Configuration Manual

MSc Research Project
Programme Name

Tommy Ryan
Student ID: x22152458

School of Computing
National College of Ireland

Supervisor: Vladimir Milosavljevic

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Tommy Ryan
Student ID:	x22152458
Programme:	Programme Name
Year:	2024
Module:	MSc Research Project
Supervisor:	Vladimir Milosavljevic
Submission Due Date:	12/08/2024
Project Title:	Configuration Manual
Word Count:	581
Page Count:	10

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Tommy Ryan
Date:	11th August 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☒
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Tommy Ryan
x22152458

1 Introduction

The aim of this project is to analyse historical Covid-19 data and create predictive models to see how different countries strategies of dealing with Covid would work if these strategies were in place in Ireland. Three predictive models were created in Phyton, linear regression, decision trees and random forest. An UI element was also created using flask in python, code snippets will be used in the following sections to show what was done.

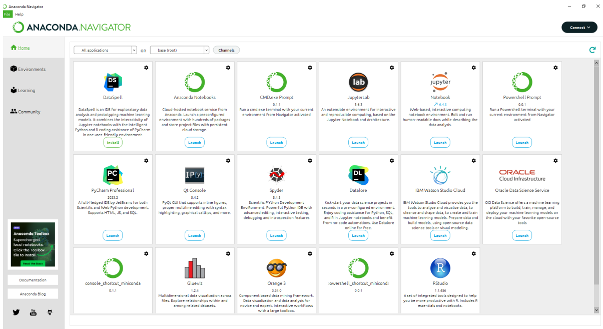
2 System Requirements

2.1 2.1 Personal computer

Processor: AMD Ryzen 5 1600 Six-Core Processor, 3200 Mhz, 6 Core(s), 12 Logical Processor(s)
Installed Physical Memory (RAM) 8.00 GB
Minimum storage space 10gb
OS: OS Name Microsoft Windows 10 Pro
GPU: NVIDIA GeForce GTX 1060 3GB

2.2 2.2 Software Configuration

Anaconda installation https://docs.anaconda.com/anaconda/install/windows/



DataSpell
DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.

Anaconda Notebooks
Cloud-hosted notebook service from Anaconda. Launch a preconfigured environment with hundreds of packages and store project files with persistent cloud storage.

CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated

JupyterLab
3.6.3
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
6.4.8
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated

PyCharm Professional
2023.2
A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.

Qt Console
5.4.2
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
5.4.3
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Datalore
Kick-start your data science projects in seconds in a pre-configured environment. Enjoy coding assistance for Python, SQL, and R in Jupyter notebooks and benefit from no-code automations. Use Datalore online for free.

IBM Watson Studio Cloud
IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

ORACLE Cloud Infrastructure
Oracle Data Science Service
OCI Data Science offers a machine learning platform to build, train, manage, and deploy your machine learning models on the cloud with your favorite open-source tools

console_shortcut_miniconda
0.1.1

Glueviz
1.2.4
Multidimensional data visualization across files. Explore relationships within and among related datasets.

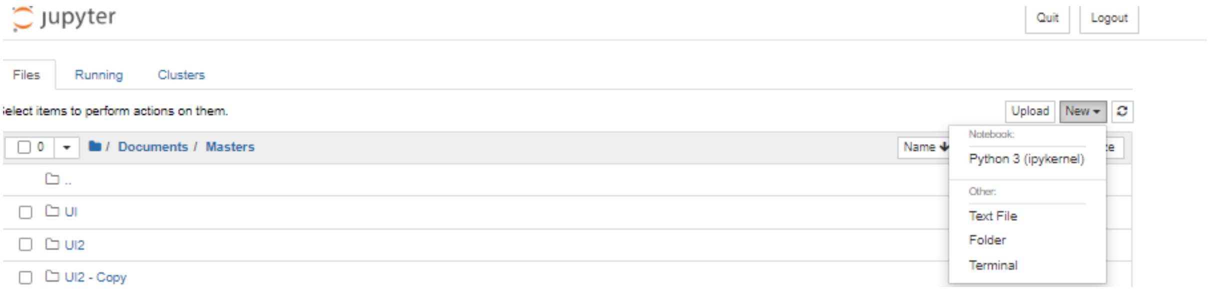
Orange 3
3.34.0
Component based data mining Framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

owershell_shortcut_miniconda
0.0.1

RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Jupyter Notebook

Launch Jupyter Notebook create a folder for your new project and create a new Python 3 project.


ielect items to perform actions on them.

UI Flask Environment
1. Setting Up Python Environment Install Python and Pip
2. Creating a virtual environment: with the command "python -m venv venv"
3. Activate the virtual environment: `venv\Scripts\activate`
4. Installing libraries: The commands "pip install flask pandas numpy scikit-learn matplotlib statsmodels"
5. Create a directory mkdir covid-project
6. Create 2 separate folders for the html and js script the mkdir html script

3 Packages

The following libraries are installed and imported.

Pandas
Numpy
CV2
Matplotlib
scikit-learn
flask
RandomForestRegressor
DecisionTreeRegressor

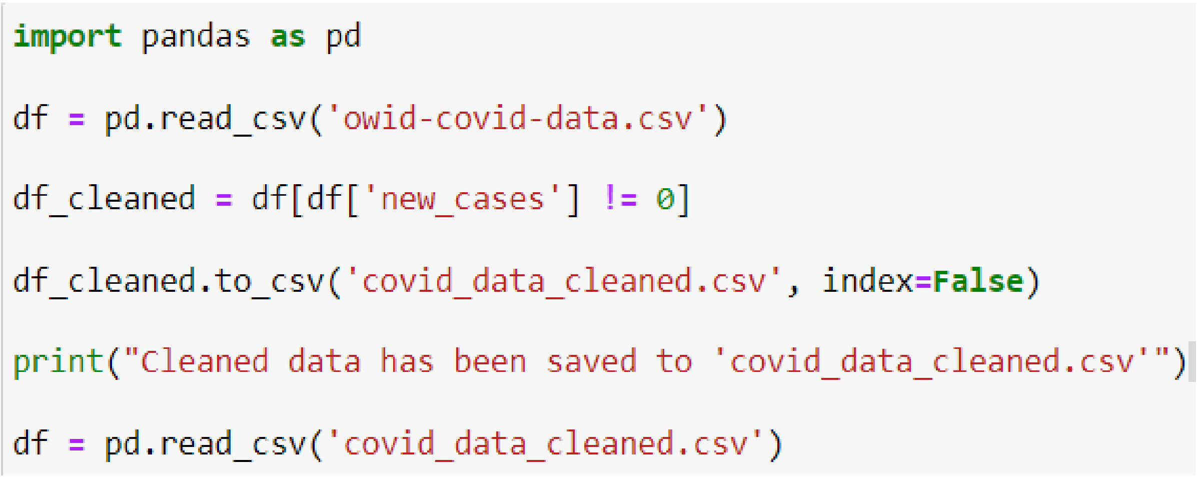
4 Implementation

4.1 Data Collection

The dataset was downloaded from the ourworldindata.org website particularly from the link https://covid.ourworldindata.org/data/owid-covid-data.csv where all the data was put into one csv.

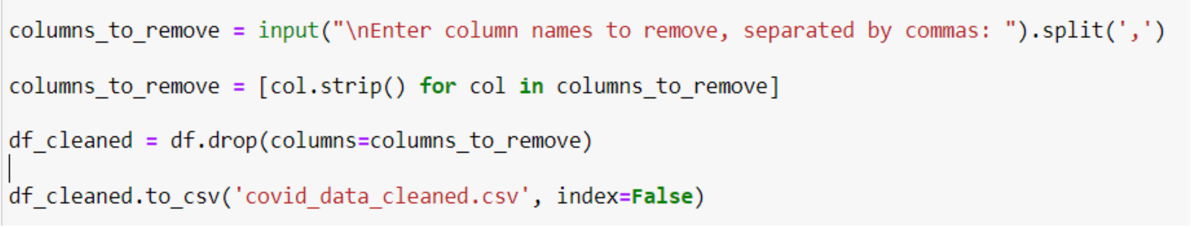
4.2 Data Preprocessing

New cases were only calculated at the end of every week so the data was cleaned to remove any rows that had no new covid cases. Then this cleaned version was saved



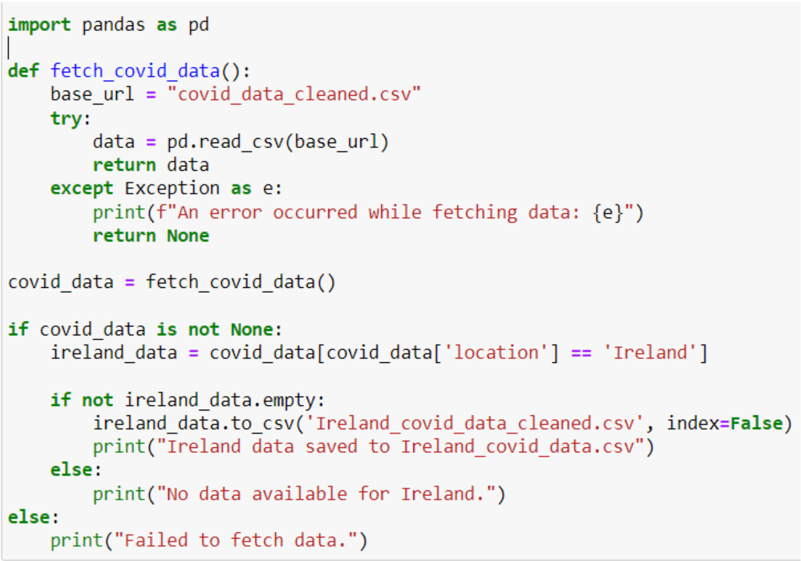
```
import pandas as pd

df = pd.read_csv('owid-covid-data.csv')

df_cleaned = df[df['new_cases'] != 0]

df_cleaned.to_csv('covid_data_cleaned.csv', index=False)

print("Cleaned data has been saved to 'covid_data_cleaned.csv'")

df = pd.read_csv('covid_data_cleaned.csv')
```

Next any irrelevant columns were removed like tests_units as this was not numeric data.



```
columns_to_remove = input("\nEnter column names to remove, separated by commas: ").split(',')

columns_to_remove = [col.strip() for col in columns_to_remove]

df_cleaned = df.drop(columns=columns_to_remove)

df_cleaned.to_csv('covid_data_cleaned.csv', index=False)
```

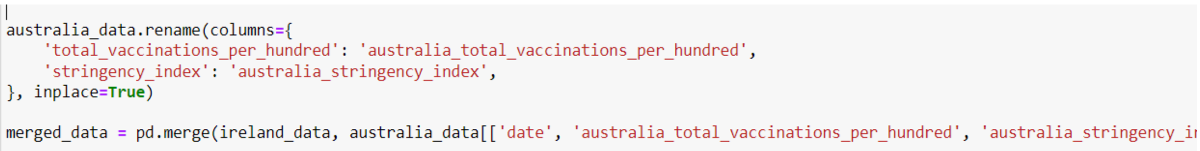
Next a csv file of the country of choice was created from the covid data to only include that countries covid data. This was done for Ireland, Australia, Israel and Sweden.



```
import pandas as pd

def fetch_covid_data():
    base_url = "covid_data_cleaned.csv"
    try:
        data = pd.read_csv(base_url)
        return data
    except Exception as e:
        print(f"An error occurred while fetching data: {e}")
        return None

covid_data = fetch_covid_data()

if covid_data is not None:
    ireland_data = covid_data[covid_data['location'] == 'Ireland']

    if not ireland_data.empty:
        ireland_data.to_csv('Ireland_covid_data_cleaned.csv', index=False)
        print("Ireland data saved to Ireland_covid_data.csv")
    else:
        print("No data available for Ireland.")
else:
    print("Failed to fetch data.")
```

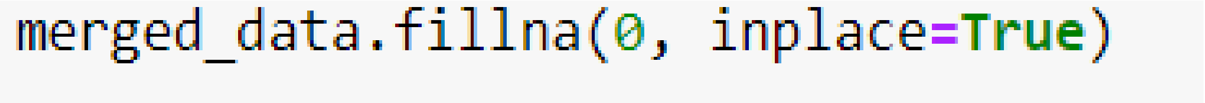
4.3 Data Transformation

Any predictors we wanted to use from the other coutries were renamed and merged into the Irelands data set.



```
australia_data.rename(columns={
    'total_vaccinations_per_hundred': 'australia_total_vaccinations_per_hundred',
    'stringency_index': 'australia_stringency_index',
}, inplace=True)

merged_data = pd.merge(ireland_data, australia_data[['date', 'australia_total_vaccinations_per_hundred', 'australia_stringency_i
```

Any columns which were NAN were set to zero.

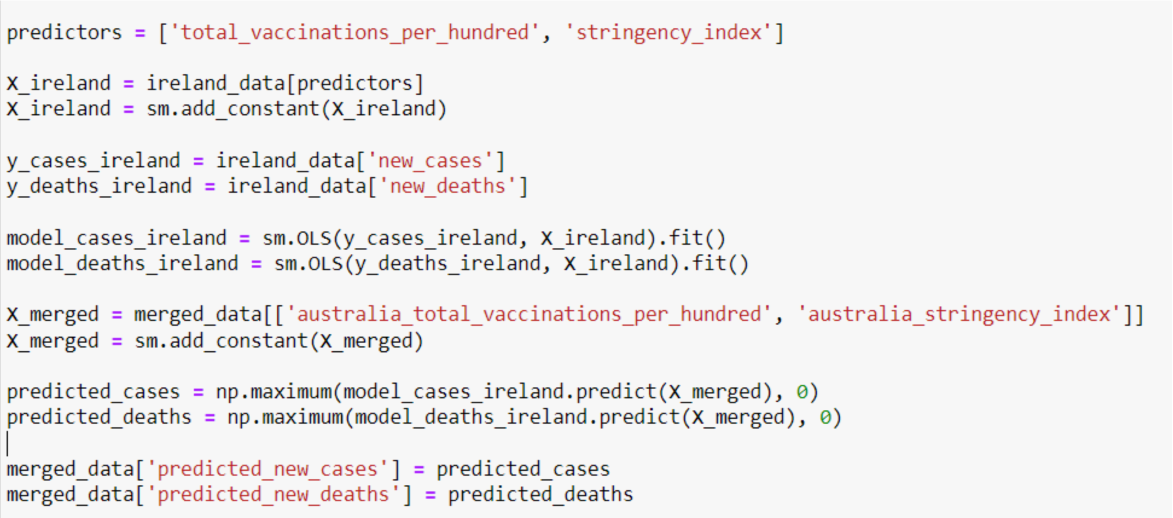


```
merged_data.fillna(0, inplace=True)
```

4.4 Model Building

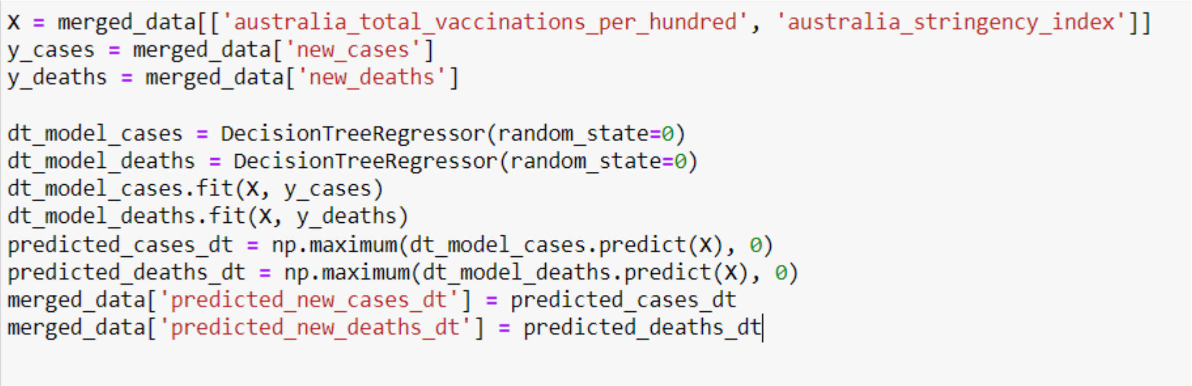
Three models were used to predict the cases and deaths, linear regression, decision trees and random forests.

Linear regression



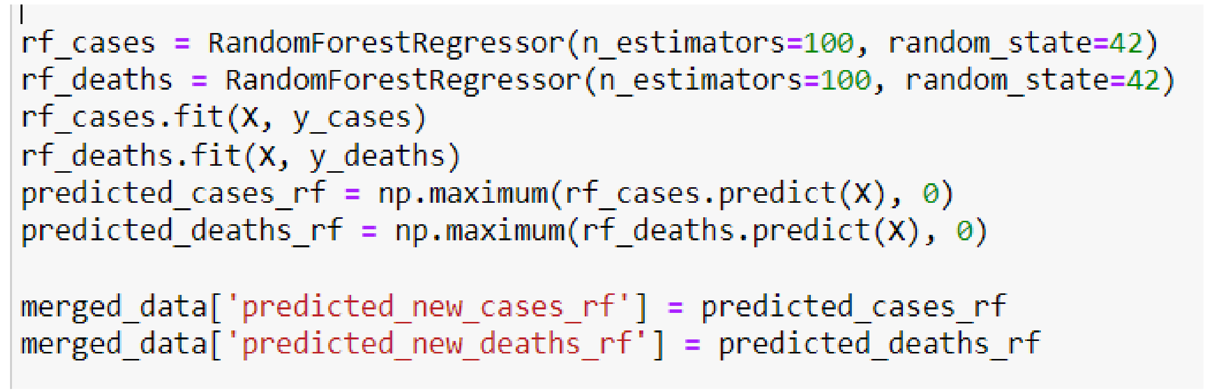
```
predictors = ['total_vaccinations_per_hundred', 'stringency_index']

X_ireland = ireland_data[predictors]
X_ireland = sm.add_constant(X_ireland)

y_cases_ireland = ireland_data['new_cases']
y_deaths_ireland = ireland_data['new_deaths']

model_cases_ireland = sm.OLS(y_cases_ireland, X_ireland).fit()
model_deaths_ireland = sm.OLS(y_deaths_ireland, X_ireland).fit()

X_merged = merged_data[['australia_total_vaccinations_per_hundred', 'australia_stringency_index']]
X_merged = sm.add_constant(X_merged)

predicted_cases = np.maximum(model_cases_ireland.predict(X_merged), 0)
predicted_deaths = np.maximum(model_deaths_ireland.predict(X_merged), 0)

merged_data['predicted_new_cases'] = predicted_cases
merged_data['predicted_new_deaths'] = predicted_deaths
```

Decision Trees



```
X = merged_data[['australia_total_vaccinations_per_hundred', 'australia_stringency_index']]
y_cases = merged_data['new_cases']
y_deaths = merged_data['new_deaths']

dt_model_cases = DecisionTreeRegressor(random_state=0)
dt_model_deaths = DecisionTreeRegressor(random_state=0)
dt_model_cases.fit(X, y_cases)
dt_model_deaths.fit(X, y_deaths)
predicted_cases_dt = np.maximum(dt_model_cases.predict(X), 0)
predicted_deaths_dt = np.maximum(dt_model_deaths.predict(X), 0)
merged_data['predicted_new_cases_dt'] = predicted_cases_dt
merged_data['predicted_new_deaths_dt'] = predicted_deaths_dt
```

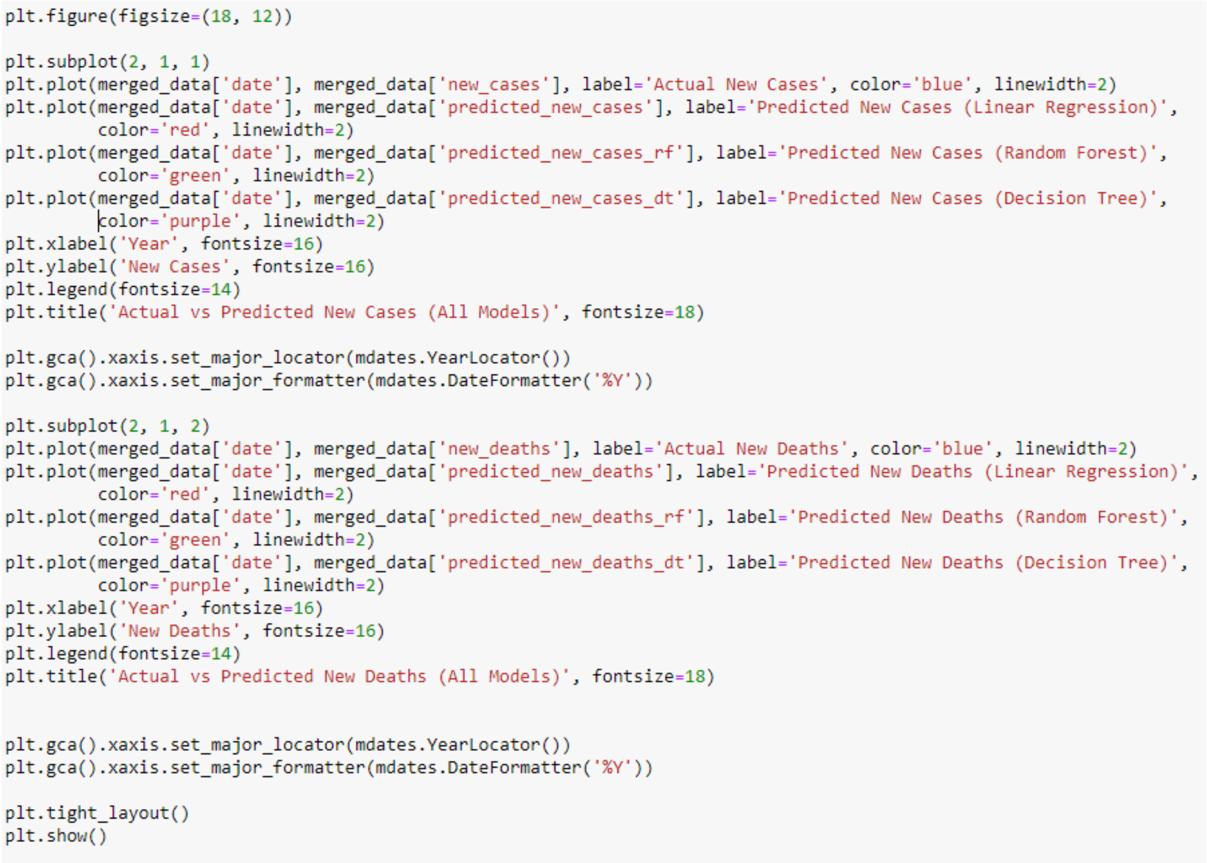
Random forest



```
rf_cases = RandomForestRegressor(n_estimators=100, random_state=42)
rf_deaths = RandomForestRegressor(n_estimators=100, random_state=42)
rf_cases.fit(X, y_cases)
rf_deaths.fit(X, y_deaths)
predicted_cases_rf = np.maximum(rf_cases.predict(X), 0)
predicted_deaths_rf = np.maximum(rf_deaths.predict(X), 0)

merged_data['predicted_new_cases_rf'] = predicted_cases_rf
merged_data['predicted_new_deaths_rf'] = predicted_deaths_rf
```

4.4.1 Plotting results

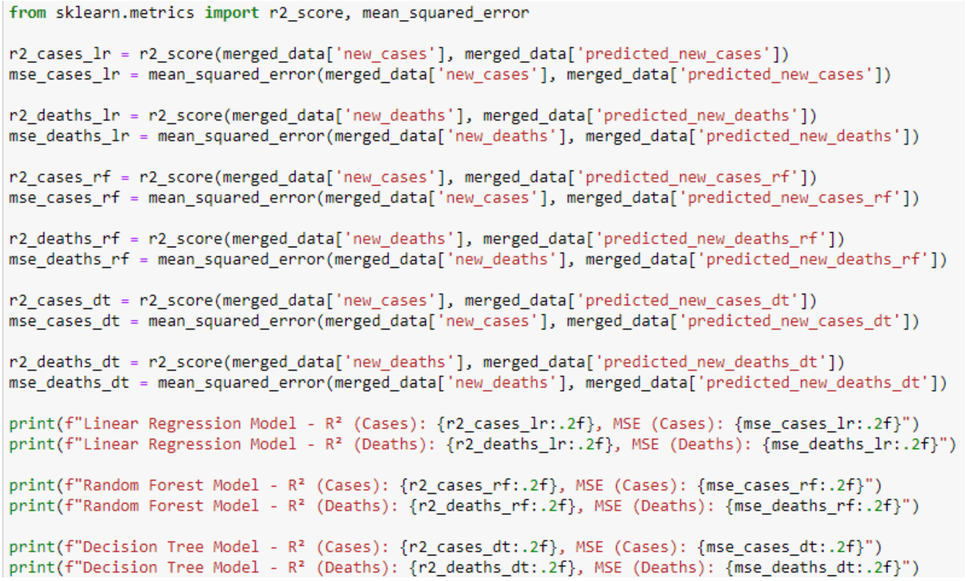
Next two graphs were plotted one for the new cases and one for the new deaths. There will be four lines on the graph, one for the actual data, one for the linear regression prediction, one for the decision tress prediction and one for the random forest prediction.



```
plt.figure(figsize=(18, 12))

plt.subplot(2, 1, 1)
plt.plot(merged_data['date'], merged_data['new_cases'], label='Actual New Cases', color='blue', linewidth=2)
plt.plot(merged_data['date'], merged_data['predicted_new_cases'], label='Predicted New Cases (Linear Regression)',
         color='red', linewidth=2)
plt.plot(merged_data['date'], merged_data['predicted_new_cases_rf'], label='Predicted New Cases (Random Forest)',
         color='green', linewidth=2)
plt.plot(merged_data['date'], merged_data['predicted_new_cases_dt'], label='Predicted New Cases (Decision Tree)',
         color='purple', linewidth=2)
plt.xlabel('Year', fontsize=16)
plt.ylabel('New Cases', fontsize=16)
plt.legend(fontsize=14)
plt.title('Actual vs Predicted New Cases (All Models)', fontsize=18)

plt.gca().xaxis.set_major_locator(mdates.YearLocator())
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))

plt.subplot(2, 1, 2)
plt.plot(merged_data['date'], merged_data['new_deaths'], label='Actual New Deaths', color='blue', linewidth=2)
plt.plot(merged_data['date'], merged_data['predicted_new_deaths'], label='Predicted New Deaths (Linear Regression)',
         color='red', linewidth=2)
plt.plot(merged_data['date'], merged_data['predicted_new_deaths_rf'], label='Predicted New Deaths (Random Forest)',
         color='green', linewidth=2)
plt.plot(merged_data['date'], merged_data['predicted_new_deaths_dt'], label='Predicted New Deaths (Decision Tree)',
         color='purple', linewidth=2)
plt.xlabel('Year', fontsize=16)
plt.ylabel('New Deaths', fontsize=16)
plt.legend(fontsize=14)
plt.title('Actual vs Predicted New Deaths (All Models)', fontsize=18)


plt.gca().xaxis.set_major_locator(mdates.YearLocator())
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))

plt.tight_layout()
plt.show()
```

5 Evaluation

These models were evaluated using R² and Mean Squared Error.

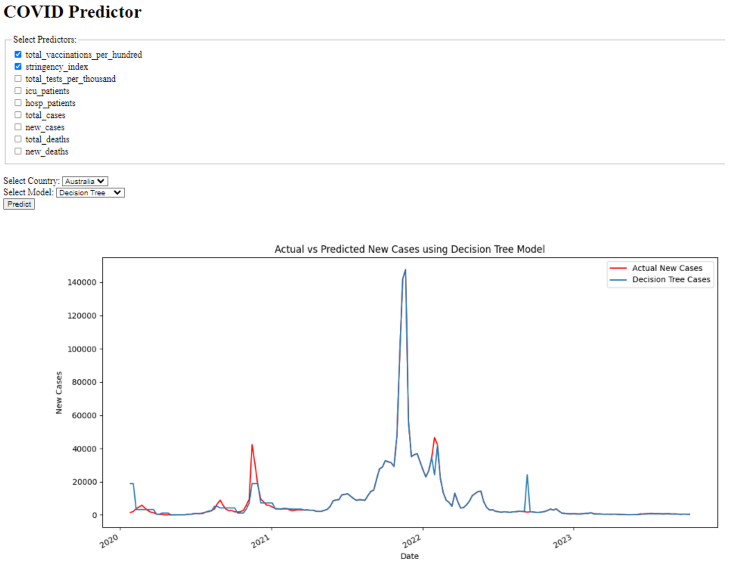


```
from sklearn.metrics import r2_score, mean_squared_error

r2_cases_lr = r2_score(merged_data['new_cases'], merged_data['predicted_new_cases'])
mse_cases_lr = mean_squared_error(merged_data['new_cases'], merged_data['predicted_new_cases'])

r2_deaths_lr = r2_score(merged_data['new_deaths'], merged_data['predicted_new_deaths'])
mse_deaths_lr = mean_squared_error(merged_data['new_deaths'], merged_data['predicted_new_deaths'])

r2_cases_rf = r2_score(merged_data['new_cases'], merged_data['predicted_new_cases_rf'])
mse_cases_rf = mean_squared_error(merged_data['new_cases'], merged_data['predicted_new_cases_rf'])

r2_deaths_rf = r2_score(merged_data['new_deaths'], merged_data['predicted_new_deaths_rf'])
mse_deaths_rf = mean_squared_error(merged_data['new_deaths'], merged_data['predicted_new_deaths_rf'])

r2_cases_dt = r2_score(merged_data['new_cases'], merged_data['predicted_new_cases_dt'])
mse_cases_dt = mean_squared_error(merged_data['new_cases'], merged_data['predicted_new_cases_dt'])

r2_deaths_dt = r2_score(merged_data['new_deaths'], merged_data['predicted_new_deaths_dt'])
mse_deaths_dt = mean_squared_error(merged_data['new_deaths'], merged_data['predicted_new_deaths_dt'])

print(f"Linear Regression Model - R² (Cases): {r2_cases_lr:.2f}, MSE (Cases): {mse_cases_lr:.2f}")
print(f"Linear Regression Model - R² (Deaths): {r2_deaths_lr:.2f}, MSE (Deaths): {mse_deaths_lr:.2f}")

print(f"Random Forest Model - R² (Cases): {r2_cases_rf:.2f}, MSE (Cases): {mse_cases_rf:.2f}")
print(f"Random Forest Model - R² (Deaths): {r2_deaths_rf:.2f}, MSE (Deaths): {mse_deaths_rf:.2f}")

print(f"Decision Tree Model - R² (Cases): {r2_cases_dt:.2f}, MSE (Cases): {mse_cases_dt:.2f}")
print(f"Decision Tree Model - R² (Deaths): {r2_deaths_dt:.2f}, MSE (Deaths): {mse_deaths_dt:.2f}")
```

6 UI Creation

After the UI Flask Environment was set up 3 files were created, a python file located in UI\covid-project, a JavaScript file located in UI\covid-project\script and HTML file located in UI\covid-project\HTML

The Python file has the regression models and data gathering as shown before. The HTML has code to be able to choose a country and its predictors to compare with Ireland and choose which model to plot. Then the script file relays which country you choose in the UI back to the python code.



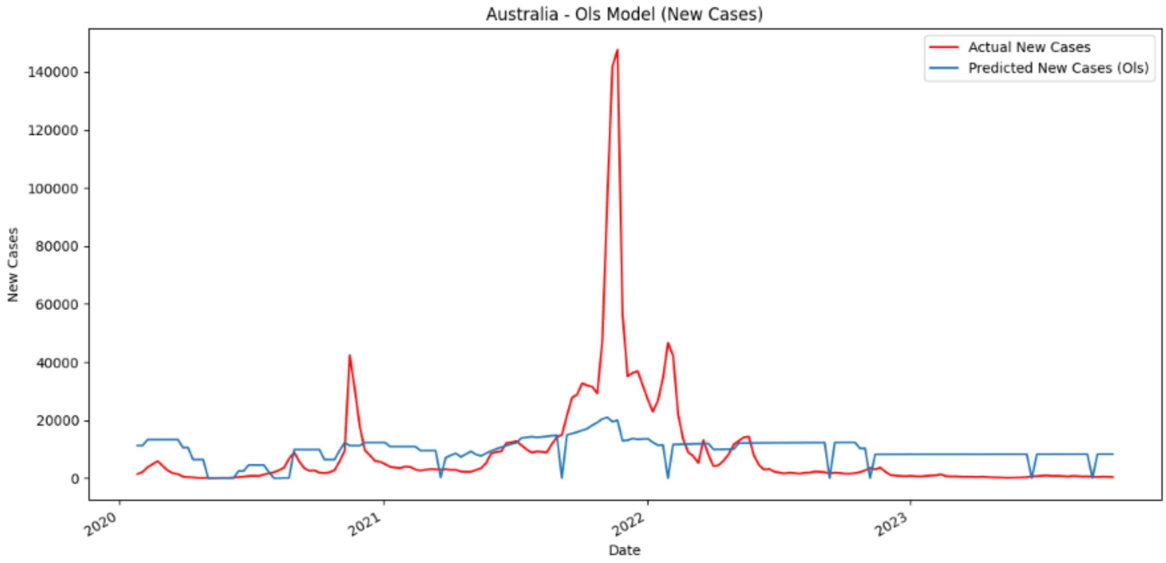
COVID Predictor

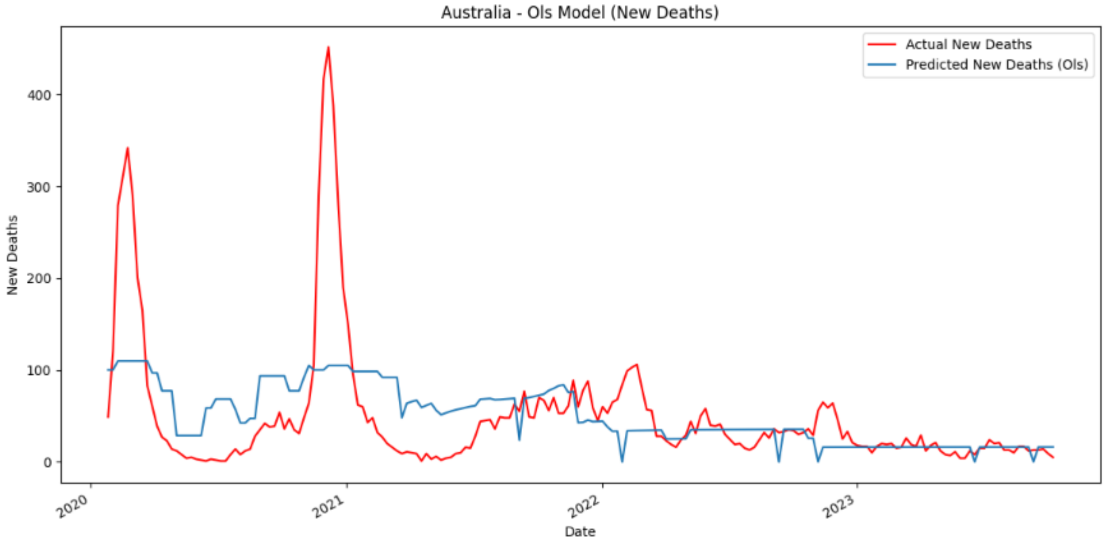
Select Predictors:

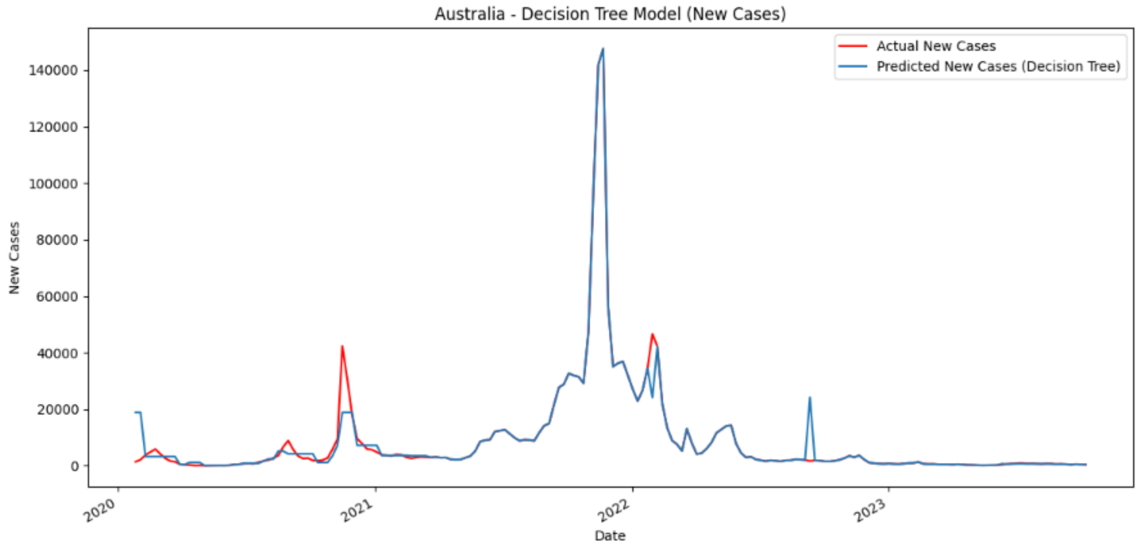
☑ total_vaccinations_per_hundred
☑ stringency_index
□ total_tests_per_thousand
□ icu_patients
□ hosp_patients
□ total_cases
□ new_cases
□ total_deaths
□ new_deaths

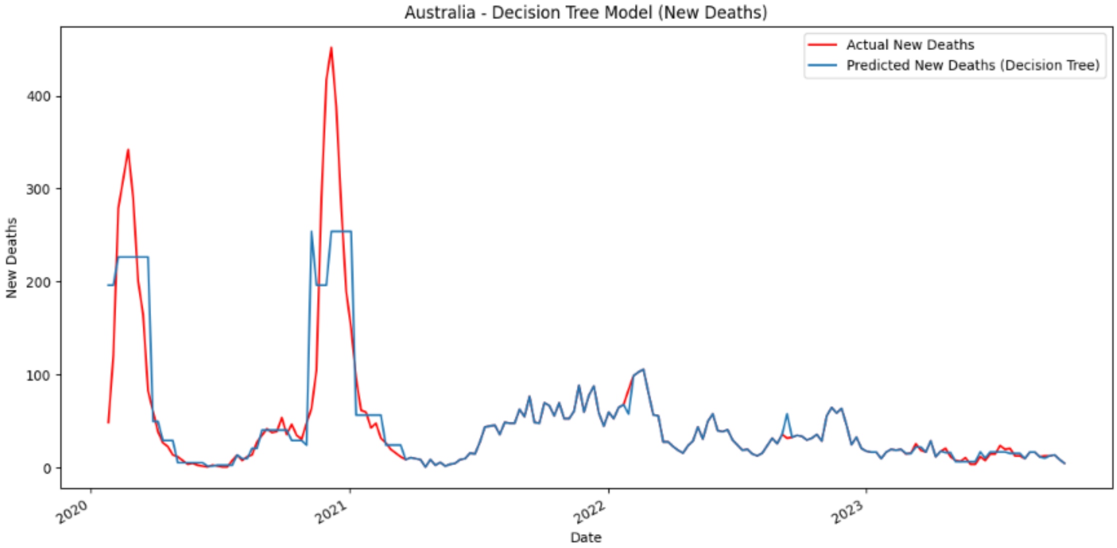
Select Country: Australia
Select Model: Decision Tree

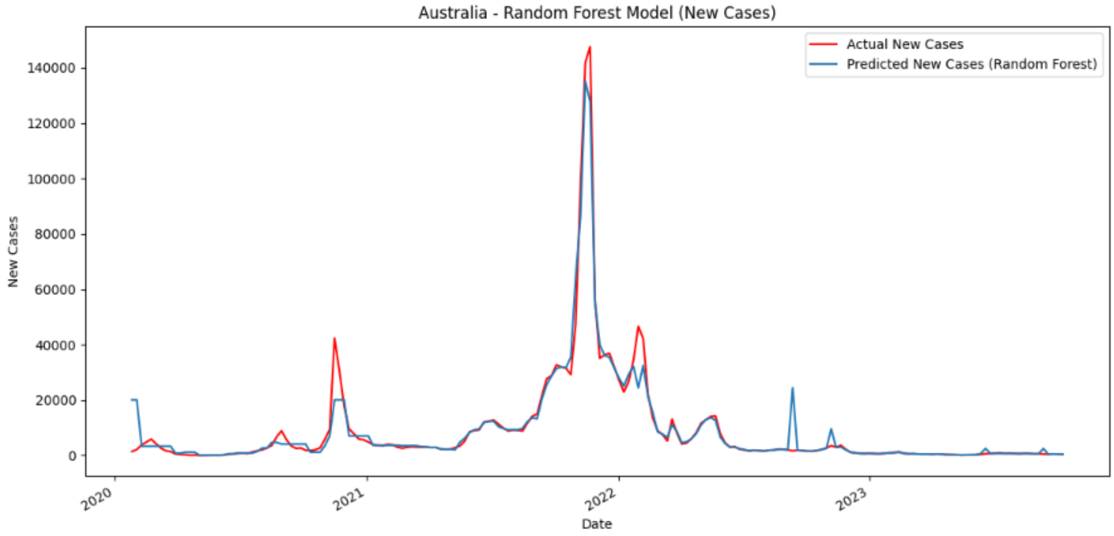
7 All Graphs

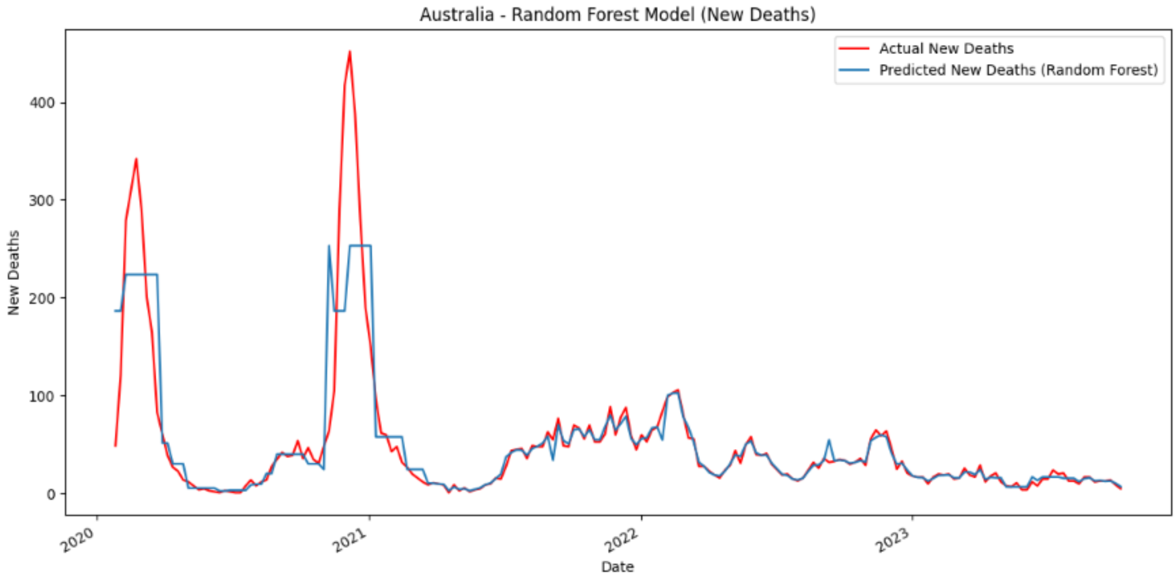

Australia - Ols Model (New Cases)
Actual New Cases
Predicted New Cases (Ols)
New Cases
140000
120000
100000
80000
60000
40000
20000
0
2020
2021
2022
2023
Date

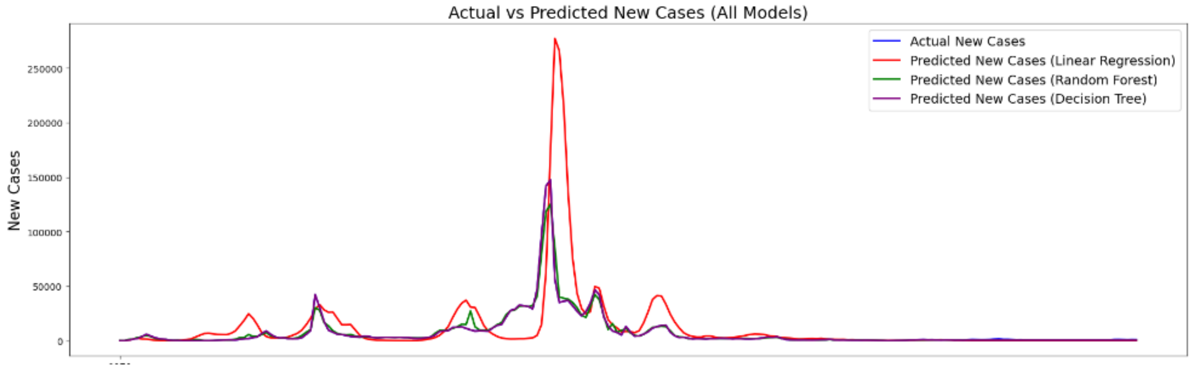

Australia - Ols Model (New Deaths)
Actual New Deaths
Predicted New Deaths (Ols)
New Deaths
400
300
200
100
0
2020
2021
2022
2023
Date

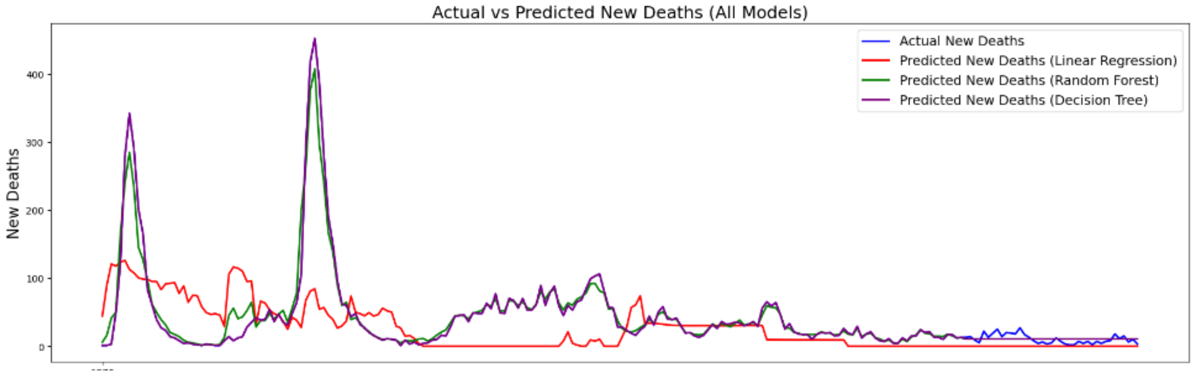

Australia - Decision Tree Model (New Cases)
Actual New Cases
Predicted New Cases (Decision Tree)
New Cases
0
20000
40000
60000
80000
100000
120000
140000
2020
2021
2022
2023
Date

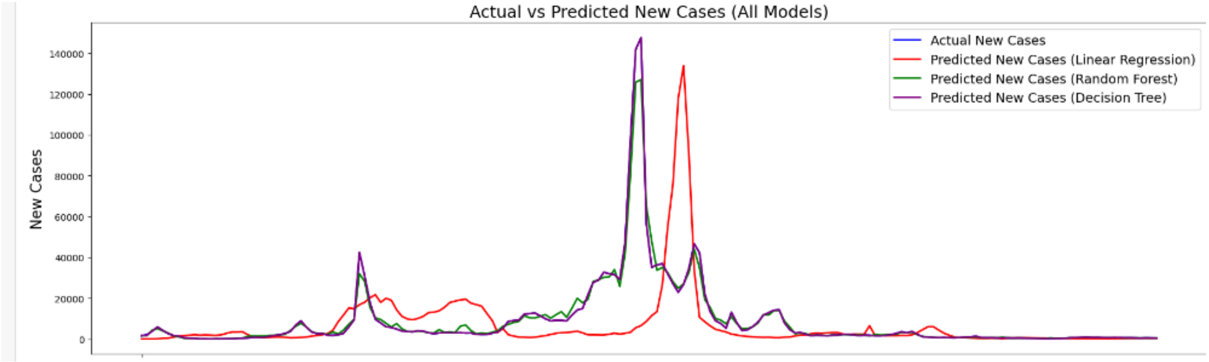

Australia - Decision Tree Model (New Deaths)
Actual New Deaths
Predicted New Deaths (Decision Tree)
New Deaths
0
100
200
300
400
2020
2021
2022
2023
Date

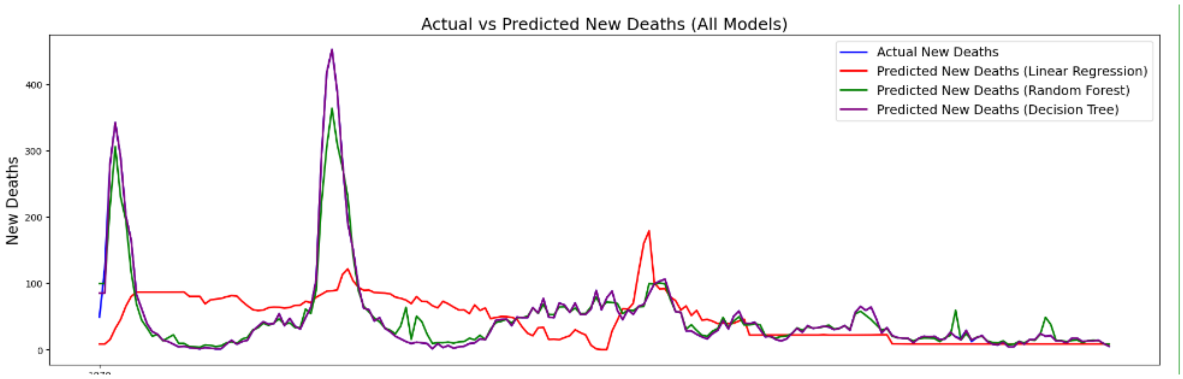

Australia - Random Forest Model (New Cases)
Actual New Cases
Predicted New Cases (Random Forest)
New Cases
0
20000
40000
60000
80000
100000
120000
140000
2020
2021
2022
2023
Date

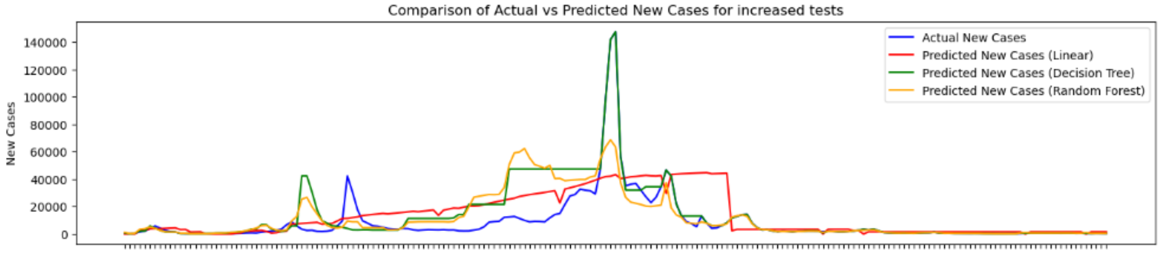

Australia - Random Forest Model (New Deaths)
Actual New Deaths
Predicted New Deaths (Random Forest)
New Deaths
400
300
200
100
0
2020
2021
2022
2023
Date

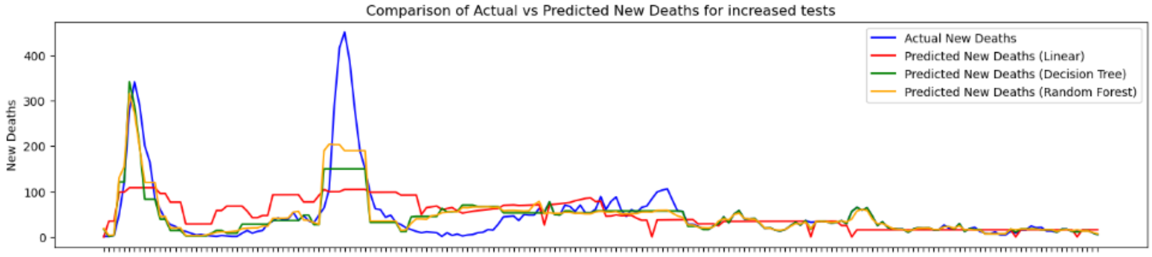

Actual vs Predicted New Cases (All Models)
Actual New Cases
Predicted New Cases (Linear Regression)
Predicted New Cases (Random Forest)
Predicted New Cases (Decision Tree)
New Cases
250000
200000
150000
100000
50000
0

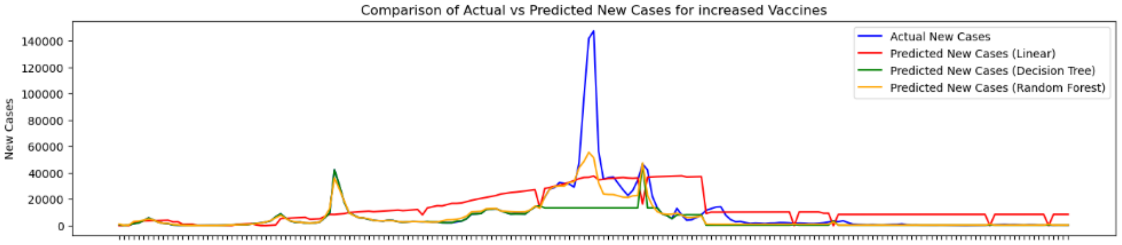

Actual vs Predicted New Deaths (All Models)
Actual New Deaths
Predicted New Deaths (Linear Regression)
Predicted New Deaths (Random Forest)
Predicted New Deaths (Decision Tree)
New Deaths
400
300
200
100
0

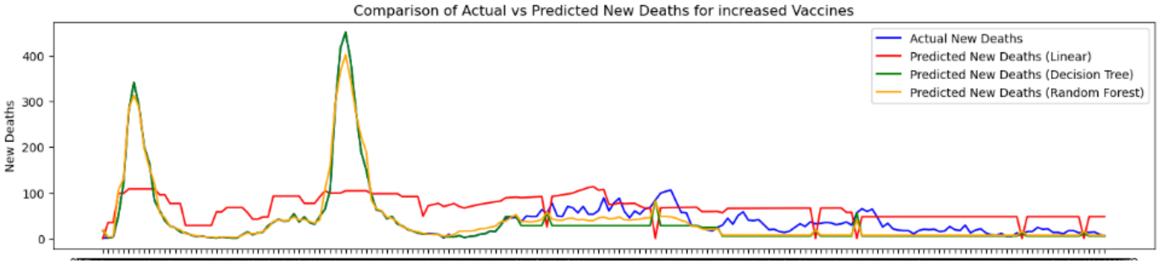

Actual vs Predicted New Cases (All Models)
Actual New Cases
Predicted New Cases (Linear Regression)
Predicted New Cases (Random Forest)
Predicted New Cases (Decision Tree)
New Cases
140000
120000
100000
80000
60000
40000
20000
0


Actual vs Predicted New Deaths (All Models)
New Deaths
0
100
200
300
400
Actual New Deaths
Predicted New Deaths (Linear Regression)
Predicted New Deaths (Random Forest)
Predicted New Deaths (Decision Tree)


Comparison of Actual vs Predicted New Cases for increased tests
New Cases
0
20000
40000
60000
80000
100000
120000
140000
Actual New Cases
Predicted New Cases (Linear)
Predicted New Cases (Decision Tree)
Predicted New Cases (Random Forest)


Comparison of Actual vs Predicted New Deaths for increased tests
New Deaths
0
100
200
300
400
Actual New Deaths
Predicted New Deaths (Linear)
Predicted New Deaths (Decision Tree)
Predicted New Deaths (Random Forest)


Comparison of Actual vs Predicted New Cases for increased Vaccines
New Cases
0
20000
40000
60000
80000
100000
120000
140000
Actual New Cases
Predicted New Cases (Linear)
Predicted New Cases (Decision Tree)
Predicted New Cases (Random Forest)


Comparison of Actual vs Predicted New Deaths for increased Vaccines
New Deaths
0
100
200
300
400
Actual New Deaths
Predicted New Deaths (Linear)
Predicted New Deaths (Decision Tree)
Predicted New Deaths (Random Forest)