.9426351270740201]
35 [10.323754889382641]
36 [-0.5063809836092497]
37 [8.872957948224256]
38 [-3.617719032244981]
39 [-1.662435439371683]
```

In [996]:
```
rfregressor.predict(econ.iloc[12, [3, 4, 5, 7, 9, 11, 12]].values.reshape(1, -1))
```

Out[996]:
```
array([2.97258068])
```

In [ ]:
```
econ.iloc[, [3, 4, 5, 7, 9, 11, 12]].values.reshape(1, -1)
```

In [704]:
```
LR = sm.OLS(y, X).fit()
LR.summary()
```
Out[704]:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.262
Model:	OLS	**Adj. R-squared:**	0.127
Method:	Least Squares	**F-statistic:**	1.949
Date:	Fri, 25 Sep 2020	**Prob (F-statistic):**	0.102
Time:	14:40:10	**Log-Likelihood:**	-72.455
No. Observations:	40	**AIC:**	158.9
Df Residuals:	33	**BIC:**	170.7
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Population 2019	-3.932e-09	4.52e-08	-0.087	0.931	-9.59e-08	8.81e-08
Lockdown Length	0.0223	0.021	1.043	0.304	-0.021	0.066
Density	-0.0001	0.000	-0.635	0.530	-0.001	0.000
Democrat	-4.5396	2.261	-2.008	0.053	-9.140	0.061
Republican	-5.1084	2.097	-2.436	0.020	-9.375	-0.841
Added Levels	-0.1435	0.422	-0.340	0.736	-1.002	0.715
Share_65	26.7997	12.788	2.096	0.044	0.783	52.816

Omnibus:	13.454	**Durbin-Watson:**	1.864
Prob(Omnibus):	0.001	**Jarque-Bera (JB):**	23.573
Skew:	0.788	**Prob(JB):**	7.61e-06
Kurtosis:	6.415	**Cond. No.**	4.48e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 4.48e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

In [1008]:
```
X_train, X_test, y_train, y_test = ms.train_test_split(X, y, test_size=
0.2, random_state = 0)
rfregressor = RandomForestRegressor(max_depth=2, random_state=0)
rfregressor.fit(X, y)
np.sqrt(np.mean((rfregressor.predict(X_test) - y_test)**2))
```

Out[1008]:
```
6.3562407944387225
```

In [998]:
```
y
```

Out[998]:
```
array([-2.18159879, -0.35539157, -2.92476745, -2.93690662, -6.13638926,
        3.0839055 , -4.20387385,  0.65645336, -3.10261679, 14.8710528 ,
       -4.17911353,  4.24575168,  2.42697252, -3.61651444,  2.38277479,
       -4.45891129,  4.25979085, 13.40087701, -5.79456245,  0.40751059,
       -4.3937593 , -3.79232787, 20.86150866,  4.19638493,  3.65887747,
       -4.01783013,  2.88852179, -0.55614195,  3.7078234 ,  0.3058497 ,
        2.31902801,  5.80890427, -1.01599634,  0.25991562, -0.45189116,
       -5.56879575,  2.14758381, -4.62135535,  3.44136913,  1.7367593
9])
```

In [1009]:
```
rfregressor.predict(econ.iloc[0, [3, 4, 5, 7, 9, 11, 12]].values.reshap
e(1, -1))
```

Out[1009]:
```
array([-1.43193918])
```

In [1010]:
```
econ.iloc[0, [3, 4, 5, 7, 9, 11, 12]].values.reshape(1, -1)
```

Out[1010]:
```
array([[4903185, 26, 93.53117906262518, 0, 1, 1, 0.16540024494282798]],
      dtype=object)
```

In [1017]:
```
rfregressor.predict(np.array([4903185, 60, 93.53117906262518, 0, 1, 1,
0.16540024494282798]).reshape(1, -1))
```

Out[1017]:
```
array([-0.04315836])
```

In [706]:
```
correlation_matrix = np.corrcoef(rfregressor.predict(X_test), y_test)
correlation = correlation_matrix[0,1]
r_squared = correlation**2
r_squared
```

Out[706]:
```
0.3328969578527192
```

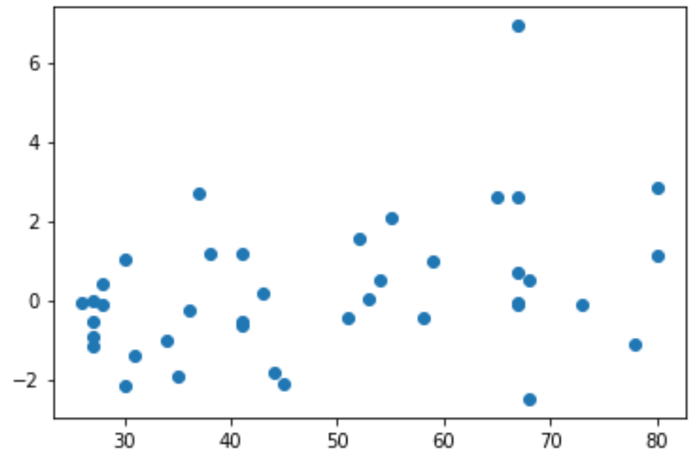
In [707]:
```
correlation_matrix = np.corrcoef(rfregressor.predict(X_train), y_train)
correlation = correlation_matrix[0,1]
r_squared = correlation**2
r_squared
```

Out[707]:
```
0.7499976542385185
```

In [708]:
```
# r2_score(y_test, rfregressor.predict(X_test))
```

In [709]:
```
plt.scatter(X.iloc[:, 1], y)
```

Out[709]:
```
<matplotlib.collections.PathCollection at 0x11f902490>
```



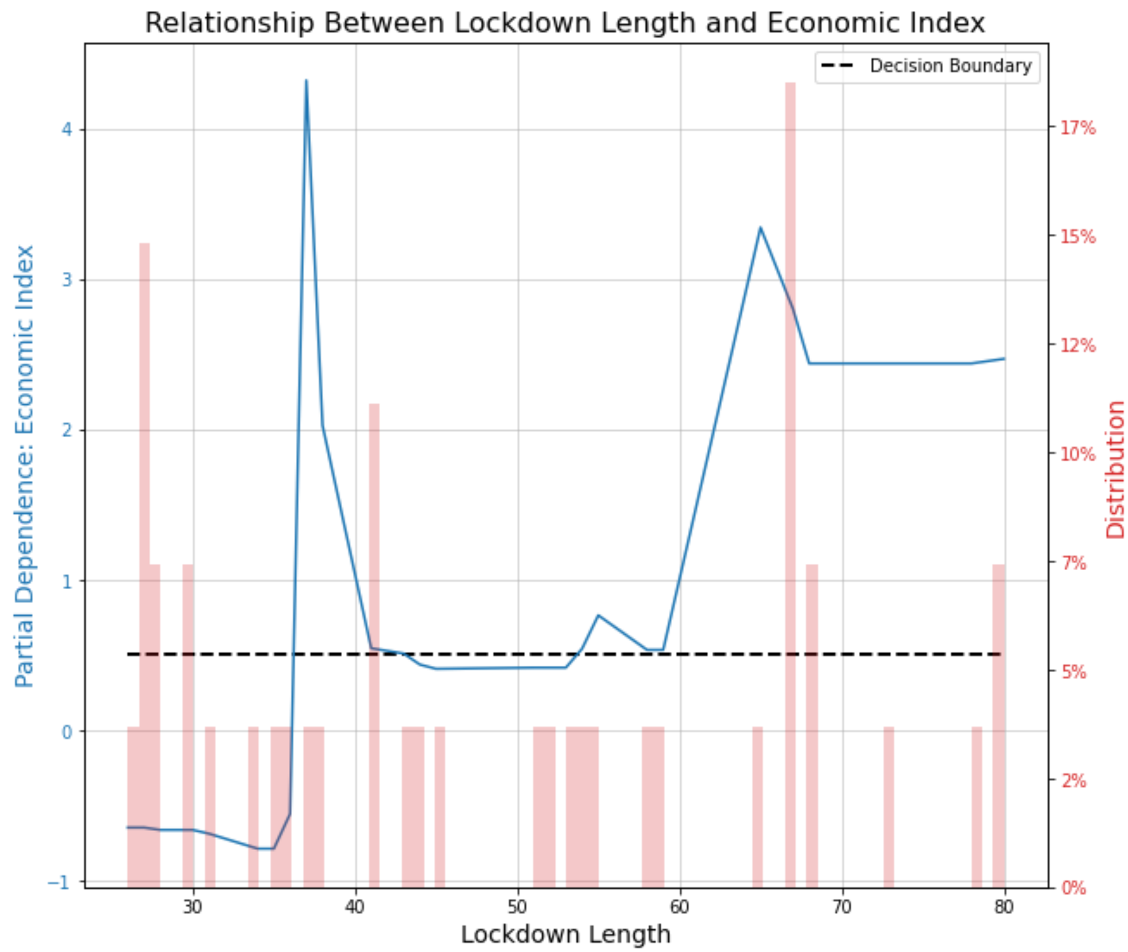
In [710]:

```
#partial dependency (added levels)
#my_plots = plot_partial_dependence(rfregressor, features=[5], X=X, gri
d_resolution=10)
```

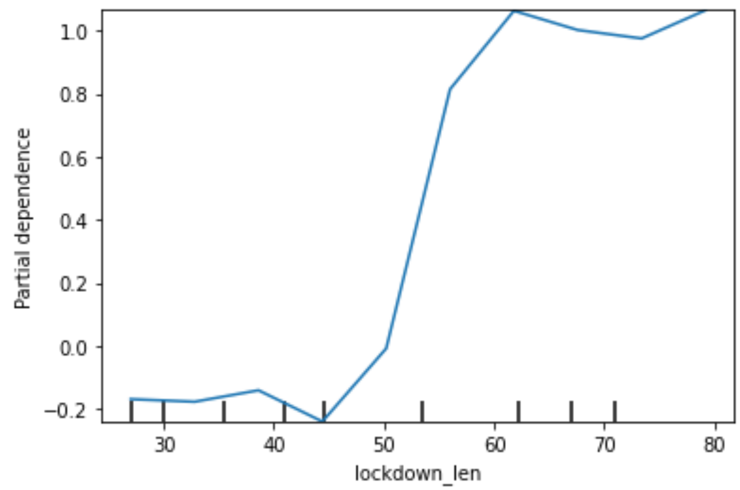
In [1004]:

```
plot_pdp(rfregressor, X, 'Lockdown Length', target='Economic Index')
```

```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:51: UserWa
rning:

FixedFormatter should only be used together with FixedLocator
```

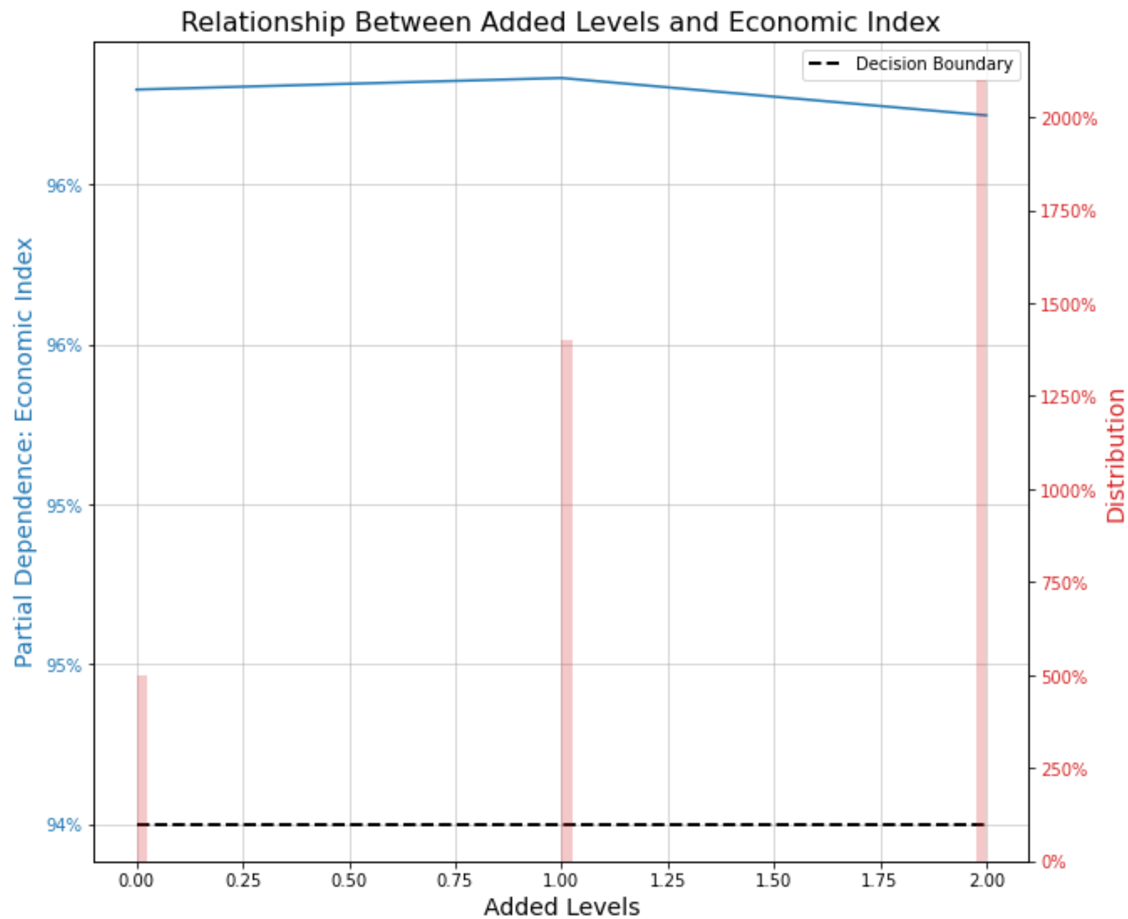


In [598]:
```
#partial dependency (lockdown length)
#my_plots = plot_partial_dependence(rfregressor, features=[1], X=X, gri
d_resolution=10)
```



In [1005]:
```
plot_pdp(rfregressor, X, 'Added Levels', target='Economic Index')
```

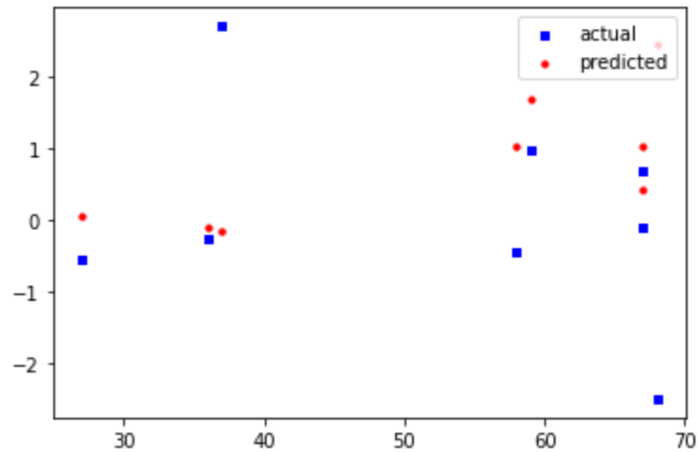
```
/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:33: UserWa
rning:

FixedFormatter should only be used together with FixedLocator

/usr/local/lib/python3.7/site-packages/ipykernel_launcher.py:51: UserWa
rning:

FixedFormatter should only be used together with FixedLocator
```



In [640]:

```
fig = plt.figure()
ax1 = fig.add_subplot(111)

ax1.scatter(X_test.iloc[:,1], y_test, s=10, c='b', marker="s", label='a
ctual')
ax1.scatter(X_test.iloc[:,1], rfregressor.predict(X_test), s=10, c='r',
 marker="o", label='predicted')
```

```
plt.legend(loc='upper right')
plt.show()
```



In [713]:

```
perm = PermutationImportance(rfregressor, random_state=1).fit(X_test, y_test)
eli5.show_weights(perm, feature_names = X_test.columns.tolist())
```

Out[713]:

Weight	Feature
-0.0026 ± 0.0118	Added Levels
-0.0053 ± 0.0516	Republican
-0.0567 ± 0.1146	Population 2019
-0.0960 ± 0.5125	Density
-0.1043 ± 0.0286	Democrat
-0.2388 ± 0.3275	Share_65
-0.4528 ± 1.1405	Lockdown Length

In [ ]: