National College of Ireland

Configuration Manual

MSc Research Project
Data Analytics

Sonal Srinath
Student ID: 19207638

School of Computing
National College of Ireland

Supervisor: Dr Martin Alain

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Sonal Srinath		
Student ID:	19207638		
Programme:	MSc Data Analytics	**Year:**	2021
Module:	MSc Research Project		
Lecturer:	Dr Martin Alain		
Submission Due Date:	16/12/2021		
Project Title:	Forecasting of Hospital Outpatient Waiting Lists in Ireland using Time-Series and Machine Learning		
Word Count:	1227	**Page Count:**	12

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: Sonal Srinath

Date: 16/12/2021

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

Sonal Srinath
Student ID:19207638

1 Introduction

The main purpose of a configuration manual is to provide a detailed outline of how the research project was conducted in a succinct and organized manner such that the project might be replicated if/when required.

Project Aim: The aim of this research was to perform a thorough forecasting analysis on the hospital outpatient waiting lists in Ireland using three supervised machine learning algorithms – Artificial Neural Network, Random Forest, Linear Regression, and Time Series analysis using a seasonal ARIMA model for four time bands 0-3, 3-6, 6-9 and 9-12 months waiting period. It is necessary to carefully mitigate the issue of managing the long outpatient waiting lists by performing accurate modelling and forecasting (Harrou *et al.*, 2020),

2 System Specifications

a) Hardware Requirements

- **Operating System:** Windows 10 Home
- **Processor:** 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 1.38 GHz
- **Installed RAM:** 16GB
- **System Type:** 64-bit operating system, x64-based processor

b) Software Requirements

- **Programming languages**: Python 3.9.5 and R.
- **Integrated Development Environment (IDE):** Jupyter Notebook (v. 6.4.0) and RStudio (version 1.4.1103)
- **Additional Tools:** Microsoft Excel, Overleaf, PowerPoint.
- **Web Browser:** Google Chrome

3 Pre-Requisites

1) The first main step is to Install Python, Jupyter Notebook, and RStudio. The links are given below.

To install Python 3.9.5 or higher on Windows: https://docs.python.org/3/using/windows.html

To install Jupyter Notebook: https://test-jupyter.readthedocs.io/en/latest/install.html

To install RStudio: https://www.rstudio.com/products/rstudio/download/

2) The second step is to import all the required packages in both Jupyter Notebook and RStudio.

To install packages in Python:
https://packaging.python.org/en/latest/tutorials/installing-packages/

To install libraries in RStudio:
https://support.rstudio.com/hc/en-us/articles/201057987-Quick-list-of-useful-R-packages

```
In [1]: import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        from keras.models import Sequential
        from keras.layers import Dense
        from sklearn.metrics import accuracy_score
        from sklearn.metrics import mean_absolute_error
        from sklearn.metrics import mean_squared_error
        from sklearn.metrics import r2_score
        from sklearn.metrics import explained_variance_score
        import math
        from sklearn.ensemble import RandomForestRegressor
        from sklearn.linear_model import LinearRegression
```

```
#Importing the required packages
library(haven)
library(fpp2)
library(tseries)
library(forecast)
library(ggplot2)
library(dplyr)
```

4 Data Selection

The data was obtained from The National Treatment Purchase Fund (NTPF). The NTPF is an independent statutory body established by the Minister for Health. There is data missing for a few months in 2021 because of the cyber attack.

Dataset Link: https://www.ntpf.ie/home/outpatient_group.htm

CSV Files

OP Waiting List By Specialty 2021 (October To Present)

OP Waiting List By Hospital 2021 (October To Present)

OP Waiting List By Group Hospital 2021 (January To March)

OP Waiting List By Group Hospital 2020

OP Waiting List By Group Hospital 2019

OP Waiting List By Group Hospital 2018

OP Waiting List By Group Hospital 2017

OP Waiting List By Group Hospital 2016

OP Waiting List By Group Hospital 2015

OP Waiting List By Group Hospital 2014

The code below displays the first 5 rows of the dataset to have a glimpse of the data.

```
In [5]: #Printing the first 5 rows of the dataset
        df.head()
```

Out[5]:

	Archive Date	Group	Hospital HIPE	Hospital	Specialty HIPE	Specialty	Adult/Child	Age Categorisation	Time Bands	Count
0	2014-01-31	Children's Hospital Group	940	Childrens University Hospital Temple Street	NaN	Other	Child	0-15	0-3 Months	4
1	2014-01-31	Children's Hospital Group	940	Childrens University Hospital Temple Street	NaN	Other	Child	0-15	3-6 Months	1
2	2014-01-31	Children's Hospital Group	940	Childrens University Hospital Temple Street	NaN	Other	Child	0-15	6-9 Months	1
3	2014-01-31	Children's Hospital Group	940	Childrens University Hospital Temple Street	NaN	Other	Child	16-64	0-3 Months	1
4	2014-01-31	Children's Hospital Group	940	Childrens University Hospital Temple Street	100.0	Cardiology	Child	0-15	0-3 Months	193

The description of each column variable of the data is given below:

Data	Description
Date	The date of the data recorded
Group	The hospitals in Ireland are organized into seven hospital groups. This displays in which group the hospital belongs to.
Hospital HIPE	This is the Hospital Inpatient Enquiry number which denoted the principal source of data information from the hospitals
Hospital	This denotes which hospital in Ireland the appointment is taken for.
Specialty HIPE	This number denoted the HIPE for a specific specialty.
Specialty	This is the defined specialty of the consultant.
Adult/Child	This is a binary variable that indicates if the patient is an adult or a child.
Age Profile	This categorizes the age of the patient.
Time Bands	This column depicts the various time frames the patients have to wait for the treatment plan, for example, 3-6 months.
Count	This is the total number of patients on the waiting list.

The data is presented in 8 separate CSV files which have to be merged for each year from 2014 to 2021. The code below merges the data into one dataframe df.

Data Reading

```
In [2]: #Loading the dataset and merging the files from 2014-2021

        df = pd.read_csv('OP Waiting List By Group Hospital 2014.csv')
        for i in range(2015,2022):
            temp = pd.read_csv(f'OP Waiting List By Group Hospital {i}.csv')
            temp.columns = df.columns
            df = pd.concat([df,temp])
```

5 Data Pre-processing

For the pre-processing of the data, various functions were performed. It is important to do so to obtain optimal performance. df.describe() gives a few statistical details of the dataset. The dataframe now consists of 582729 rows and 10 columns shown in the code below.

```
In [7]: #Printing some statistical details of my dataset
        df.describe()
```

Out[7]:

	Hospital HIPE	Specialty HIPE	Count
count	582729.000000	581507.000000	582729.000000
mean	673.954540	2451.189733	71.365340
std	311.489671	2272.810090	129.140346
min	0.000000	0.000000	1.000000
25%	403.000000	900.000000	5.000000
50%	726.000000	1800.000000	25.000000
75%	913.000000	2600.000000	81.000000
max	1270.000000	9000.000000	4239.000000

```
In [8]: #Printing the rows and columns in a tuple
        df.shape
```

```
Out[8]: (582729, 10)
```

The dataset was then checked for missing values in each column. These values were dropped.

```
In [9]: #Checking for missing values in each column of the dataframe
        df.isna().sum()
```

```
Out[9]: Archive Date             0
        Group                    0
        Hospital HIPE            0
        Hospital                 0
        Specialty HIPE        1222
        Specialty                0
        Adult/Child              0
        Age Categorisation     603
        Time Bands               2
        Count                    0
        dtype: int64
```

```
In [10]: #Dropping all the rows with null values
         df.dropna(inplace = True)
```

```
In [11]: #Printing the total null value
         df.isnull().sum().sum()
```

```
Out[11]: 0
```

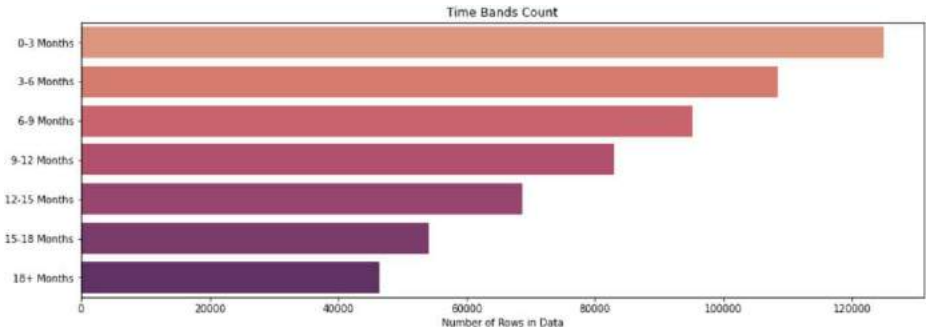
The following instructions were displayed to understand the data better on RStudio after formatting and aggregating the data.

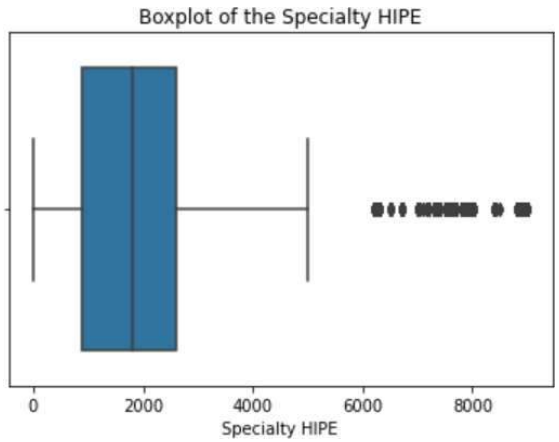
```
16 #Understanding the data
17 dim(data)
18 head(data)
19 tail(data)
20 View(data)
21 str(data)
22 summary(data)
```

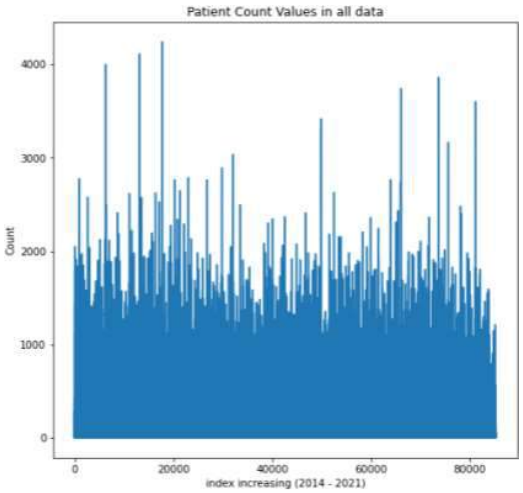
6 Data Analysis

a) Exploratory Data Analysis on Jupyter Notebook

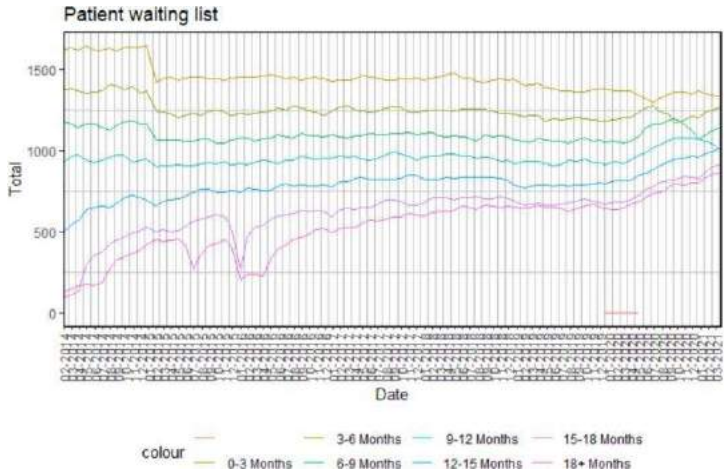
A few plots of various kind like bar charts, box plots and pie charts were plotted in order to give a better understanding of the data. To analyse how many patients each hospitals had, or what age the group the patients were mainly of.

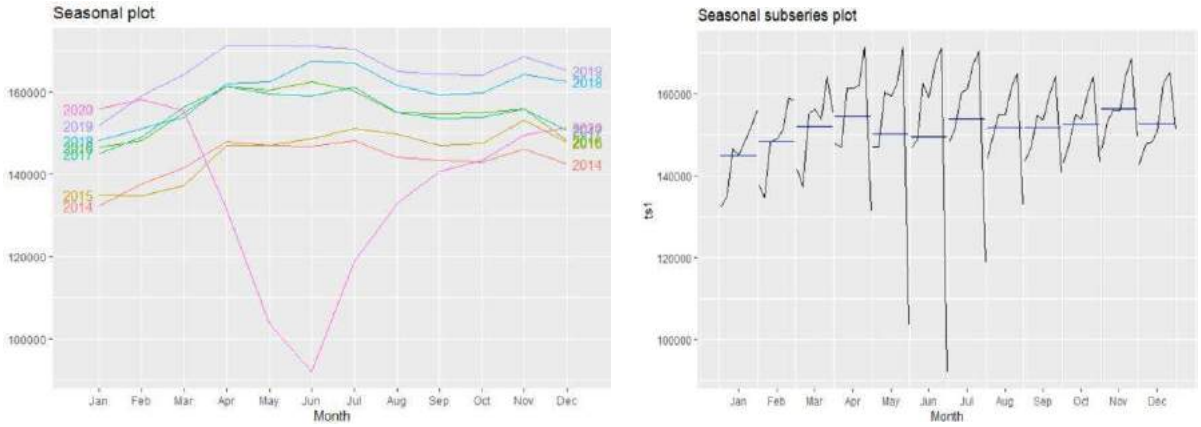






b) Plots on RStudio





These plots were displayed on RStudio to study the data. Seasonal plots and Seasonal Subseries plots were printed for each of the time series.

7 Implementation of Models

a) Implementation of Machine Learning

This section is implemented on Jupyter Notebook using python programming language. Before model building, One Hot Encoding was done to four of the categorical columns below as shown below. 6

```
In [24]: # One Hot Encoding the Group Column and joining it with main dataframe

         df.index = range(len(df))
         Group = pd.get_dummies(df['Group'])
         df = df.join(Group)

In [25]: # One Hot Encoding the Adult/Child Column and joining it with main dataframe

         adult_child = pd.get_dummies(df['Adult/Child'])
         df = df.join(adult_child)

In [26]: #One Hot Encoding the Age Categorisation Column and joining it with the main dataframe

         age_cat = pd.get_dummies(df['Age Categorisation'])
         df = df.join(age_cat)

In [27]: #One Hot Encoding the time bands Column and joining it with the main dataframe

         time_bands = pd.get_dummies(df['Time Bands'])
         df = df.join(time_bands)
```

The first model to be implemented is the Artificial Neural network, a sequential simple model with four layers. The optimizer used was adam for 100 epochs.

```
def get_regression_model():
    model = Sequential()
    model.add(Dense(128, input_shape=(X_train.shape[1],), activation='relu'))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(16, activation='relu'))
    model.add(Dense(1, activation='linear'))
    model.compile(loss='mean_squared_error', optimizer='adam', metrics=['mse'])
    model.summary()

    return model

Ann_model = get_regression_model()
Ann_model.fit(X_train, y_train, batch_size = 256, epochs=100, verbose=True, shuffle = False)
1816/1816 [==============================] - 3s 2ms/step - loss: 0.0882 - mse: 0.0882
Epoch 92/100
1816/1816 [==============================] - 3s 2ms/step - loss: 0.1604 - mse: 0.1604
Epoch 93/100
1816/1816 [==============================] - 3s 2ms/step - loss: 0.1116 - mse: 0.1116
Epoch 94/100
1816/1816 [==============================] - 4s 2ms/step - loss: 0.0999 - mse: 0.0999
Epoch 95/100
1816/1816 [==============================] - 5s 3ms/step - loss: 0.0664 - mse: 0.0664
Epoch 96/100
1816/1816 [==============================] - 4s 2ms/step - loss: 0.0900 - mse: 0.0900
Epoch 97/100
1816/1816 [==============================] - 4s 2ms/step - loss: 0.1502 - mse: 0.1502
```

To evaluate this model, the below code was implemented in order to obtain MSE, RMSE,MAE and R^2.

```
print("Mean Absolute Error: ",mean_absolute_error(y_test,y_pred))
print("Mean Square Error: ",mean_squared_error(y_test,y_pred))
print("Root Mean Square Error: ",math.sqrt(mean_squared_error(y_test,y_pred)))
print('Explained_Variance: ',explained_variance_score(y_test, y_pred))
print("r2 Score: ",r2_score(y_test,y_pred))
```

```
Mean Absolute Error:  50.919817092074304
Mean Square Error:  9515.46200632329
Root Mean Square Error:  97.54722961890455
Explained_Variance:  0.425820193286368
r2 Score:  0.4207699399879936
```

The next model is a random forest model. The model was run for 100 trees.

```
regr = RandomForestRegressor(random_state=0, verbose = 2)
regr.fit(X_train, y_train)
```

```
building tree 85 of 100
building tree 86 of 100
building tree 87 of 100
building tree 88 of 100
building tree 89 of 100
building tree 90 of 100
building tree 91 of 100
building tree 92 of 100
building tree 93 of 100
building tree 94 of 100
building tree 95 of 100
building tree 96 of 100
building tree 97 of 100
building tree 98 of 100
building tree 99 of 100
building tree 100 of 100
[Parallel(n_jobs=1)]: Done 100 out of 100 | elapsed:  1.8min finished
```

Out[35]:
```
RandomForestRegressor(random_state=0, verbose=2)
```

The evaluation for this model is shown below.

```
print("Meab Absolute Error: ",mean_absolute_error(y_test,y_pred))
print("Mean Square Error: ",mean_squared_error(y_test,y_pred))
print("Root Mean Square Error: ",math.sqrt(mean_squared_error(y_test,y_pred)))
print('Explained_Variance: ',explained_variance_score(y_test, y_pred))
print("r2 Score: ",r2_score(y_test,y_pred))
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
```

```
Meab Absolute Error:  22.798456703455045
Mean Square Error:  3361.1843943141903
Root Mean Square Error:  57.97572245616427
Explained_Variance:  0.7953969384235459
r2 Score:  0.7953962679756108
```

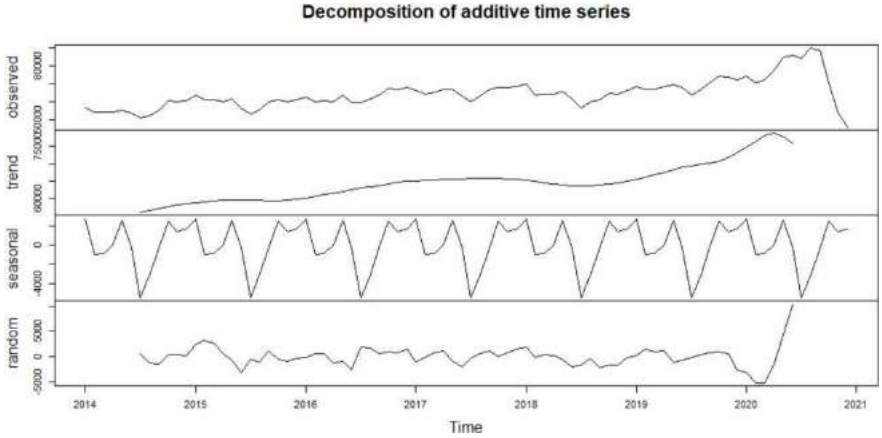
The final supervised machine learning algorithm implemented is Linear Regression. Thecode is shown below along with the evaluation metrics.

```
In [38]: #Building and Fitting Linear Regression Model

         from sklearn.linear_model import LinearRegression
         reg = LinearRegression()
         reg.fit(X_train, y_train)

Out[38]: LinearRegression()

In [39]: #Getting predictions from trained Linear Regression model and using evaluation matrix

         y_pred=reg.predict(X_test)

         print("Meab Absolute Error: ",mean_absolute_error(y_test,y_pred))
         print("Mean Square Error: ",mean_squared_error(y_test,y_pred))
         print("Root Mean Square Error: ",math.sqrt(mean_squared_error(y_test,y_pred)))
         print('Explained_Variance: ',explained_variance_score(y_test, y_pred))
         print("Score: ",r2_score(y_test,y_pred))

Meab Absolute Error:  1.147230896443099e-12
Mean Square Error:  2.7366821122998875e-24
Root Mean Square Error:  1.6542920275150598e-12
Explained_Variance:  1.0
Score:  1.0
```

b) Implementation of Time Series

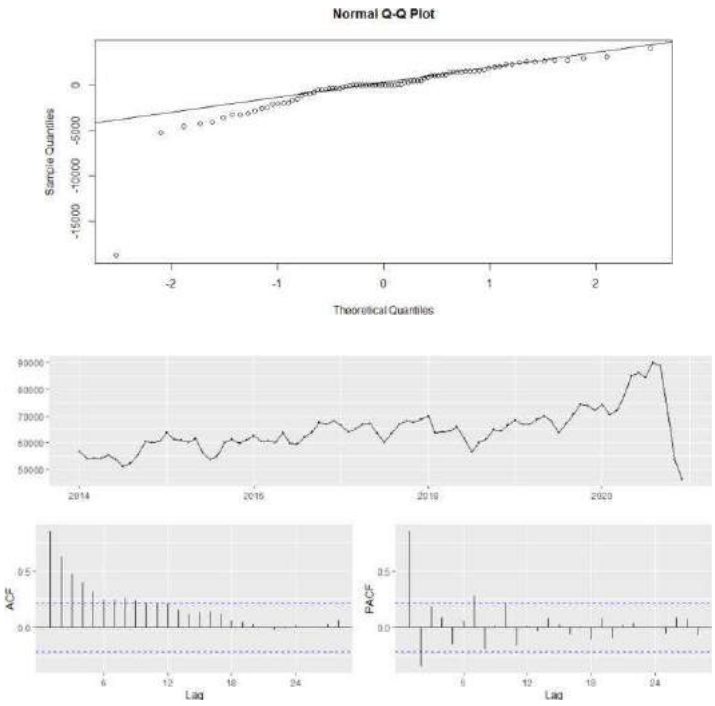
This section was implemented on RStudio using R programming language. For the time series implementation, first, the additive seasonal decomposition was plotted in order to study the trends. Augmented Dickey-Fuller test was run to check the stationarity. The simple moving average plots and the seasonal plots were displayed in order to further confirm seasonality.

```
#Seasonal decomposition using decompose() - additive
fit.decadd<-decompose(ts1, type = "additive")
fit.decadd
plot(fit.decadd)

plot(ts1)

#check order of differencing required
ndiffs(ts1)

#assess stationarity
adf.test(ts1)

Acf(ts1)
Pacf(ts1)
#moving average of order 1
ggtsdisplay(ts1)
```

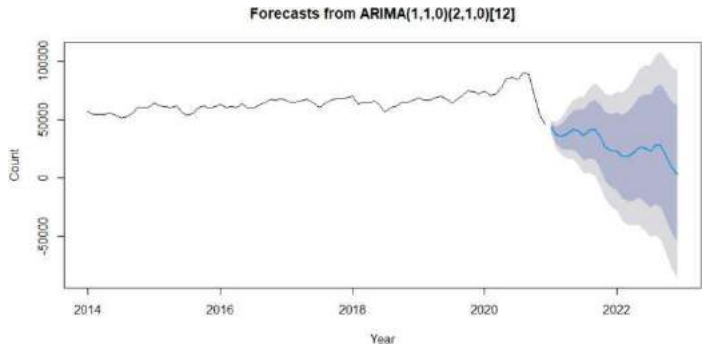


After this, a seasonal ARIMA(p,d,q)(P,D,Q)m was fit to each of the time series. For evaluation, Normal Q-Q plot was displayed along with the residuals plot which shows if the data is normally distributed. After this, the forecasting values are displayed for the next two years and then plotted. Auto.arima() function is meant to give the best model for ARIMA to compare with the manual model.

```
fit.arima1 <- arima(ts1, order=c(3,1,0), seasonal = c(0,1,1))
fit.arima1

#evaluating model
qqnorm(fit.arima1$residuals)
qqline(fit.arima1$residuals)
Box.test(fit.arima1$residuals, type="Ljung-Box")
checkresiduals(fit.arima1)
accuracy(fit.arima1)

forecast(fit.arima1,24)
plot(forecast(fit.arima1,24), xlab = "Year", ylab="Count")

fit.auto<-auto.arima(ts1)
fit.auto
ts1%>%auto.arima()%>%accuracy
```



Shown below is the forecasting plot for the next two years for the time series.



8 Conclusion

All the necessary steps to reproduce the exact implementation are clearly outlined in this document. The specific details from the installation of the software and IDE, data selection, data analysis, and implementation of the models are described in a structured manner.

References

Harrou, F., Dairi, A., Kadri, F. and Sun, Y., 2020. Forecasting emergency department overcrowding: A deep learning framework. *Chaos, Solitons & Fractals*, *139*, p.110247.