**The Brave New World of Living with Covid-19**

Circle

Description automatically generated

1. **Project Overview**:

This project was initiated to satisfy the requirements of the “Final Project” assignment for the Monash Data Analytics Bootcamp.

These requirements are as follows.

Table

Description automatically generated

Graphical user interface, text, application

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

1. **Team Members**: Megan Greenhill

Hesh Kuruppuge

Jacqueline Xia

Mike Murphy

1. **Brief Project Description**:

The project uses the machine learning approach to create a model for analysing and forecasting the Covid-19 Pandemic.

The project:

* **evaluates various regression model techniques** to find the **optimal regression model** for analysing and forecasting Covid-19 - Confirmed Cases, Active Cases, Recovered Cases and Deaths from the **John Hopkins University (JHU) time series data sets** which are published daily,
* uses **time series analysis** and the **optimal regression model** identified above to produce forecasts of Covid-19 - Confirmed Cases, Active Cases, Recovered Cases and Deaths,
* uses **Tableau** to read the forecasts produced above and **apply its exponential smoothing for forecasting and plotting visualisations** of Covid-19 - Confirmed Cases, Active Cases, Recovered Cases and Deaths,
* compares the results of the **optimal regression model** to the results produced by **Tableau,**
* **summarises the project results / conclusions**,
* performs Extract, Transform and Load to extract Covid-19 data from the **John Hopkins University (JHU) time series data sets**,
* combines the data sets into a **single integrated table in a PostGreSQL data base**,
* uses a **Python Flask-powered API to access the integrated PostgreSQL database table** and:
  + use a **Python library called “psycopg2” to extract the table and create a JSON file**,
  + assigns each column of the database table to a dictionary,
  + **JSONifies the dictionary**,
  + **returns the JSON dictionary** through the app,
  + the app is then called in the **Java Script to create visualisations**;

1. **Project Rationale**:

The project:

* **satisfies the requirements for the Final Project for the Monash Data Analytics Bootcamp**,
* provides an approach for fine tuning the **optimal regression model** identified as future data becomes available,
* provides a **model for visualising current and future Covid forecasts**, and
* provides a **useful tool for further development of analysis and forecasting capability**.

1. **Project Methodology**:
   * **Literature Review**

An extensive literature review was conducted to identify work that may have done in the area of analysis and forecasting Covid-19 time series analysis data.

A complete list of the reviewed literature is provided in Section 12.

* + **Citations**

From the reviewed literature three articles in particular were chosen for detailed analysis and citation purposes.

They are:

* **[1]** **“Analysis and Prediction of Covid-19 using Regression Models and Time Series Forecasting”** – Authors: Saud Shaikh; Jaini Gala; Aishita Jain; Sunny Advani; Sagan Jaidhara; Mani Roja Edinburgh which can be found at the following link:

<https://ieeexplore.ieee.org/document/9377137>

This article describes the evaluation of Linear and Polynomial Regression Models for

forecasting future Covid-19 cases from a data set of some 7 months cases in India.

* **[2]** **“Polynomial Regression” – Author: - Animesh Agarwal**

<https://towardsdatascience.com/polynomial-regression-bbe8b9d97491>

This article is part of a series of blogs published by the author describing Polynomial

Regression and ways of achieving the best fit of the Regression line to the data. It provides

sample code to demonstrate how to minimise the effects of over-fitting and under-fitting the Regression line to the data.

* **[3]** **“Lasso , Ridge & Elastic Net Regression: A Complete Understanding (2021)”**

– Author: - Rohit Bhadauriya

<https://medium.com/@creatrohit9/lasso-ridge-elastic-net-regression-a-complete-understanding-2021-b335d9e8ca3>

This article provides an excellent explanation of Regression and ways of achieving the best fit of the Regression line to the data sets. It also provides sample code to demonstrate how to minimise the effects of over-fitting and under-fitting the Regression line to the data.

* + **Extracting the Data**

The Extract phase uses **urls / wget downloads in place of API calls** are APIs are not available for the datasets needed. The JHU time series data sets were retrieved using this method.

* + **Transforming the Data**

The detailed description of the **Data Transformation** is covered in section 11*.* It is too granular to be presented here. It covers **both data cleansing and data transformation** and does the typical:

* removing unwanted or duplicate data,
* fixing structural issues,
* handling missing data,
* removing outliers,
* providing a quality assurance check on the data prior to regression analysis.
  + **Loading the Data**

*Summarise data load to db here*

* + **Evaluation of LinearRegression Models**
* **LinearRegression**
  + **Lasso**
  + **Ridge**
  + **ElasticNet**
* **Lasso, Ridge** and **ElasticNet** are models used to minimise the errors of overfitting and underfitting which can occur when applying regression to data sets.
* There are two methods of overcoming overfitting:
  + reducing the model complexity, and,
  + regularisation which tries to improve the accuracy of the model.
* Regularisation is where Lasso, Ridge and ElasticNet come into play.
* Lasso – the Least Absolute Shrinkage and Selection model aims to overcome overfitting by applying the penalty L1 which is the sum of the absolute value of the beta coefficients of the quadratic equation that describes the line of best fit.
* Ridge - the model aims to overcome overfitting by applying penalty L2 which is the sum of the square of the magnitude of beta coefficients of the quadratic equation that describes the line of best fit.
* ElasticNet – combines the techniques of Lasso and Ridge to get the best of both worlds.
* ***[Please refer to: - [3] -* “Lasso , Ridge & Elastic Net Regression: A Complete Understanding (2021)”** ***for a detailed explanation of these concepts.]***
* *Summarise evaluation here*
  + **Evaluation of Polynomial Regression Models**

The **“Polynomial Regression” article** **[2]** deals with the issue of choosing an optimal model. To answer this question we need to understand the bias vs variance trade-off.

**Bias**refers to the error due to the model’s simplistic assumptions in fitting the data. A high bias means that the model is unable to capture the patterns in the data and this results in **under-fitting** the model to the data points.

**Variance**refers to the error due to the complex model trying to fit the data. High variance means the model passes through most of the data points and it results in **over-fitting** the data to the data points.

Ideally, a machine learning model should have **low variance and low bias**.

But practically it’s impossible to have both. Therefore, to achieve a good model that performs well both on the training and unseen data, a **trade-off** is made between **Bias** and **Variance**.

* *Summarise evaluation here*
  + **Time Series Forecasting**
* *Summarise evaluation here*
  + **Results and Conclusions**
* *Summarise results and conclusions here*

1. **Technologies Used**:

The project used the following technologies:

* Linear Regression Evaluation:
  + Lasso
  + Ridge
  + PlasticNet
* Polynomial Regression Evaluation:
  + Regression Analysis Code from **[2]** - Blog by Animesh Agarwal
* Regression Executio
  + Scikit-Learn
* Tableau
* PostGreSQL Data Base
* Python Pandas
* Python Flask Powered API
* Python Library - psycopg2
* Java Script D3.js
* HTML/CSS Bootstrap
* Heroku
* GitHub
* Other as required when we deploy

1. **Project Datasets**:

The datasets for the project can be found at the following link.

“JHU – Time Series Daily Reports”

<https://github.com/CSSEGISandData/COVID-19/tree/master/csse_covid_19_data/csse_covid_19_daily_reports>

1. **Database QuickDB Code**

*We need to revalidate the Quickdb Code*

The QuickDB code used to create the data base schema follows.

|  |  |
| --- | --- |
|  | {\colortbl;\red255\green255\blue255;} |
|  | {\\*\expandedcolortbl;;} |
|  | \paperw11900\paperh16840\margl1440\margr1440\vieww11520\viewh8400\viewkind0 |
|  | \pard\tx566\tx1133\tx1700\tx2267\tx2834\tx3401\tx3968\tx4535\tx5102\tx5669\tx6236\tx6803\pardirnatural\partightenfactor0 |
|  |  |
|  | \f0\fs24 \cf0 country\_codes\ |
|  | -\ |
|  | country\_id VARCHAR(255) PK\ |
|  | country\_name VARCHAR(255)\ |
|  | continent\_name VARCHAR(255)\ |
|  | \ |
|  | covid\_cases\ |
|  | -\ |
|  | country\_id VARCHAR(255) FK - country\_codes.country\_id\ |
|  | date VARCHAR(255)\ |
|  | confirmed INT\ |
|  | deaths INT\ |
|  | recovered INT\ |
|  | active INT\ |
|  | new\_cases INT\ |
|  | new\_deaths INT\ |
|  | new\_recovered INT\ |
|  | \ |
|  | population\ |
|  | -\ |
|  | country\_id VARCHAR(255) FK - country\_codes.country\_id\ |
|  | population INT\ |
|  | \ |
|  | vaccinations\ |
|  | -\ |
|  | country\_id VARCHAR(255) FK - country\_codes.country\_id\ |
|  | date VARCHAR(255)\ |
|  | fully\_vaccinated\_per\_hundred INT\ |
|  | not\_fully\_vaccinated\_per\_hundred INT\ |
|  | boosted\_per\_hundred INT} |

1. **Database Schema – Entity Relationship Diagram**

*We need to revalidate the ERD*

Text

Description automatically generated

1. **Database Description**

The key to the data base was to use the International Standards Organisation (iso\_code: ISO 3166-1 alpha-3 – three-letter country code) henceforth referred to as “iso-code”, to create relationships between the tables.

The “country-codes” table contains the “iso-code” and matching “country-name” for all countries covered by the “iso-code” and was generated during the Extraction phase of the project.

The “covid-cases” table contains the basic cleansed data from the JHU Data Sets which form the basis of the global view of Covid-19 cases.

The “vaccinations” table contains vaccination status from the Our World in data Vaccination data set.

1. **Database Meta Data**

**“country” table**

* country-id: the iso\_code: ISO 3166-1 alpha-3 – three-letter country code
* country-name: the name of the country in the ISO data set

**“covid-cases” table**

* country-id: the iso\_code: ISO 3166-1 alpha-3 – three-letter country code
* date: date of the observation
* confirmed: the total number of cumulative confirmed Covid-19 cases regardless of the variant
* deaths: the total number of cumulative deaths attributed to Covid-19 regardless of the variant
* recovered: the total number of cumulative recovered Covid-19 cases
* active: the total number of cumulative active Covid-19 cases
* new\_cases: the total number of incremental new Covid-19 cases
* new\_deaths: the total number of incremental new Covid-19 deaths
* new\_ recovered:

the total number of incremental new recovered Covid-19 cases

**“population” table**

* country-id: the iso\_code: ISO 3166-1 alpha-3 – three-letter country code
* population: the population of the country at 31/12/2020

**“vaccinations” table**

* country-id: the iso\_code: ISO 3166-1 alpha-3 – three-letter country code
* date: date of the observation
* vaccinated\_per\_hundred:

total number of people who received at least one vaccine dose. If a person receives the first dose of a 2-dose vaccine, this metric goes up by 1. If they receive the second dose, the metric stays the same i.e., 1.

* fully\_vaccinated\_per\_hundred:

people vaccinated per 100 people in the total population of the country. If a person receives the first dose of a 2-dose vaccine, this metric stays the same. If they receive the second dose, the metric goes up by 1.

* not\_fully\_vaccinated\_per\_hundred:

 people not vaccinated per 100 people in the total population of the country

* boosted\_per\_hundred:

people who have received their booster dose per 100 people in the total population of the country

1. **Data Transformation:**

The data Transformation steps are as follows:

1. Save DFs to CSVs to do exploratory data analysis.
2. Conduct exploratory data analysis.
3. Use melt() to unpivot DataFrames from current wide format 265 rows × 749 columns into long format 208600 rows × 6 columns.
4. Remove recovered data for Canada due to mismatch issue. Canada recovered data is counted for the whole Country instead of by Province/State which is how Canada and the rest of the world count data for "Confirmed Cases" and "Deaths".
5. Merge the three JHU dataframes, Confirmed Cases, Deaths, Recovered Cases.
   1. merge confirmed\_df\_long and deaths\_df\_long into full\_table
   2. merge full\_table and recovered\_df\_long
6. Check Canada data in "full\_table" - "recovered" should be 0 and check of CSV file confirms that it is.
7. Convert date from string to datetime.
8. Detect missing values NaN.
9. Replace 'recovered' NaNs with zero.
10. Three cruise ships need to be treated differently to the rest of the cases.So extract and remove data for these ships.
11. Calculate active cases = confirmed cases - deaths – recovered cases.
12. Aggregate data into Country/Region and group by Date and Country/Region.
13. Calculate daily New cases, New deaths and New recovered by deducting the corresponding accumulative data on the previous day
14. Use pd.merge to group the final data frame on Country/Region / Date.
15. Fix the new data types as integer.
16. The final data frame is sorted by Date and Country/Region ascending where: -

Confirmed Cases, Deaths, Recovered and Active are cumulative data for the entire period, and,

New cases, New deaths and New Recovered are daily incremental data.

1. Convert data frame to a csv file for backup.
2. Select Australia to check that data is correct. Validate the final data frame against the JHU Dashboard for 06/02/2022.

Both showed Confirmed Cases = 2,704,275 and Deaths = 4,154 for Australia.

1. Read the Vaccination dataset - csv file into a data frame.
2. Derive the “people\_not\_vaccinated” from the “people\_fully\_vaccinated”.
3. Detect missing values NaN
4. Replace NaNs with zero
5. Data cleansing replace ”United States” with “US” to standardise data.
6. Save cleansed vaccination data to a CSV for backup.
7. Read the Population data set - csv file into a data frame.
8. Detect missing values NaN
9. Replace NaNs with zero
10. Save cleansed Population data to a CSV for backup.
11. Copy OWID Vaccination data frame, as we want to use OWID country codes.
12. Add Africas to match population data frame.
13. Edit “full\_grouped” covid case data frame to include country ID.
14. Change structure of data frames to match structure of tables created in the database.
15. Set index of country codes data frame and remove null index row.
16. Covid Cases table - copy only the columns needed into a new Data Frame.
17. Rename columns to fit the tables created in the database.
18. Vaccinations table - copy only the columns needed into a new Data Frame.
19. Rename columns to fit the tables created in the database.
20. Create PostgreSQL database connection.
21. Confirm database tables.
22. Load data frames to the database tables