Bacheloroppgave

IBE600 IT og digitalisering

Finding the Correlation between Features and Covid using Machine Learning

Kandidat nummer 1

Totalt antall sider inkludert forsiden:

Molde, 31.05.2022

Høgskolen i Molde
Vitenskapelig høgskole i logistikk

Obligatorisk egenerklæring/gruppeerklæring

Den enkelte student er selv ansvarlig for å sette seg inn i hva som er lovlige hjelpemidler, retningslinjer for bruk av disse og regler om kildebruk. Erklæringen skal bevisstgjøre studentene på deres ansvar og hvilke konsekvenser fusk kan medføre. Manglende erklæring fritar ikke studentene fra sitt ansvar.

Du/dere fyller ut erklæringen ved å klikke i ruten til høyre for den enkelte del 1-6:		
1.	**Jeg/vi erklærer herved at min/vår besvarelse er mitt/vårt eget arbeid, og at jeg/vi ikke har brukt andre kilder eller har mottatt annen hjelp enn det som er nevnt i besvarelsen.**	☒
2.	**Jeg/vi erklærer videre at denne besvarelsen:** • ikke har vært brukt til annen eksamen ved annen avdeling/universitet/høgskole innenlands eller utenlands. • ikke refererer til andres arbeid uten at det er oppgitt. • ikke refererer til eget tidligere arbeid uten at det er oppgitt. • har alle referansene oppgitt i litteraturlisten. • ikke er en kopi, duplikat eller avskrift av andres arbeid eller besvarelse.	☒
3.	**Jeg/vi er kjent med at brudd på ovennevnte er å betrakte som fusk og kan medføre annullering av eksamen og utestengelse fra universiteter og høgskoler i Norge, jf. Universitets- og høgskoleloven §§4-7 og 4-8 og Forskrift om eksamen §§16 og 36.**	☒
4.	**Jeg/vi er kjent med at alle innleverte oppgaver kan bli plagiatkontrollert, jf. høgskolens regler og konsekvenser for fusk og plagiat**	☒
5.	**Jeg/vi er kjent med at høgskolen vil behandle alle saker hvor det forligger mistanke om fusk etter høgskolens retningslinjer for behandling av saker om fusk**	☒
6.	**Jeg/vi har satt oss inn i regler og retningslinjer i bruk av kilder og referanser på biblioteket sine nettsider**	☒

Personvern

Personopplysningsloven

Forskningsprosjekt som innebærer behandling av personopplysninger iht. Personopplysningsloven skal meldes til Norsk senter for forskningsdata, NSD, for vurdering.

Har oppgaven vært vurdert av NSD? ☐**ja** ☒**nei**

- Hvis ja:

Referansenummer:

- Hvis nei:

Jeg/vi erklærer at oppgaven ikke omfattes av Personopplysningsloven: ☒

Helseforskningsloven

Dersom prosjektet faller inn under Helseforskningsloven, skal det også søkes om forhåndsgodkjenning fra Regionale komiteer for medisinsk og helsefaglig forskningsetikk, REK, i din region.

Har oppgaven vært til behandling hos REK? ☐**ja** ☒**nei**

- Hvis ja:

Referansenummer:

Publiseringsavtale

Studiepoeng: 15

Veileder: Ketil Danielsen

Fullmakt til elektronisk publisering av oppgaven

Forfatter(ne) har opphavsrett til oppgaven. Det betyr blant annet enerett til å gjøre verket tilgjengelig for allmennheten (Åndsverkloven. §2).

Alle oppgaver som fyller kriteriene vil bli registrert og publisert i Brage HiM med forfatter(ne)s godkjennelse.

Oppgaver som er unntatt offentlighet eller båndlagt vil ikke bli publisert.

Jeg/vi gir herved Høgskolen i Molde en vederlagsfri rett til å gjøre oppgaven tilgjengelig for elektronisk publisering:	☒**ja**	☐**nei**
Er oppgaven båndlagt (konfidensiell)?	☐**ja**	☒**nei**
(Båndleggingsavtale må fylles ut)		
- Hvis ja:		
Kan oppgaven publiseres når båndleggingsperioden er over?	☒**ja**	☐**nei**

Dato: 31.05.22

Antall ord: 10440

Preface

This is dedicated to my friends and family, who supported me and without whom this would not be possible. And to the dedicated, knowledgeable lecturers at Molde University College

Summary

Using dataset containing stringency index, life expectancy, GDP per capita and population density machine learning is used to order these features to explore how they correlate to covid cases and covid deaths. This experiment was run on 3 different machine learning models determining that the random forest model performed best at predicting new covid cases. This model determined that the stringency index was the feature that had the strongest correlation to covid cases. The poor performance scores of the models when predicting covid deaths meant no reliable conclusion could be drawn about the importance of the different features.

Content table

1.0 Introduction

For my bachelor thesis I have selected to write about machine learning. Machine learning is a big theme containing many exciting areas. I have also selected to do a practical experiment using open-source datasets. In this thesis I will go over some background about what machine learning is, the area of machine learning I will focus on and the tools and models I have used in my experiment. I will go over the method; what I did in the experiment and why I did it. Finally, I will show my results, go through and discuss them before drawing a conclusion.

In the IBE400 machine learning subject in autumn 2021 one of the bigger assignments consisted of cleaning a dataset about the passengers on the Titanic. After cleaning the dataset we used it to train a machine learning model. The machine learning model would make predictions about if a person with the different input features would survive the sinking of Titanic or not. I enjoyed this assignment very much but what got my attention was the coefficients. Using the “coef_” attribute we got a list over the different columns and next to it a number showing how important this was to the predictions. In other words for the Titanic we could see what was the most important feature for surviving the sinking of the Titanic. This got me thinking about what other uses this might have. Being 2021 we were already over a year into the Coronavirus pandemic. Not only would combining the two be very relevant and topical but the Corona pandemic also generated a large amount of open source datasets.

I devised an experiment that I hope would look into the different factors that might affect a country's effectiveness in fighting the Coronavirus. How a virus is transmitted, how it moves in a population, and what factors affect the spread is a subject all to itself that is as big if not bigger than machine learning. With this in mind I restricted myself to only the factors I already knew and that I could more easily find open source datasets on. This is partially due to time constraints, but also because this thesis is focused around machine learning. To spend part of my restricted time on something that will be secondary to the experiment would not be a good investment. In a real world case an experiment like this would most likely be a partnership between medical and machine learning experts.

The experiment consisted of gathering the necessary datasets, preparing them for use in and training of machine learning models. Using the score attribute of the model will show how well the model performed and using feature importance will show how important different features are for predicting Covid deaths and Covid cases. Several different models were trained and compared to find which of them performed best with my dataset.

For the experiment I hypothesized the following;

- Government action will be the most important feature for predicting covid cases, because this is a targeted response based on covid case trends.
- Government action will also be the most important feature for predicting covid deaths, as deaths are linked to cases.
- The decision tree will be the best performing model, because this model is less linear it is more flexible. It also features more tunable parameters which means it can handle more complex datasets.

2.0 Background

2.1 Machine learning

There is no consensus definition of what machine learning is, however most if not all contain commonalities. Perhaps the most important aspect of machine learning is making machines solve tasks that are not explicitly programmed. That is to say machine learning models can, by finding patterns in data, solve problems without explicitly being told how they should be solved. Normally machines can only execute a solution a programmer has already made, the machine automates the solution so that a programmer will not need to manually do it each time. The programmer must define and solve the problem before the machine can execute this solution to solve the problem. With machine learning on the other hand the programmer defines the problem and trains the machine using the necessary data. The machine then finds patterns in this data to “learn” how to solve the problem. This means that machine learning can adapt better than traditional machine

execution, and can also solve problems where machine execution fails or would be too time consuming.

Generally speaking there are 3 kinds of machine learning: supervised learning, unsupervised learning and reinforcement learning.

Supervised learning is when you use labeled data to train a machine learning model. Labeled data means the data has been explicitly labeled, for example the price of a house will be labeled as “house price”. This training data will consist of a pair of input/output data. The machine learning model uses this data to “learn” and when asked to produce an output from a new input it will generalize from it’s training data to predict the output pair to this input. There are many uses for this kind of model, for example facial recognition, market trend prediction, and fraud detection. For the latter the model would be trained using user data and when the real data does not match the prediction the transaction will be flagged. The advantages for the bank is that this takes less manpower and secures customer privacy. The drawback with supervised machine learning is that you need lots of labeled data. Collecting and labeling this data can be a hard, time consuming and laborious process.

Unsupervised machine learning uses unlabeled data, meaning the data does not have to be labeled by a human instead the model learns the data without human interaction. The model uses only input data, no output data needs to be provided. Instead of using input/output pairs the unsupervised machine learning model will try to relate the input data to other data points. The model will do this to find patterns in the data that it can be used for prediction. It tries to correlate data and look for features that indicate the correlation of a pair of data points. Unsupervised machine learning is harder to understand and measure the performance of because we do not know the expected output. This means the model is harder to evaluate. Unsupervised learning does come with the benefit of not needing labeled data, meaning data can be collected much quicker, however, the training times are longer. There are many uses for unsupervised machine learning. One such use is to label news articles into different categories. Another use is to condense articles without losing key points. (Deep Garg, Kamal. Khullar, Vikas. Kumar Agarwal, Ambuj. 2021)

Reinforcement learning is the last major area of machine learning. It uses a reward score to evaluate the model, then seeks to maximize this. Reinforcement learning models are complicated and hard to explain. The simplest way to explain it is by using the game of Pac-Man as an example (Bhatt, Shweta 2018). In Pac-Man the player’s goal is to collect

all the white balls and not get hit by the ghosts. On top of that there are bonus points for collecting the “white pill” that allow Pac-Man to “eat” the ghosts. Pac-Man can move up, down, left and right within a 2D board containing walls Pac-Man must move around. The reinforcement learning model is going to play a game of Pac-Man. For the first attempt the model will not know it will lose if it hits a ghost. The model gets points and an award for collecting the white balls, but trying to collect a ghost the same way results in losing and no reward or loss of reward. Next time the model will know to collect white balls and to avoid ghosts. Now to just collect balls and win might seem like what the reinforcement learning model will do from now on. This is not ideal though because to score maximum points on Pac-Man you will need to collect the white pills that allow Pac-Man to “eat” the ghosts. This model would be less useful if it never even explored this option. Therefore reinforcement learning models must try to balance the need for using what it has learned to get a reward for performing inline with expectations with the need to explore and learn more to achieve a better reward score. Like unsupervised learning models, reinforcement learning models do not need labeled data. One use for reinforcement learning is in industrial automation. With reinforcement learning powered robots' ability to adapt to their environment and learn from their experiences being very useful in some industrial roles. Reinforcement learning was also used in the AlphaGo Zero project to beat a world champion Go player (deepmind.com 2022).

2.2 The types of machine learning

My experiment used supervised learning. Within supervised machine learning there are two main types of supervised machine learning; regression and classification. (Müller, Andreas C. and Sarah Guido. 2016. page 25.)

A classification model uses labeled data input/output pairs to learn and find patterns. The model then uses this pattern to place input data into a predefined class. For example using a dataset of animals, 3 classes are defined; lizards, mammals and birds. Looking at the data the classification model might find patterns like 4 legs and no fur tends to be lizard, 2 legs and wings, will be a bird. When presented with an input of a cat, it will classify it as a mammal as from the data those tend to have 4 legs and fur. Of course there are problems with this model, because 4 legs and no fur would mean lizard but it could also be an elephant. This problem would be solved with a bigger and more complete dataset where the model could find more solid patterns.

Regression is aimed at predicting a continuous, or measurable number. An example of this would be a labeled dataset showing different factors that might affect one's income. The regression model would use these labeled input/output pairs to learn and look for patterns. When the regression model is met with new input data without an output pair, the regression model will predict the output that pairs with that input. This predicted output

will be a number in a range, for example a yearly income of 95,000 usd based on features like education and age.

For my experiment regression was the most appropriate. There are many different regression models. I tried and compared the performance of 3 different models; linear regression, ridge regression and random tree. I was also interested in whether or not they agreed about the most important features or if different models would value different features.

Linear regression is based around minimizing mean square error. Mean square error being the sum of the difference squared between the prediction and the true value of the target variable, for regression this is usually called Y (Müller, Andreas C. and Sarah Guido. 2016. page 47). Linear regression only requires the input/output data pairs. This can be a benefit as the model becomes easier to set up, but lacking any tunability might not be ideal in some cases.

Ridge regression builds on linear regression, with the addition of the “Alpha” parameter. This restricts the model in order to limit relative importance of each feature. This leads to a more complex model that might need some tuning to get the highest performance out of it. As a result of the restriction the model will most likely score worse on the training data, but higher on the test data. With less impactful features the model generalizes better, it is better at adapting to previously unseen data.

Random forest models are quite complicated. A random forest is made up of multiple decision trees. Decision trees can be used for both regression and classification. The way a decision tree works is by making rules or that have the goal of splitting the data. An example could be the earlier classification problem where data was put into 3 classes; birds, lizards and mammals. The decision tree might split them into 2: those who have wings and those who do not. This would instantly tell it if an input is a bird or not. Next it might ask if the input has scales or not, splitting lizards and mammals. In a random forest the model will train not just one decision tree but many, and each of them will be slightly, randomly, different. Decision trees tend to overfit to the training data, therefore random forest makes many decision trees and averages them to minimize overfitting while attempting to keep the predictive power of the decision trees (Müller, Andreas C. and Sarah Guido. 2016. page 83). Random forest models come with a plethora of opportunities for adjustment with numerous parameters. In my experiment I only utilized a few like n_estimators and max_features while restricting the randomness of the trees with

random_state. n_estimators is quite a simple parameter, it decides the amount of trees in the random forest. A model with n_estimators equal to 5 will have only 5 decision trees in the random forest. More trees are generally seen as better, the limiting factors are time and computational power, meaning at one point you will gain little in terms of predictive performance while spending more and more time and power to run the model. max_features tells the model how many features it should look at when spliting the data. This will affect the randomness of each tree in the random forest model. If your model is overfitting, doing very well on the training data but performing poorly on test data, lowering the max_features can dampen the overfitting. Since my experiment is a regression problem I have used “max_features=”log2”” as that is generally recommended for regression. While for a classification problem I would probably use “max_features=”sqrt” (Müller, Andreas C. and Sarah Guido. 2016. page 88). random_state is the internal randomness of what data the model uses for test and training.

For the experiment I have utilized the Pandas and the scikit-Learn (skLearn) libraries for Python, coded using Jupyter Notebook. Python was selected for its solid machine learning and data science libraries. Pandas is a Python library that is used for data handling and data analysis (pandas.pydata.org. 2020). Featuring useful tools such as the dataframe tool which restructures data and allows for more advanced data handling opportunities. Scikit-learn or skLearn is a library for the Python programming language which provides open source machine learning tools (Müller, Andreas C. and Sarah Guido. 2016. page 6).

2.3 The datasets

The first dataset I used was the “Covid-19 Government Response Tracker” dataset. This is made by Oxford COVID-19 Government Response Tracker, Blavatnik School of Government and University of Oxford. It can be found at https://www.bsg.ox.ac.uk/research/research-projects/covid-19-government-response-tracker . The aim of this dataset is to track the government response to the covid pandemic in different countries. The dataset covers the covid response of over 180 countries and the recording for the dataset started first of January 2020. The response of a given government is measured using 20 policy areas including; Restrictions on gatherings, Vaccination policy, and Fiscal Measures. These are given a value that is then

used to calculate one of five indexes; Government response index, Containment and health index, Stringency index, Economic support index and Legacy stringency index. Each of these indexes also have a second “display” version. In the display version of the index, if an index can not be calculated, for example due to missing values, this will be smoothed over by using the closest complete data to fill the gaps in the last seven days (Phillips, Toby and “tatlowhelen”. 2020.). These are then given a score, the max values vary and it is tied to levels of government action in the area. These results are put into their equation to find the index (source: https://github.com/OxCGRT/covid-policy-tracker/blob/master/documentation/index_methodology.md): index = 1kj=1kIj

Where Ij is given with the equation:

$$I_{j,t} = 100u_{j,t} - 0.5(F_j - f_{j,f})N_j$$

Where Nj is equal to the max value of the indicator, for example the C4 indicator can have any values in the range from 0 to 4, mean the Nj value for C4 would be equal to 4. and Fj indicates if the indicator has a flag variable associated with it, and will have a value equal to 1 if the indicator does have a flag variable associated with it and a value equal to 0 if the indicator is not associated with any flag variable.
and Vj ,t is the value recorded for a policy. It will always be on an ordinal scale.
and fj ,t is the value recorded for the associated indicator and will always have a value of 0 or 1.
There is also I, j, tto explain. Iis the sub index score of , j which is equal to any given indicator and twhich is any given day (Phillips, Toby and “tatlowhelen”. 2020.).

Next dataset I used is the “WHO COVID 19 global data” dataset. This contains 8 columns; Date_reported, Country_Code, Country, WHO_region, New_cases, Cumulative_cases, New_deaths and Cumulative_deaths. This dataset can be found at https://covid19.who.int/data or downloaded from https://covid19.who.int/WHO-COVID-19-global-data.csv as a CSV format file. This dataset is made by the World Health Organization (WHO). WHO is an agency of the United Nations (UN) and they are tasked by the UN to lead international public health work. To count the amount of deaths and cases from COVID-19 the WHO has relied mainly on laboratory confirmed cases and deaths (covid19.who.int. 2020). In some places this strategy of counting has been adapted to better fit the local situation. Things like the definition of a COVID-19 case, the detection time slowness in reporting cases might differ from country to country. Which of

course makes it hard to collect high quality data from all countries, so inevitably the data quality will be lower for some countries. The Date_reported column features the data that the case or death was reported which can in some cases differ from the date symptoms started showing or when the person was originally infected with the COVID-19 virus. Whilst the dataset is not perfect, there were no more suitable datasets found. The dataset is also open source and anyone can use it.

The next dataset I used was the “Life expectancy at birth (years)“. This dataset also came from the WHO. For this dataset the WHO has defined life expectancy as; “The average number of years that a newborn could expect to live, if he or she were to pass through life exposed to the sex- and age-specific death rates prevailing at the time of his or her birth, for a specific year, in a given country, territory, or geographic area.” (who.int. 2020.). The main source of their data is civil records with good cover, in other words records about any given civilian that are well kept and thorough throughout their life. To supplement this as a second data source they have used population censuses and household surveys. For some low income member states their data will be using estimation models to predict the life expectancy, therefore the data in this dataset might be different from the member states own data as different estimation models might have been used. The data in the dataset cover all 183 member states of the WHO.

I also used the “GDP per capita vs population density, 2020” dataset. This dataset can be found at https://ourworldindata.org/grapher/population-density-vs-prosperity . The dataset consists of population density, the Gross Domestic Production (GDP) per capita and the population of the country. For the population density they have aggregated data from multiple sources, all listed under “sources” on their web page. Among these are for example the UNs Department of Economic and Social Affairs’ Population Dynamics database. For GDP per capita they have used the World Bank as their source. They have also used the Purchasing Power Parity (PPP), which takes a given country’s GDP and converts the GDP into international dollars. This is done using the PPP rates. This international dollar should have the same purchasing power for GDP as the US dollar has in the United states (ourworldindata.org. 2021.).

3.0 Methodology

First thing I had to do was look for datasets with the features I thought might be useful or interesting to look at together with the Covid cases and deaths. After setting out the data I was looking for; general health, population density, GDP, stringency index and of course Covid cases and deaths, I wanted to see what, if any, connection I could find between these features. I hypothesized that a country with a healthier population might have less Covid deaths. I also wondered if GDP might affect Covid infections and deaths. A higher GDP might be an indication of a more developed economy, which might imply a healthier population with jobs that allow for flexible work from home solutions. An economy that is mostly supported by people doing manual labor or going to the market to sell their wares might not be able to have large parts of the population work from home. Conversely, an economy mostly based on people working at offices, offers more opportunity for people to work from home with access to the necessary tools to enable this, like laptops, stable internet and electricity supplies. More developed economies can also open the door for more loans and safety nets for their populations during a pandemic, which reduces stress for people and would have a health benefit. More resources also alleviates stress on the country's healthcare apparatus as their governments can afford more resources that less developed economies might not be able to afford without external loans or financial help or donations. Population density may also be a key factor that could affect the infection rate. Living in a densely populated place like Tokyo with public transport functioning as the main mode of transportation, infections might happen more easily and spread further than a more rural area where private transport such as cars are the main mode of transportation. The final factor that I wanted to investigate was the impact of government action on the spread of covid infections. My goal was to see if I could use machine learning to understand the relationship between the policies and the infections and deaths. Once I had a list of features I wanted to look at I went looking for a suitable dataset. It became clear early on that there was no single, open source, dataset that I could use for all these features. Finding good open source datasets for single features turned out to be hard enough.Ultimately, I combined the 4 different datasets, which I have discussed earlier in this paper.

First I started with the stringency dataset. I imported it into a Jupyter Notebook document and created a dataframe using Pandas' dataframe feature. Mixed types in the dataset means it was recommended to turn low_memory to False. The Stringency dataset contained 285204 rows and 61 columns (the dataset is still being updated so the amount of rows will change. After inspecting the dataset I found that only 3 columns were of interest to my experiment ("CountryName", "Date" and "StringencyIndex"). Covid deaths and cases were part of this dataset but the numbers were not per day and instead a cumulative/total.This way of measuring cases and deaths is not suited for my purpose, because the total would not correlate with the features, that is a high cumulative covid case count would appear to not correlate with the stringency index, as government response is based on the current caseload. The cumulative case count would only strongly correlate with time/progression of the pandemic.

Taking only the columns that were useful to me left me with a 285204 rows by 3 columns dataset. Limiting it further, instead of an end date of 23rd of April 2022, I limited it to between 1st of January 2020 and 1st of January 2021. This leaves me with 114915 rows - each row representing one day for one reporting area, which is sufficient data. Limiting the dataset this way also eliminates potential confounding factors such as vaccination. Limiting the dataset also meant finding corresponding datasets with the other features I wanted to include would be easier, as data for only 2020 would be needed. Further inspection of this dataset revealed that some countries, such as the US, had multiple entries per day. Different states within the US had different policies in place and even different ways of counting and defining covid cases and covid deaths, therefore the dataset included these multiple entries to better reflect these differences. To solve this problem I used the ".groupby" feature and grouped my data on country and date. Meaning, for example, only one row of Belgium 6th of June 2020 could exist. The stringency index would be added together and the mean value calculated. This might not be the perfect solution as these did not account for population size, however as the original dataset could not account for all differing regions of all counties. For this experiment I think this is sufficient though I would have liked to look more into different techniques to solve this problem had I had the time allowance for this. With all that done I was down to 68629 rows and 3 columns and I moved on to the next dataset.

Moving on to the global Covid dataset from WHO. This dataset is somewhat smaller at 179409 rows and 8 columns. Only 4 of the columns contain data that is useful to the experiment. I then make a new dataframe with only the columns that are useful;

"Date_reported", "Country", "New_deaths" and "New_cases". Before I can combine this with the earlier dataset a problem presents itself. The columns counting new Covid cases and deaths are of the string type and are therefore not numbers that can be used to perform mathematical operations. Why this is the case I do not know, but this emphasizes the need to check data due to there being no universally accepted standard format. Pandas features functions that can change the type, known as the Dtype, of a column. Using the function that transforms strings to integers resulted in an error code because there were symbols that could not be converted. Upon investigation, the dataset contained negative numbers for covid deaths and cases. Most probably this comes from the need to correct for previous errors in reporting that have subsequently been corrected. This turned out to be a harder challenge to overcome than it first appeared as dropping the rows containing negative numbers was impossible as the columns contained text and therefore no negative numbers were found. Since a special character was used instead of a dash, I attempted to drop all columns containing this. This was also ineffective and in lieu of a better solution I manually dropped each number as it came up in the error code. These negative numbers would not have helped my machine learning model anyways. With more time and if it was practically possible I could have run the model twice, once with the negative numbers in and once with them out. I could compare the two and see what effect this had on the model. After that I limited the data range from 1st of January 2020 and 1st of January 2021. I also grouped the data on country and date so only one country and date combination would exist. Once this was done I renamed the "Date_reported" and "Country" columns to "Date" and "Country_name" respectively. This is done so that the names would match the "Mainset_df" dataframe. Then I could easily merge the two dataframes on "Country_name" and "Date". Due to the result of the merge being limited to the overlap between the two dataframes being merged the Mainset_df now stands at 59032 rows and 5 columns.

Moving on to other WHO dataset. This dataset contains life expectancy, both from birth and at age 60, from different years.The dataset is also shared into "Male", "Female" and "Both sexes". I was interested in the life expectancy since birth, for both sexes and I was only interested in the year 2019. This was the closest to 2020 that I could find and I assumed that the change from 2019 to 2020 would be minimal, it is also likely that the pandemic itself will have had a negative impact on life expectancies. It is a comparatively lean dataset with only 4393 rows and 34 columns. I limited the "Period" to 2019 and I limited "Dim1", which holds the variable for sex, to "Both sexes". Then I found that "IndicatorCode " held the key to only get life expectancy at birth in years so I limited this to only allow data equal to "WHOSIS_000001", giving me life expectancy since birth. Now

the dataset contains only life expectancy, both sexes, the year 2019 and country. Only 2 of these are really needed, the column for sex only contains "Both sexes" over and over and the same is true for the "Period" column, containing only "2019" over and over. I dropped both of these columns. Leaving me with a 183 rows and 2 columns dataset containing only "Location" and "Value", or country name and life expectancy. I renamed the columns, "Location" became "CountryName" and "Value" became "LifeExpectancy". Merging this dataset with the Main_set_df resulted in a dataset of 55026 rows and 6 columns. Because it was only merged on one column this means every column with a country that matches across the datasets would get life expectancy written in for every entry.

Before the machine learning part of my experiment, I had one more dataset to clean and merge. I started by importing the "Population Density vs Prosperity" dataset. This dataset contained 74296 rows and 7 columns with data points like "Year" with a span from -10000 to 2100. As shown this data is not only historical but also predictive. I am of course only interested in the data from the year 2020. So that was the first step of processing to extract all the columns that contain data from 2020. Next up I renamed the columns from "Entity" to "CountryName". "population_density" became "PopulationDensity" to fit the overall style of the main dataset better. "GDP per capita, PPP (constant 2017 international $)" was changed to "GDPperCapita" to make it fit the style of the mainset but also to make it more readable and clearer. Then I dropped all the columns that I felt were not needed. This leaves a dataset of 287 rows and 3 columns; "CountryName", "PopulationDensity" and "GDPperCapita". Because the dataset used "Entity" and not country, my column "CountryName" now contains entries like "World" which is not a country. This would be solved in a merge on the "CountryName" column. To make sure there was no problem with this I made a list from the "main_set_df" using the values from the "CountryName" column containing all the unique country names in it. I transformed it to a list type and sorted it, this was just for human readability, making it easier for me to check that the top and bottom part of the list matched the dataframe. The process of making this list was automated, if there was an error it would most likely be me, the human’s, fault. Therefore if the top and bottom few matched chances are good the rest match too. Then I made sure there were no duplicate entries by using the "set" attribute of lists. I then said I wanted a dataset that is only the rows that have a country from the checklist in the "CountryName" column. Merging this on "CountryName" with the "main_set_df" dataframe result in a dataset of 54661 rows and 8 columns.

Before starting any machine learning I applied the "dropna()" function to the "main_set_df". This removes all the rows with Not a Number (NaN) values. These values would be useless when presented to the machine learning model and in fact the machine learning model would reject any dataset with NaNs in it, meaning that this step is crucial to being able to progress. This left me with 50287 rows and 8 columns.

Moving on to the machine learning part. I started off by splitting the dataframe into a y dataset that would contain the target variable - as I elected to have two different target variables (cases of Covid infection and cases of deaths resulting from Covid infection) and a X dataset containing all my features and dropping new cases and new deaths. After that I applied normalization. I used min-max normalization. I applied this to the other part of the split that contains only the features. The feature split contains the columns "StringencyIndex", "LifeExpectancy", "GDPperCapita" and "PopulationDensity". The y split and the X split is then split again into two parts; one that is 70% of the data and one that is the remaining 30%. Doing this I end up with a "test" and "train" each set with features and a different set with the target variable. After that I imported the LinearRegression() function and fit the training sets into the linear regression model. Running the model I then printed the R2 score. This R2 score works like an accuracy score telling me how well the model has performed at predicting the target variable. I repeated this with the other target variable and printed this second accuracy score. These are reported in the results section below.

For the ridge regression model, I first imported Ridge from sklearn. I created an instance of the model and fitted the datasets into it. I initially ran the model without modifying the alpha value to create a baseline prediction score. Unmodified it returned an accuracy score of 0.03231. The linear regression model yielded a score of 0.03216. The difference of 0.00015 is not a significant amount better at predicting. The benefit Ridge has over linear regression is that it has an alpha value that can be modified. I first set the alpha value to 10, this improved the prediction to 0.03231. Whilst not a big improvement this was going in the right direction. Since the improvements with 10 as alpha value were relatively small I checked to see if going the other way to 0.1 would yield a better result. Setting the alpha value to 0.1 and running the model again gave a result of 0.03217. Scoring better than without modifying the alpha parameter but only 0.00001 which is a very small amount and lower than with the higher alpha set to 10. Trying again with an alpha value of 0.01 and another with the alpha set to 100. Resulting in a score of 0.03216 and 0.03509 respectively. Seeing the pattern that higher alpha values seem to yield the best prediction, I tested two more with an alpha values of 1000 and 1500. Resulting in

scores of 0.03104 and 0.02891 respectively. These scores are worse than the other alpha values I have tried and even perform worse than the linear regression. Seeing that an alpha value of 100 has so far yielded the best prediction and over 1000 led to the worst prediction I then tried to identify the best alpha value. Trying to identify ceiling values I tried an alpha value of 500, leading to a score of 0.03348. Better than 1000 and 0.1 but still not as good as 100. Halving the alpha value to 250 further improves prediction with a score of 0.03474. Still not performing as good as an alpha value of 100. Lowering the alpha value further to 150 yields 0.03508. Closing in on the score of the alpha value 100. Narrowing it down to a value of 125 does yield a better score of 0.03511. Doing an alpha of 130 resulted in a score of 0.03510. I concluded that the best alpha value is 125 with a performance score of 0.03511.
I repeated this for the other dataset too, with the only difference being the target variable. This returned an accuracy score of -0.04261 with alpha set to 1, 0.1, 10 and 0.0001. Meaning this model did not overfit therefore working to lower the overfitting had no effect on the accuracy scores. An accuracy of -0.04261 is very low.

For the last model I used a random forest model. This model makes multiple decision trees that make up rules that split the dataset with the goal of either predict or classify new input data. Importing the RandomForestClassifier and fitting my dataset into the model. The main parameters of the model are n_estimators and max_features. n_estimators decide how many trees will be made and max_features decide the randomness of the trees. I ran n_estimators in a range from 5 to 15000. The general rule is that the more trees you have the better the model will perform. Of course more trees mean more computing power is needed and you will meet a wall where the gains are getting small and the time and resource demands are rapidly increasing. At 15000 trees the time the model took to make a prediction was too long. I also found that the best accuracy score was found at 575 trees. This was for when new cases of covid were the target variable. For when new deaths as a result of covid is the target variable I ran the model in a range from 5 to 5000. I found that the best performance was found at around 1000 trees.

4.0 The results

For my experiment there are two sets of results, one consists of the ranking of different features, by importance, and the other consists of the performance score for each model.

The table below displays the feature importance of the three different models used in the experiment over the two different target variables; new covid cases and new covid deaths.

New Covid Cases	**Random Forest Model**	**StringencyIndex**	**0.458488**
		LifeExpectancy	**0.264681**
		PopulationDensity	**0.168981**
		GDPperCapita	**0.107850**
	Ridge Regression Model	**StringencyIndex**	**2467.668404**
		LifeExpectancy	**2991.040142**
		PopulationDensity	**990.241535**
		GDPperCapita	**-2090.821880**
	Linear Regression Model	**StringencyIndex**	**2523.689077**
		LifeExpectancy	**3511.541432**
		PopulationDensity	**3286.628769**
		GDPperCapita	**-2847.324158**

New Covid Deaths	**Random Forest Model**	**StringencyIndex**	**0.352123**
		LifeExpectancy	**0.330217**
		PopulationDensity	**0.152018**
		GDPperCapita	**0.165642**
	Ridge Regression Model	**StringencyIndex**	**68.155307**
		LifeExpectancy	**82.434175**
		PopulationDensity	**-20.010521**
		GDPperCapita	**-50.280563**
	Linear Regression Model	**StringencyIndex**	**69.868979**
		LifeExpectancy	**100.463720**
		PopulationDensity	**-78.974417**
		GDPperCapita	**-70.891163**

The table below displays the performance score of each of the different models used in the experiment over the two different target variables; new covid cases and new covid deaths.

New Covid Cases	**Linear Regression Model**	**Test score**	**0.03216**
		Training score	**0.03344**
	Ridge Regression Model	**Test score**	**0.03511**
		Training score	**0.03285**
	Random Forest Model	**Test score**	**0.71898**
		Training score	**0.91012**
New Covid Death	**Linear Regression Model**	**Test score**	**-0.04261**
		Training score	**0.05497**
	Ridge Regression Model	**Test score**	**-0.04261**
		Training score	**0.05497**
	Random Forest Model	**Test score**	**-0.02065**
		Training score	**0.88456**

5.0 Discussion of the results

First I will discuss the performance results of the different models. I have used the ".score" attribute to show the accuracy of each of the models. This score is a result of the difference between the actual value of the output and the model's predicted output. The best accuracy score will be 1.0. The accuracy score can be negative as a model can be very bad at predicting. One reason is that the model does not fit the trend of the dataset, or the dataset does not have a clear trend or pattern, which would result in a negative value for the accuracy score.

If the model is good at predicting the target variable when faced with new input data then the model is good at generalizing. Overfitting occurs when the model gets too focused on individual data points, and generally indicates that a model is too complex. This leads to a good or even perfect score on the training dataset but a poor score when faced with new data that differs from the training set due to poor generalization. Underfitting is the opposite in many ways and occurs when the model is too simple. This will lead to the model performing poorly both on generalization and on the training dataset. We can see that our model is underfitting when the test and training accuracy scores are very close. For models with tunable parameters there will be a point where the generalization reaches a peak; beyond this point the training dataset accuracy score might improve but the generalization will start dropping off. This tipping point is the ideal spot for best performance.

The least complex of the models I have used is the Linear Regression model. The linear regression model has no tunable parameters. Being a simple model it has the benefit of being easy to use. The computation power needed to execute the model is also relatively low. When the target variable is new covid cases the linear regression model scored a 0.03216 on the test dataset and a score of 0.03344 on the training dataset. The first thing that becomes clear is that we likely have a case of underfitting. The other thing that is clear is that a generalization score of 0.03216 is a very poor performance score. For the other target variable, new death cases, the accuracy score for the training dataset is 0.05497 and an accuracy score of -0.04261. For the training set we can see that the model performed better than with the "new covid cases" target variable but performed

significantly worse on the test dataset. Looking at the difference between the test score and the training score it does look like with this target variable the model tends toward overfitting however it is still a poor overall score. The reason for the poor scores in both cases is probably because the data is not linear or that the dataset fits the linear model poorly. The way a regression model works is that it makes a line for a single feature, here we have more features so it makes a hyper-plane, a multidimensional plane that is always a dimension down on its environment. Predicting complex and nonlinear data like in this dataset becomes very difficult using this method.

Next, the ridge regression model. This model is similar in many ways to the linear regression model but it features a parameter known as “alpha”. This alpha parameter applies L2 regularization to the regression model. The goal of this is to make the features have as little impact on our model as possible while still allowing for good generalization. This can remedy overfitting and result in better performance if the parameter is tuned correctly. For new covid cases this model scored 0.03285 for the training dataset and 0.03511 for the test dataset. We can see that the generalization is better for the ridge model. We can also see that the training score is about 7% lower than the test dataset. Reasons for this might include random differences in the dataset when I split the dataset leading to the model working better on the test dataset. The difference however, is not significant enough to show a clear error in the dataset or how the dataset works with the model, but it is peculiar and something that I would look more into with a more generous timetable at my disposal. When it comes to performance we can see that compared to the linear regression the ridge regression predicts about 10% better but the accuracy score is still very low. For the model with the new deaths as its target variable the training accuracy score is 0.05497 and -0.04261 for the test dataset. Peculiarly this is the exact same accuracy scores as the linear regression model for both training and test datasets. This points to this being the best linear regression possible for this target variable with this dataset. This confirms that this dataset is not well suited to linear models generally but especially when “new deaths” is the target variable. Having tried different alpha values I found that not only were 0.05497 and -0.04261 the best I could get but this was also a common number I got when running the model with a low alpha score. Both 10, 1, 0.1 and 0.00001 returned these values for accuracy scores. In this case ridge regression offered us no real advantage over linear regression but on other datasets this would be different.

Finally we have the random forest regression model. This is the most complex model used in this experiment and features many tunable parameters. The way this model works is to make many decision trees with small random changes to them. All these trees try to

predict the target variable. Decision trees will generally overfit to the training data which leads to less generalization. When we have multiple trees we can find an average of all the trees. This aims to reduce overfitting while keeping the prediction power. For a complex dataset such as the one used in this experiment I assumed this model would give the best accuracy score. When new covid cases were the target variable the random forest model returned a training accuracy score of 0.91012 and a test accuracy score of 0.71898. That is a *significant* increase in performance. Compared to the ridge regression model on the same target variable the random forest model performed over 20 times better. This supports the idea that this dataset is a poor fit for linear models. When new covid deaths is the target variable the training accuracy score is 0.88456 and the test accuracy score is -0.02065. This is a sign of overfitting as we see the training score is good, and much higher than the test score, which being negative indicates a very poor performance with the test data. With more time I would look into tuning more parameters, especially max_depth, to see if the overfitting can be remedied. There is also a chance that this would not improve the score significantly because it might be that the dataset just works poorly against the new deaths target variable. It might also be the case that new deaths do not strongly correlate with any of the features I tested.

Having shown the performance result of the three different models I will now show the order of which feature each model thought was the most important for predicting the target variable. This will be shared in two; the first set of results will use new cases of covid as their target variable and a second set of results will use new covid deaths as their target variable.

For new cases as target variable. The linear regression model selected the order, from most to least important, as; LifeExpectancy, PopulationDensity, StringencyIndex and GDPperCapita. This means that LifeExpectancy is the most important feature to use when predicting new covid cases. Which can be interpreted as this being the most important way to avoid covid cases. If this is the feature most linked to new covid cases it stands to reason that if you have a favorable life expectancy that is the best way to avoid covid cases. Both population density and life expectancy is given weight here over political policies to stop the spread of covid (StringencyIndex). When looking at these results though it is important to remember that this model scored poorly on its accuracy score. Next we have the ridge regression model, it ordered the features, from most to least important, as; LifeExpectancy, StringencyIndex, PopulationDensity and GDPperCapita. This model again points to life expectancy as the most important feature though this time government policy is not only the second most important feature but also much closer to

life expectancy in importance. This model also suffers from a low accuracy score. The random forest regressor model ordered the features, from most to least important, as; StringencyIndex, LifeExpectancy, PopulationDensity and GDPperCapita. This model indicates that short term, targeted government policy to stop the spread of covid was more important than more long term policy leading to longer life expectancy through investment in general population health and GDP per capita through long term good financial policies. The model indicated that these policies were also more important than more natural features of a country like population density. The random forest regressor model also has a significantly better accuracy score than the other models in this experiment. Therefore this ranking of feature importance will be the one I use to draw a conclusion.

Feature Importance	**Linear Regression**	**Ridge Regression**	**Random Forest**
#1	LifeExpectancy	LifeExpectancy	StringencyIndex
#2	PopulationDensity	StringencyIndex	LifeExpectancy
#3	StringencyIndex	PopulationDensity	PopulationDensity
#4	GDPperCaptia	GDPperCapita	GDPperCapita

For new covid deaths as target variable. The linear regression model ordered the features, from most to least important, as; LifeExpectancy, StringencyIndex, GDPperCapita and PopulationDensity. This ordering does make some logical sense. From this data we can see that GDP per capita has moved up a spot, being more important for if you die of covid or not. It stands to reason that richer countries might have better access to more advanced healthcare and might have more resources to help its healthcare apparatus cope with the influx of covid cases leading to less deaths. However, again it is life expectancy and government policies which are most important to stop the spread of covid according to the linear model, just as it was with the new covid cases. Again the accuracy score of this model is very poor, and it is difficult to reliably draw a conclusion from this model.

The ridge regression model ordered the features, from most to least important, as; LifeExpectancy, StringencyIndex, PopulationDensity and GDPperCapita. Here population density and GDP per capita have swapped places. Life expectancy and government policy to stop the spread of covid keep the top spots with the only difference being that

they are now closer in importance. This model is again plagued with very poor accuracy scores.
The random forest regressor ordered the features, from most to least important, as; StringencyIndex, LifeExpectancy, PopulationDensity and GDPperCapita. Here we see government action beating life expectancy, even if they are now close together. Using these numbers we can draw the conclusion that robust government action is the best way to limit covid deaths with the general health of the population as a close second indicator. This model did score better on accuracy then the other models but it is still very low.

The accuracy scores for all three “new deaths" models are so low that they can not be reliably used to draw a conclusion.

Feature Importance	**Linear Regression**	**Ridge Regression**	**Random Forest**
#1	LifeExpectancy	LifeExpectancy	StringencyIndex
#2	StringencyIndex	StringencyIndex	LifeExpectancy
#3	GDPperCapita	PopulationDensity	GDPperCapita
#4	PopulationDensity	GDPperCapita	PopulationDensity

5.1 Possible improvements

There are many things I would have done differently with a more generous time allowance and with the knowledge I have gained from executing this experiment. There are general things like doing more research into statistical methods to enhance the usefulness of my dataset for linear regression models. Both the tree based random forest and the linear models would benefit from the features, and in the case of linear models the target variable too, being Gaussian distributed. Gaussian distributed values will have a “bell” shaped curve. Changing the distribution of the features, and target variable, can be done with mathematical operations such as applying log to them.

Another performance enhancer is to use a binning technique. Linear models are limited to only linear relations. This can be overcome by dividing input features into bin. This

increases the flexibility of the linear model. Now the model only needs to find linear relations within the bin. The model will have to make more predictions but the model would make better predictions.

Another idea I had a bit too late in the process was to group the data. Because the stringency index, indicating government response to the covid pandemic, does not follow new cases and new deaths from covid 1 to 1. It is not like schools were closed yesterday but they are open today due to less new covid cases and deaths. The reason for policy change comes from seeing the bigger picture and seeing the trends in covid cases over time. The reason for policy change on day x was not day x's covid cases but the development of cases from day x-14 to x-1 for example. Knowing this I wondered how grouping the cases by when the stringency index changed, leaving GDPperCapita, population density and life expectancy untouched but adding together all the covid cases and deaths resulting from covid will affect the models. I do not know that this would improve performance or improve our ability to draw conclusions about what is the most important feature for covid cases, but I think it would have been interesting to try had the time allowed. I would also like to explore statistical functions that could account for the rate of change in the dataset rather than the day by day.

I also wonder if I could have made this a classification project instead, by making each class a range of covid cases or covid related deaths. The models would predict which class a given input should be placed in. Whilst this would decrease the accuracy of the output of the model by only fitting to the predefined ranges, this would improve confidence in the model due to the higher reliability. I would have been interested to see how this would affect the predictions and the feature importance.

With the benefit of hindsight I would also have run the target variable as new deaths one more time, except this time I would include "new cases" as one of the features. I would also want to see the difference when I ask the models to predict both new deaths and new cases. New deaths, quite logically, are more tied to new cases. With how poorly all the models performed with this target variable I think it would be interesting to compare without new cases as a feature and with it included. I assume most people who die from covid do not die the day they get it, though some cases might not be registered until close to their death being logged. Therefore I do not know if this really would make a big difference more so when the performance of the models is so poor. A big increase in performance is needed and I am not sure this change would be enough.

There are also more features like the average age of a citizen in each country as younger people were shown to have a much better chance at surviving. Therefore this would be a feature that could have been of interest in this experiment.

There is also something to be said about the datasets I have used. They are of good quality all over, and being open source they are free and available for anyone to analyze. Yet there are parts where estimations and poor quality of data collected on the ground have been used. With covid cases there are problems like delay in reporting, or the case not being serious enough to be discovered. Different test strategies in different countries will almost certainly affect the quality of the data. I do not think that there are better datasets that are open sourced, and the datasets do have reputational sources backing them, however the lack of any better data sources does not resolve some of the inherent issues with covid data.

6.0 Conclusion

To conclude I have purposed and executed an experiment with two purposes; the first purpose was to compare different machine learning models to see how they compare when it comes to the accuracy of the model when performing a prediction. The second purpose was to see which of the features (government action, life expectancy, GDP per capita, and population density) would be most important to predict new cases of covid infection and deaths resulting from covid infection.

The result of the first part of the experiment was that the random forest model had the best accuracy when predicting new cases of covid infection and when predicting deaths as a result of covid infection. I therefore choose to use this model to determine the most important features. I did this because no other model had a high enough prediction accuracy to be valid. For the random forest model this is only true when the target variable was new cases of covid infection, for when the target variable was new deaths as a result of covid infection even the forest model did not perform well enough to be considered. My thought is that this might be a result of the deaths not being connected to the features in the same way as new cases of covid are. Many new cases might mean a higher stringency index as the government rush to stem the spread of covid. While deaths are usually happening some time after an infection and the government is probably not using this to guide their policies. In the end this left me with only the random forest model for new cases of covid infection to use to look at the features most important to predict new covid cases. The random forest model ordered the features, from most to least important, as StringencyIndex, LifeExpectancy, PopulationDensity and GDPperCapita. This would indicate that the short term government policy aimed at stopping the spread of covid, measure as stringency index, is the most important feature to consider when you want to stop covid cases.

To conclude on my hypothesis I was correct that the best performing model was the random forest model. I did not expect the random tree model to outperform the other models by quite such a high margin, but random forest coming out on top was not really surprising. For the feature importance I was again correct that government action would be the most important feature. My hypothesis was that government action would be the

most important no matter the target variable. It ended up being but for covid deaths as target variable the accuracy scores were so low I will conclude that the result is inconclusive.

Bibliography

Deep Garg, Kamal. Khullar, Vikas. Kumar Agarwal, Ambuj. 2021. "Unsupervised Machine Learning Approach for Extractive Punjabi Text Summarization".

Müller, Andreas C. and Sarah Guido. 2016. "Introduction to Machine Learning with Python A Guide for Data Scientists". Sebastopol:O'Reilly Media.

Pandas.pydata.org. 2020. "About Pandas". updated: 30.05.2022. URL: https://pandas.pydata.org/about/

Phillips, Toby and "tatlowhelen". 2020. "index_methodology.md". github.com. URL: https://github.com/OxCGRT/covid-policy-tracker/blob/master/documentation/index_methodology.md

covid19.who.int. 2020. "WHO Coronavirus (COVID-19) Dashboard". Updated: 31.05.2022. URL: https://covid19.who.int/data

who.int. 2020. "Life expectancy at birth (years)". Updated 30.05.2022. URL https://www.who.int/data/gho/data/indicators/indicator-details/GHO/life-expectancy-at-birth-(years)

ourworldindata.org. 2021. "GDP per capita vs population density, 2020". Updated: 30.07.2021. URL: https://ourworldindata.org/grapher/population-density-vs-prosperity

Bhatt, Shweta. 2018. "Reinforcement Learning 101". towardsdatascience.com. URL: https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292

deepmind.com. 2022. "AlphaGo". updated: 30.03.2022. URL: https://www.deepmind.com/research/highlighted-research/alphago