Kovai.co Data scientist Interview Assessment

Exploration & Technical documentation

Code repository

https://github.com/Sanjay-Nandakumar/Datascientist-Interview-Assesments

Experimented & Documented by:

Sanjay Kumar N

Table of Contents

Assessment 1

Problem statement

Predict the daily total patronage figure (number of boardings) on the Light Rail Network in Canberra

Objective

- The algorithm that predicts daily patronage for the next day, 7 days, next 14 days, and next 30 days with three predictions P10, P50, and P90
- Utilise the right error metrics
- Python code in GitHub
- A project report that highlights the output of exploratory data analysis, any interesting insights with visualisation, algorithm technical documentation and prediction output
- **Data source:** URL

Exploratory Data Analysis

The goal of this EDA process involves uncovering its key characteristics, identifying patterns and relationships, and detecting any anomalies or issues that might impact further analysis.

Discover patterns and trends: EDA helps us visually and statistically observe data trends, correlations between variables, and potential groupings within the data. This lets us formulate hypotheses and guide further analysis.

Identify outliers and anomalies: Unusual data points can skew results and distort models. EDA helps you spot outliers and assess their potential impact, allowing us to decide on appropriate handling strategies.

Understand data distribution: Visualizations and descriptive statistics in EDA clearly show how data is distributed (e.g., normal, skewed) across different variables. This knowledge is crucial for choosing appropriate statistical methods and interpreting results accurately.

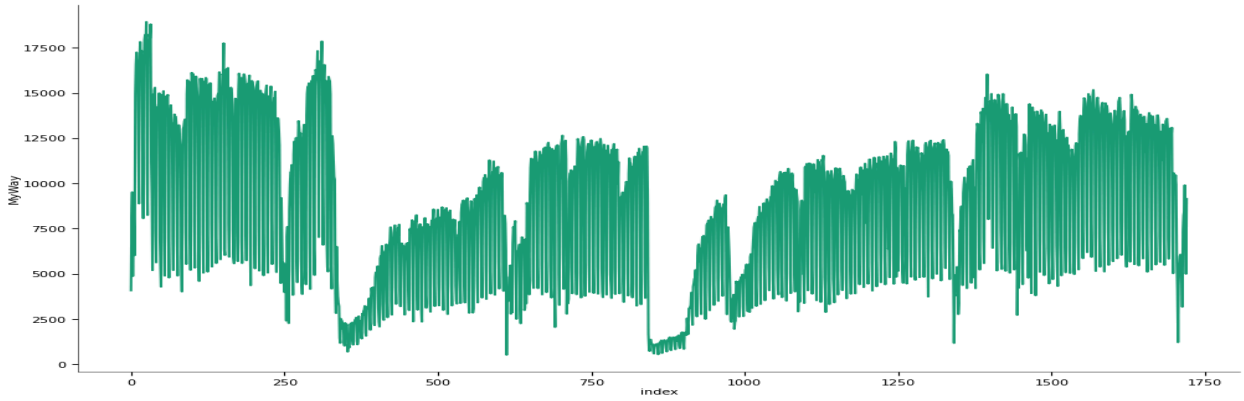
Check data quality: EDA helps you identify missing values, inconsistencies, and errors in the data. This lets us clean and pre-process the data before moving on to more advanced analysis.

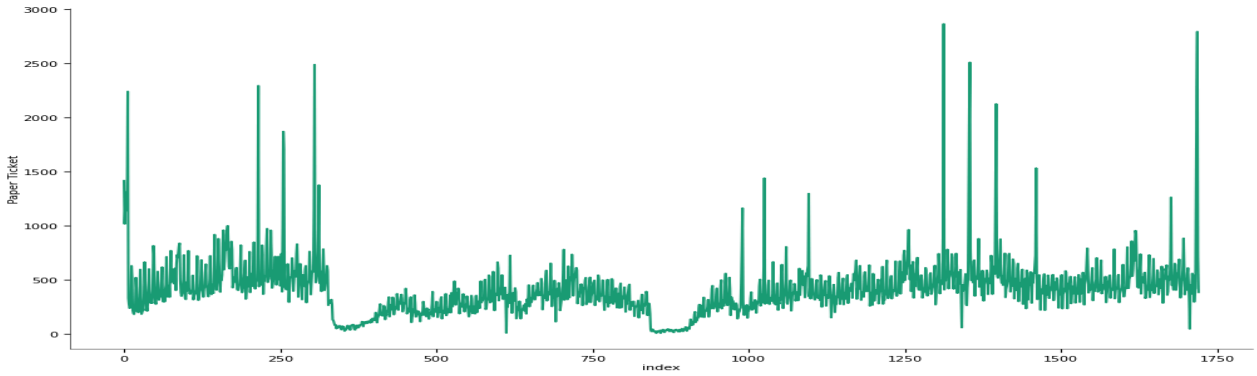
Inform model building: Insights gained from EDA can guide us in selecting relevant features, building efficient models, and evaluating their performance. Generate hypotheses and questions: EDA is an iterative process. As you explore the data, new questions and hypotheses may arise, leading to further investigations and deeper understanding.

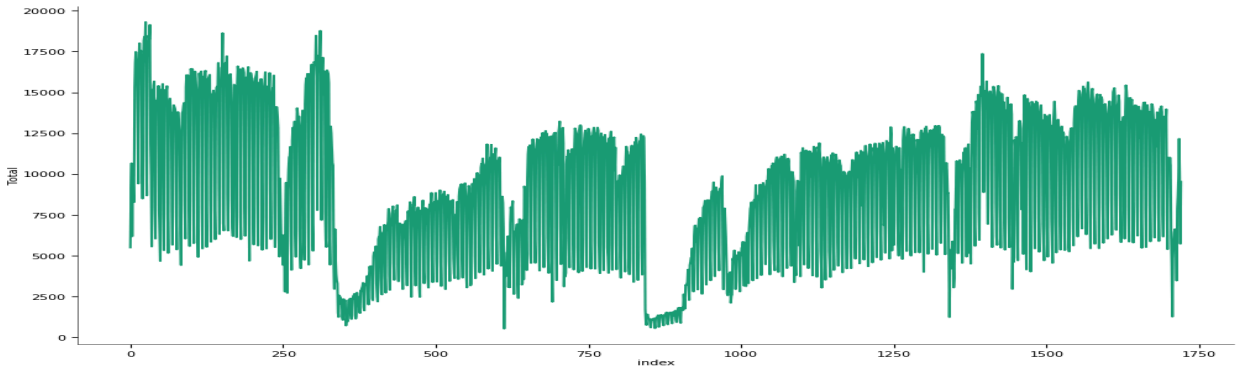
Section 1 - Checking the Shape of the data

- There are 1721 records in the dataset
- There are 4 columns - Date, My Way, Paper Ticket and Total
- Total is the sum of My Way and Paper Ticket
- The date is simply a time dimension
- The earliest record available is April 2019
- The last record available is January 2024
- Hence, it is roughly 57 months of data & can be considered as a time series.

index	Date	MyWay	Paper Ticket	Total
0	22/04/2019	4075	1418	5493
1	23/04/2019	8134	1018	9152
2	24/04/2019	9518	1147	10665
3	25/04/2019	4866	1313	6179
4	26/04/2019	9074	1143	10217
5	27/04/2019	7124	1320	8444







Section 2 - Checking the Missing Values

- Fortunately, there are no missing values/empty values/Nan/Inf present in the dataset.

```
Date            0
MyWay           0
Paper Ticket    0
Total           0
dtype: int64
```

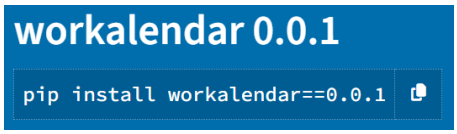
Section 3 - Checking if the day of the week has any impact on the daily Patronage

- For this, we have added an extra column to detect which day corresponds to each date.
- Fortunately, there is no spike or breakage seen for any particular day.
- Hence, we can consider the Day has no impact on the time series flow.

index	Date	Day
0	2019-05-01 00:00:00	Wednesday
1	2019-05-02 00:00:00	Thursday
2	2019-05-03 00:00:00	Friday
3	2019-05-04 00:00:00	Saturday
4	2019-06-01 00:00:00	Saturday
5	2019-06-02 00:00:00	Sunday

Section 4 - Checking if the public holidays have any impact on the daily Patronage

- For this, we have used a library called “Workalendar” which gives the list of all the public holidays in Western Australia & that are very near to our place of interest - Canberra



- These are the general public holidays per year -

	index	date	name
0	5	2019-04-22	Easter Monday
1	6	2019-04-25	Anzac Day
2	7	2019-06-03	Western Australia Day
3	8	2019-12-25	Christmas Day
4	9	2019-12-26	Boxing Day
5	0	2020-01-01	New year
6	1	2020-01-26	Australia Day
7	2	2020-01-27	Australia Day shift
8	3	2020-03-02	Labour Day
9	4	2020-04-10	Good Friday
10	5	2020-04-13	Easter Monday
11	6	2020-04-27	Anzac Day

- There are a total of 49 holidays present during our data collection period across 57 months.
- Out of that the set operation in Python shows 3 dates for missing records

```
<ipython-input-438-d5a41a32ae49>:1: UserW
  original_df = list(pd.to_datetime(data[
{Timestamp('2019-06-03 00:00:00'),
 Timestamp('2019-12-25 00:00:00'),
 Timestamp('2021-12-25 00:00:00')}
```

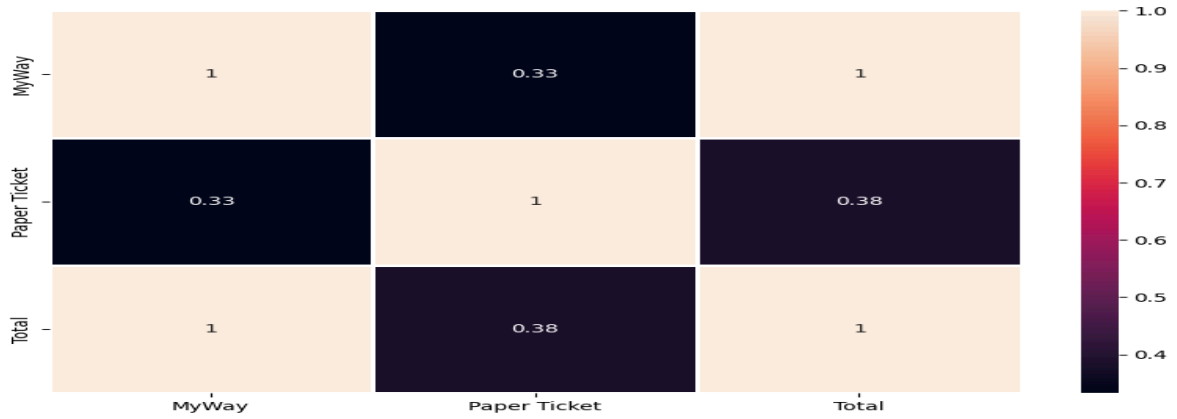
- 2019-06-03 is Western Australia Day
  2019-12-25 is Christmas Day
  2021 -12 -25 is Christmas Day

- When we cross-verified in the original dataset, the data point for 2019-06-03 was present. This might be because Workalendar is providing the data for Western Australia that might not be 100% matching with Canberra.

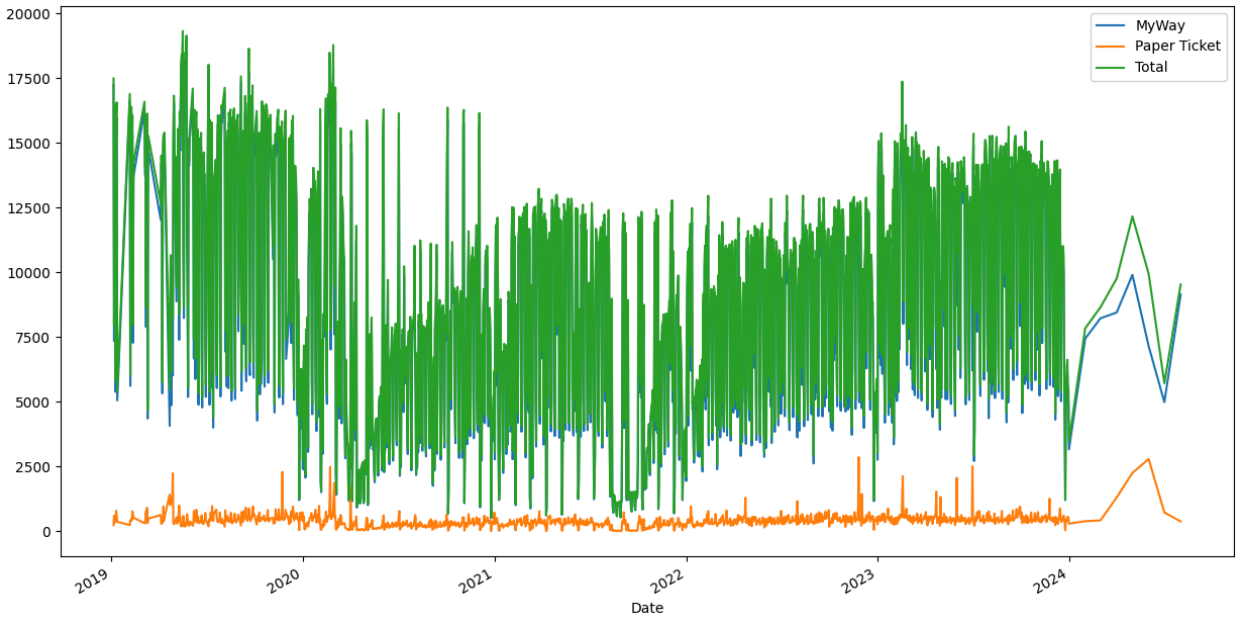
41	02/06/2019	5622	413	6035
42	03/06/2019	13762	307	14069
43	04/06/2019	14223	286	14509

- However, the data points for the 2019 & 2021 Christmas days are missing. Hence, we need to impute these 2 missing values. We have imputed these 2 missing values with the data point of 2023 Christmas.
- Apart from that, it is evident from the charts that there is no significant relationship between holidays and this time series (spike or breakage).

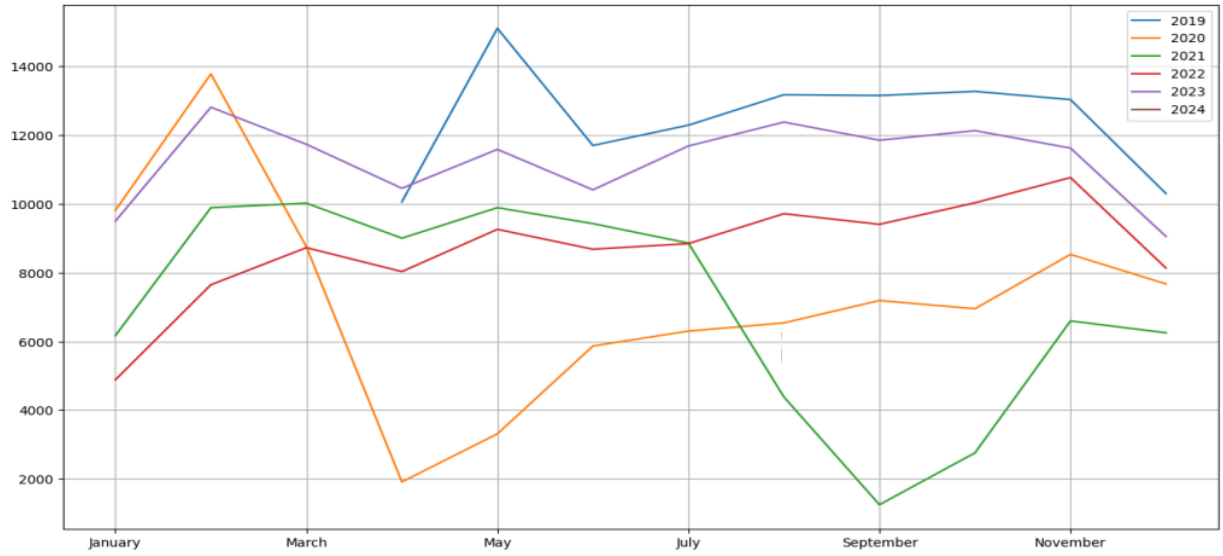
Section 5 - Checking if there is any correlation between time-dependent variables



It is clear that there are no pairs that have at least 60% correlation value. Hence, we can consider they are independent given the fact that “Total” is a summation of “My Way” & “Paper Ticket”.



Section 6 - Checking if there is any impact for the “Year” & “Month” in patronage



- It is clear that there is a slow rate of decrement in daily patronage as each year progresses (not an exact but an approximate decrement).
- 2019 had the highest and 2020/2021 had the lowest patronages.
- After covid breakout in 2020, the daily patronages seem to have come down.
- The highest month recorded is 2019 May.
- The lowest month recorded in 2021 September.
- December month in all of the years has low patronages.
- April month in all the years also seems to have a small dip.

Section 7 - Hypothesis testing for stationarity checking

While hypothesis testing isn't strictly necessary before every time series forecasting task, it can play a crucial role in several crucial steps, making it a valuable practice in many cases.

1. Understanding Data Characteristics:

Stationarity Test: Hypothesis testing helps us check for stationarity, a key requirement for many forecasting models. It allows us to reject the null hypothesis of non-stationarity and ensure the data's statistical properties remain constant over time, improving model accuracy.
Trends and Seasonality: Testing for trends and seasonality through hypothesis testing lets us confirm their presence and guides us in choosing appropriate forecasting methods or transformations.

2. Model Selection and Improvement:

Feature Selection: Identifying statistically significant features through tests helps eliminate irrelevant ones, reducing model complexity and improving its interpretability and accuracy.
Model Parameterization: Testing different model parameters and comparing their performance against null hypotheses allows us to select the optimal configurations for our forecasts.

3. Evaluating Forecast Validity:

Confidence Intervals: Hypothesis testing informs the construction of confidence intervals around forecasts, quantifying the uncertainty associated with our predictions. This transparency is crucial for decision-making based on the forecasts.
Model Comparison: Testing the null hypothesis that two forecasting models perform equally well helps us objectively compare different options and select the most reliable one.

4. Avoiding Pitfalls:

Identifying Spurious Relationships: Tests can help us avoid being misled by chance correlations or random patterns in the data, improving the overall quality and reliability of our forecasts. Overfitting: Testing different model complexities against null

hypotheses prevents overfitting, where the model memorizes the specific data instead of capturing generalizable patterns for future prediction.

In time series analysis, the distinction between stationary and non-stationary data is crucial.

Stationary time series maintain a consistent character throughout, meaning:

- **Mean and variance remain constant:** No upward or downward trends and the spread of data points stays the same. Imagine a plate of cookies where the average size and the variation in sizes stay roughly the same over time.
- **Autocorrelation fades over time:** The dependence of a data point on its past values weakens as you go further back. Think of conversations at the party; references to what someone said five minutes ago are less likely than to what they just said.
- **Non-stationary time series are vice-versa.** Their statistical properties fluctuate, making them trickier to analyze and model:
- **Trending mean or variance:** bold textThe data exhibits a clear upward or downward drift or its spread changes over time. Picture a stack of cookies gradually growing or shrinking throughout the party.
- **Persistent autocorrelation:** The influence of past values lingers, meaning distant data points can still significantly impact current ones. Imagine party gossip spreading long after the initial source, influencing conversations throughout the night. Here's a table summarizing the key differences:

Feature	Stationary Time Series	Non-Stationary Time Series
Mean and variance	Constant	Trending or changing
Autocorrelation	Fades over time	Persistent

Why does Stationarity matter?

Stationarity is important for several reasons:

Reliable analysis: Stationary data allows for the use of simpler and more reliable statistical methods for analysis and forecasting. Non-stationary data can lead to misleading results.

Model stability: Models built on stationary data are more likely to be stable and accurate over time. Non-stationary data can lead to models that quickly become outdated.

Easier interpretation: Stationary data is easier to interpret and understand, as its patterns and relationships are consistent. Non-stationary data can be more complex and challenging to decipher. So, how do you handle non-stationary data?

There are ways to transform non-stationary data into stationary form. Common techniques include

Differencing: This involves taking the difference between consecutive data points, often removing trends and seasonality. Think of it as calculating the change in cookie stack size between each observation.

Logarithmic transformation: This can stabilize the variance of non-stationary data, making it more predictable. Imagine plotting cookie sizes on a logarithmic scale to compress the spread.
We have used the Augmented Dicky Fuller test to check the stationarity with the below Hypothesis-
NullHypothesis: The series has a unit root which means the series is non-stationary.
Alternative Hypothesis: The series has no unit root which means the series is stationary

```
ADF Test Statistic : -3.586004119231434
p-value : 0.006034305708287155
#Lags Used : 21
Number of Observations Used : 1699
strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data has no unit root and is stationary
```

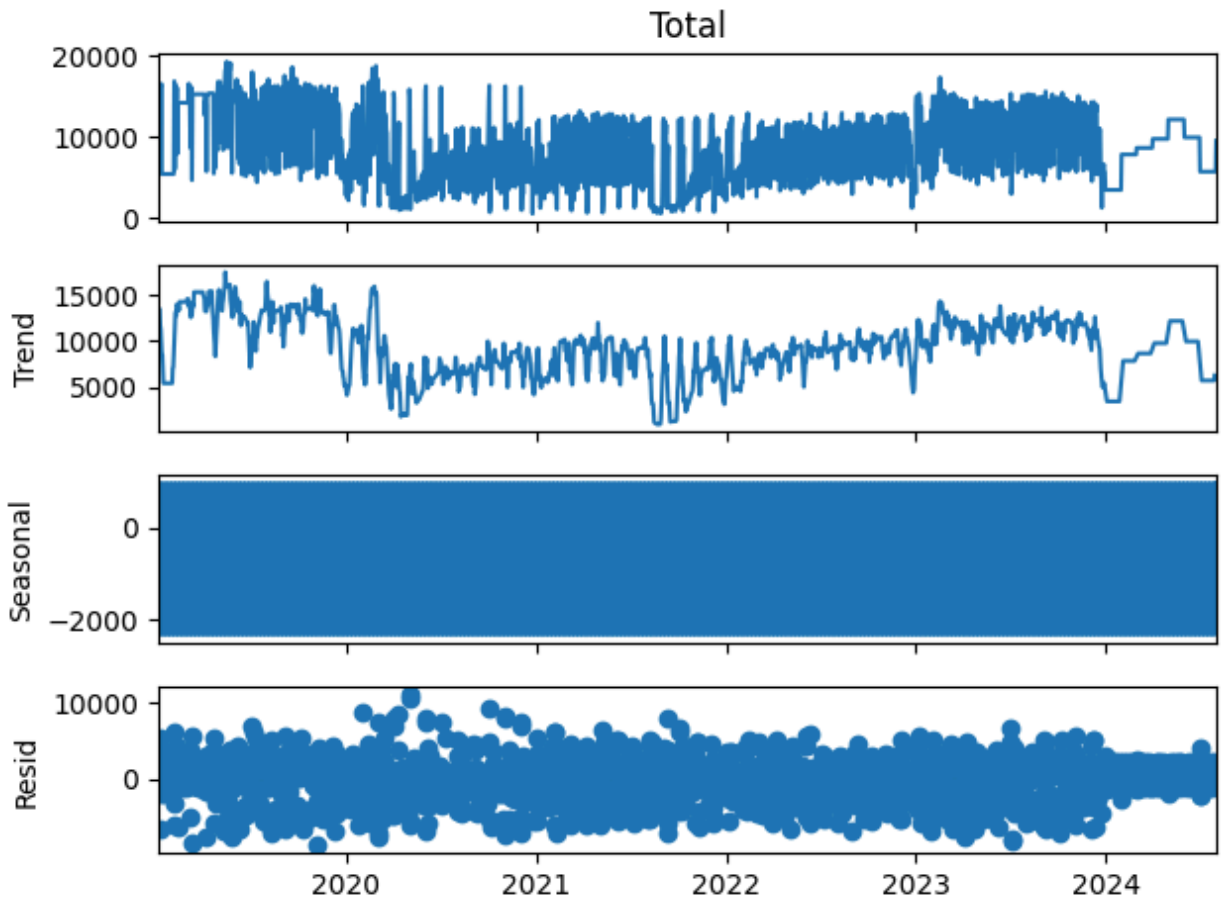
Section 8 - Decomposing the Time series

The main components of a time series that decomposition

Trend: This captures the long-term, sustained upward or downward movement.
Seasonality: This represents recurring patterns over specific periods, like daily, weekly, or annual fluctuations.
Cyclical: These are fluctuations not strictly tied to regular intervals, but rather longer-term oscillations.
Irregularity or Noise: These are random, unpredictable variations that don't fit any specific pattern.

The need for time series decomposition

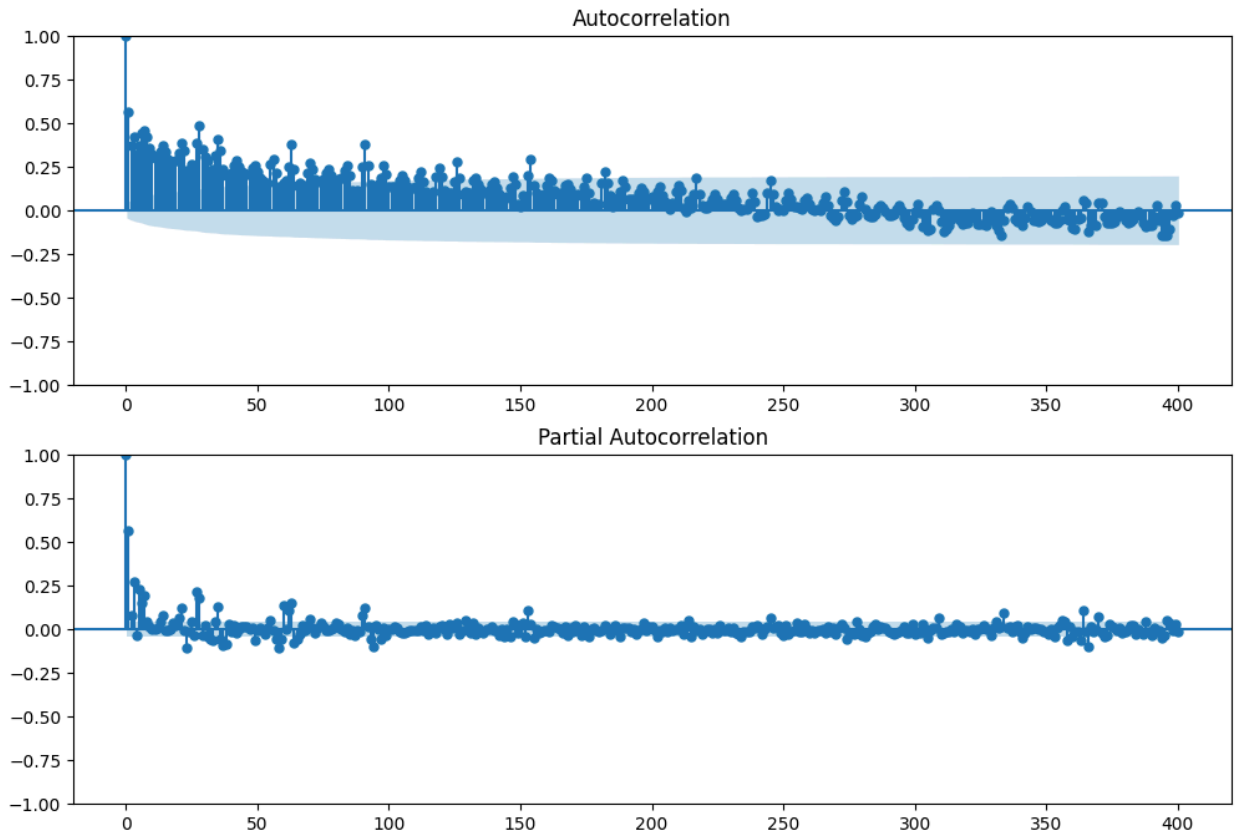
- Improved understanding
- Data anomaly detection
- Model selection and simplification



If the magnitude of the seasonal component changes with time, then we can consider the time series to be multiplicative. Otherwise, it is addictive.

ACF (Autocorrelation Function):
Measures the correlation between a time series at a given time t and its values at previous times (lags). It shows how much the current value depends on its past values.

PACF (Partial Autocorrelation Function):
Similar to ACF, it measures the partial correlation between a time series at a given time t and its values at previous times (lags), controlling for the values at all shorter lags. It helps identify the presence of autoregressive (AR) terms in a time series model. A significant PACF at lag k suggests that there's a direct relationship between the current value and the value k periods ago, even after accounting for the influence of all previous values.



We will be using these values when we implement the ARIMA model.

Section 9 - Model 1: Forecasting the values with Vector Auto regression

A VAR model, or Vector Autoregression model, is a powerful tool for forecasting multiple time series that influence each other, in contrast to traditional models that focus on single series.
Here's a breakdown of the key aspects of a VAR model:

Multiple variables: As mentioned, VAR models handle two or more time series simultaneously. This allows you to understand the interdependencies between them and how they evolve together.

Autoregressive nature: Each variable in the model is explained by its past values (called lags) and the past values of all other variables. This reflects the idea that future values are influenced by what happened previously in the system.

Equation systems: A VAR model consists of one equation for each variable, where the left side represents the future value and the right side includes lagged values of all variables and a constant term.

Strengths of the VAR model

- Captures interdependencies between multiple time series.
- Relatively easy to understand and implement.
- Flexible and adaptable to various data types.

Weaknesses of the VAR model

- Can be computationally expensive for large datasets.
- Requires careful selection of the lag order (number of past values used).
- Sensitive to outliers and structural breaks in the data.

Hypothesis for 3 variables: For double confirmation, we run 3 ADF tests for My Way, Paper Ticket & Total to ensure they are stationary.

```
Augmented Dickey-Fuller Test:
ADF test statistic        -3.681943
p-value                    0.004374
# lags used               21.000000
# observations          1699.000000
critical value (1%)       -3.434205
critical value (5%)       -2.863243
critical value (10%)      -2.567676
Strong evidence against the null hypothesis
Reject the null hypothesis
ts_input_data has no unit root and is stationary


Augmented Dickey-Fuller Test:
ADF test statistic        -3.150879
p-value                    0.023010
# lags used               21.000000
# observations          1699.000000
critical value (1%)       -3.434205
critical value (5%)       -2.863243
critical value (10%)      -2.567676
Strong evidence against the null hypothesis
Reject the null hypothesis
ts_input_data has no unit root and is stationary


Augmented Dickey-Fuller Test:
ADF test statistic        -3.586004
p-value                    0.006034
# lags used               21.000000
# observations          1699.000000
critical value (1%)       -3.434205
critical value (5%)       -2.863243
critical value (10%)      -2.567676
Strong evidence against the null hypothesis
Reject the null hypothesis
ts_input_data has no unit root and is stationary
```

Hence, it is confirmed that we can reject the null hypothesis with strong confidence.

Section 10 - Train & Test Split of the Time Series Data

Splitting time series data for training and testing requires a different approach than standard data because of the inherent temporal dependence between observations. Randomly splitting won't work here, as it disrupts the time order and creates unrealistic scenarios for testing. Here are some methods for efficiently splitting time series data:

1. Simple Fixed Split: This is the easiest method, where you define a percentage of the data for training and the remaining for testing. For example, 80% for training and 20% for testing. While simple, it might not be the most statistically sound, as the chosen split point might influence the model's performance.

2. Time-based Split: This method respects the temporal order of your data. You define a specific date or timestamp as the split point, ensuring the training data is

chronologically before the testing data. This approach is effective for simulating real-world situations where you wouldn't have access to future data for training.

3. Rolling Window Split: This method involves creating multiple training and testing sets by sequentially moving a window of data across the entire time series. Each window becomes a separate train-test pair, leading to a more robust assessment of the model's performance across different periods.

Splitting the data in 3:1 ratio (75%:25%) for training & testing

```
train = ts_input_data[:int(0.75*(len(ts_input_data)))]
valid = ts_input_data[int(0.25*(len(ts_input_data))):]
```

We are splitting the ratio of 75% into training & 25% into testing.

Section 11 - Performance evaluation of the VAR model

We are using 4 metrics here-

1. Root mean squared error
2. Mean absolute error
3. Mean absolute percentage error
4. Mean squared error

```
RMSE value for MyWay is: 3805.1557930621225
RMSE value for Paper Ticket is: 229.007419011055
RMSE value for Total is: 3893.8671633999847


MAE value for MyWay is: 3323.5183288734897
MAE value for Paper Ticket is: 140.8110074785864
MAE value for Total is: 3382.4055364557653


MAPE value for MyWay is: 0.4053359247520457
MAPE value for Paper Ticket is: 0.3790625593252592
MAPE value for Total is: 0.39463504853397907


MSE value for MyWay is: 14479210.609474229
MSE value for Paper Ticket is: 52444.39796210491
MSE value for Total is: 15162201.486204643
```

Since, RMSE, MSE & MAE are magnitude-oriented, let's look at MAPE to see how much deviation is prediction has from the actual.

-
- Myway - 40% MAP error
- Paper ticket - 37% MAP error
- Total - 39% MAP error
-

The model is looking decent although there are scopes for further improvement that we can consider as a part of future enhancement.
But we have generated the predictions as mentioned in the required documents

Forecast for 7 days

	MyWay	Paper Ticket	Total
0	9096.783592	403.872789	9500.656381
1	9018.747976	412.691009	9431.438985
2	8942.398536	416.150299	9358.548836
3	8879.016492	416.719862	9295.736354
4	8831.440730	415.938792	9247.379522
5	8798.376025	414.694240	9213.070265
6	8776.932972	413.447097	9190.380069

Forecast for 14 days

	MyWay	Paper Ticket	Total
0	9096.783592	403.872789	9500.656381
1	9018.747976	412.691009	9431.438985
2	8942.398536	416.150299	9358.548836
3	8879.016492	416.719862	9295.736354
4	8831.440730	415.938792	9247.379522
5	8798.376025	414.694240	9213.070265
6	8776.932972	413.447097	9190.380069
7	8763.990337	412.398639	9176.388977
8	8756.825931	411.604752	9168.430683
9	8753.327613	411.049005	9164.376618
10	8751.987514	410.686083	9162.673598
11	8751.802702	410.465343	9162.268046
12	8752.150988	410.341935	9162.492923
13	8752.675804	410.280716	9162.956520

Forecast for 30 days

index	MyWay	Paper Ticket	Total
0	9096.783591519927	403.8727893432361	9500.656380863158
1	9018.747975745824	412.6910089491853	9431.438984695003
2	8942.398536308521	416.1502993565669	9358.548835665082
3	8879.016491921397	416.71986224049607	9295.736354161887
4	8831.440729901282	415.9387920053335	9247.379521906609
5	8798.376024822712	414.6942404920304	9213.070265314735
6	8776.932971928634	413.44709739700545	9190.380069325634
7	8763.990337430827	412.3986393704635	9176.388976801283
8	8756.825930763687	411.60475238434464	9168.430683148024
9	8753.327612695124	411.0490054269762	9164.376618122093
10	8751.98751419515	410.686083390777	9162.673597585923
11	8751.80270211029	410.46534340820114	9162.268045518487
12	8752.150988324545	410.3419345963562	9162.492922920896
13	8752.67580372615	410.2807162003177	9162.956519926462
14	8753.193467967576	410.25640943321395	9163.449877400784
15	8753.625076623623	410.25208358727497	9163.87716021089
16	8753.950029805226	410.2571587515024	9164.20718855672
17	8754.176437930357	410.26551786068575	9164.441955791037
18	8754.323624290793	410.2739688757078	9164.597593166494
19	8754.412702952344	410.2811059414716	9164.693808893811
20	8754.462185573315	410.28652688697895	9164.748712460289
21	8754.486483753613	410.2903313941283	9164.776815147736
22	8754.495913242263	410.29282188770765	9164.788735129963
23	8754.497351999265	410.2943406965337	9164.791692695795
24	8754.49507778207	410.2951926929262	9164.79027047499

P10, P50 & P90 Errors

```
P10 Error % : 0.17496571789565854
P50 Error % : 0.3661847714512469
P90 Error % : 0.48798381395680 62
```

Section 12 - Model 2 - ARIMA (Auto Regressive Integrated Moving Average) & SARIMA (Seasonal ARIMA)

ARIMA and SARIMA are both powerful statistical models used for time series forecasting. They analyze and predict future values based on past observations, but they differ in their ability to handle trends and seasonality.
ARIMA (Autoregressive Integrated Moving Average):

- **Autoregressive (AR):** Takes into account the influence of past values on the current value.
- **Integrated (I):** Applies differencing to make the data stationary (mean and variance constant over time).
- **Moving Average (MA):** Considers the average of past error terms to account for random fluctuations.

Feature	ARIMA	SARIMA
Focus	Non-seasonal trends and patterns	Seasonality and non-seasonal trends
Notation	ARIMA(p, d, q)	SARIMA(p, d, q)(P, D, Q, m)
Seasonality	Not considered	Explicitly modeled
Applications	Stationary time series without strong seasonality	Time series with seasonality and/or non-stationarity

ARIMA focuses on non-seasonal trends and patterns in the data.

Uses past values of the time series and past errors to predict future values.
Represented by the notation ARIMA(p, d, q):
p: The number of autoregressive terms (past values influencing current value).
d: The degree of differencing (transforming data to make it stationary).
q: The number of moving average terms (past errors influencing current value).

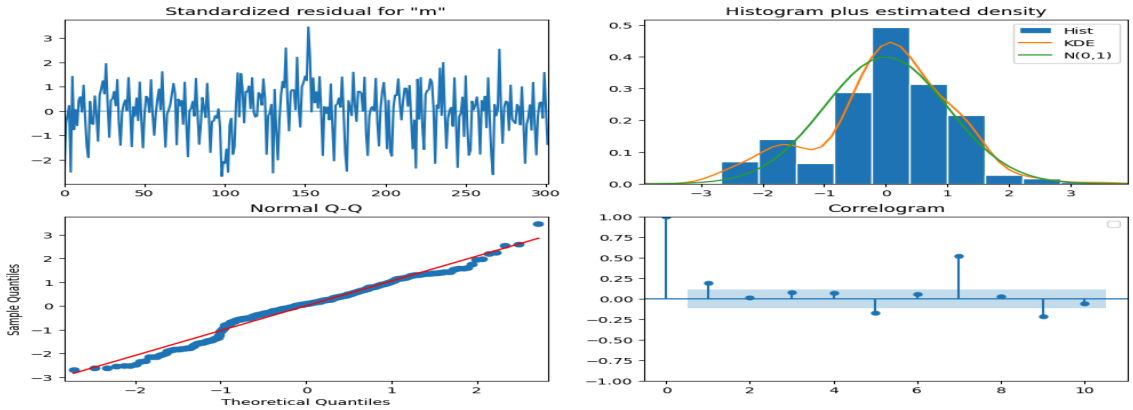
SARIMA is an extension of ARIMA that incorporates seasonality (recurring patterns at specific intervals).

(p, d, q): Same as ARIMA for non-seasonal components.
m: The seasonality period (e.g., 7 for weekly data, 12 for monthly data).

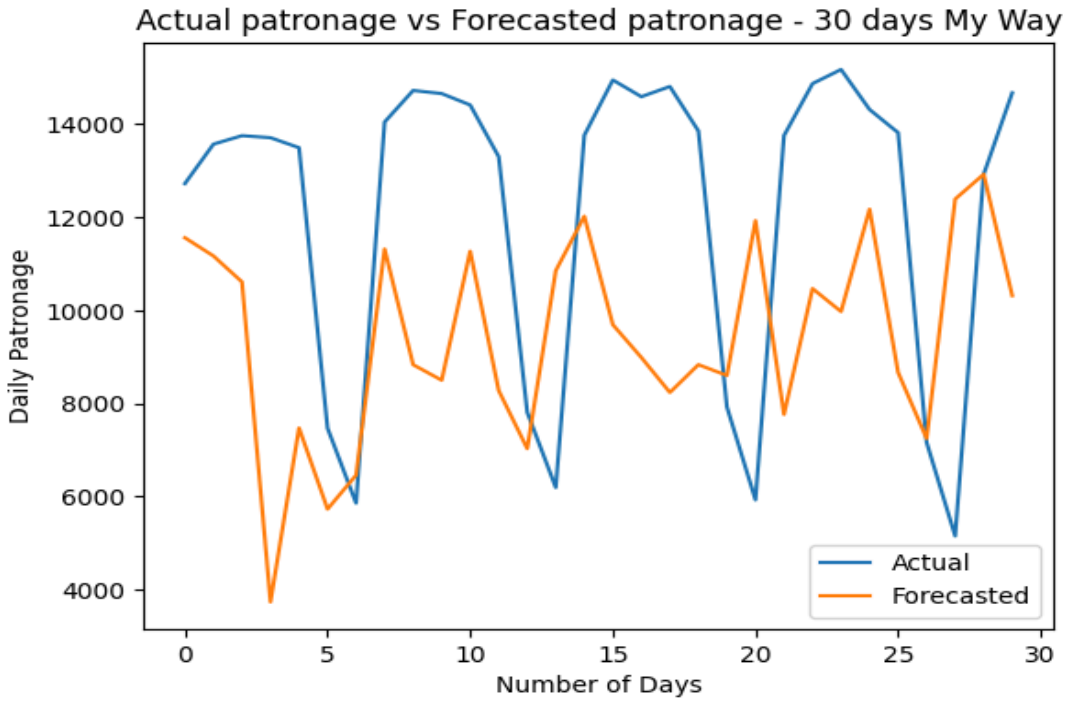
Using the Auto ARIMA package in Python to find the best parameters for the ARIMA model

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(0,0,0)[0] intercept   : AIC=8182.472, Time=0.65 sec
 ARIMA(0,1,0)(0,0,0)[0] intercept   : AIC=8438.447, Time=0.02 sec
 ARIMA(1,1,0)(0,0,0)[0] intercept   : AIC=8440.425, Time=0.03 sec
 ARIMA(0,1,1)(0,0,0)[0] intercept   : AIC=8440.373, Time=0.06 sec
 ARIMA(0,1,0)(0,0,0)[0]             : AIC=8436.452, Time=0.01 sec
 ARIMA(1,1,2)(0,0,0)[0] intercept   : AIC=inf, Time=0.97 sec
 ARIMA(2,1,1)(0,0,0)[0] intercept   : AIC=8227.748, Time=0.50 sec
 ARIMA(3,1,2)(0,0,0)[0] intercept   : AIC=8297.704, Time=0.69 sec
 ARIMA(2,1,3)(0,0,0)[0] intercept   : AIC=inf, Time=0.81 sec
 ARIMA(1,1,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.45 sec
 ARIMA(1,1,3)(0,0,0)[0] intercept   : AIC=8239.621, Time=0.63 sec
 ARIMA(3,1,1)(0,0,0)[0] intercept   : AIC=8245.706, Time=0.21 sec
 ARIMA(3,1,3)(0,0,0)[0] intercept   : AIC=8077.875, Time=1.68 sec
 ARIMA(4,1,3)(0,0,0)[0] intercept   : AIC=7986.728, Time=1.04 sec
 ARIMA(4,1,2)(0,0,0)[0] intercept   : AIC=7990.358, Time=2.52 sec
 ARIMA(5,1,3)(0,0,0)[0] intercept   : AIC=7916.340, Time=2.50 sec
 ARIMA(5,1,2)(0,0,0)[0] intercept   : AIC=7904.385, Time=2.55 sec
 ARIMA(5,1,1)(0,0,0)[0] intercept   : AIC=8023.473, Time=1.00 sec
 ARIMA(4,1,1)(0,0,0)[0] intercept   : AIC=8153.624, Time=2.37 sec
 ARIMA(5,1,2)(0,0,0)[0]             : AIC=7903.562, Time=1.96 sec
 ARIMA(4,1,2)(0,0,0)[0]             : AIC=7988.733, Time=2.01 sec
 ARIMA(5,1,1)(0,0,0)[0]             : AIC=8021.784, Time=0.79 sec
 ARIMA(5,1,3)(0,0,0)[0]             : AIC=7914.503, Time=2.52 sec
 ARIMA(4,1,1)(0,0,0)[0]             : AIC=8146.011, Time=1.12 sec
 ARIMA(4,1,3)(0,0,0)[0]             : AIC=7984.115, Time=2.37 sec
```

Fitting the ARIMA for the “My way” variable

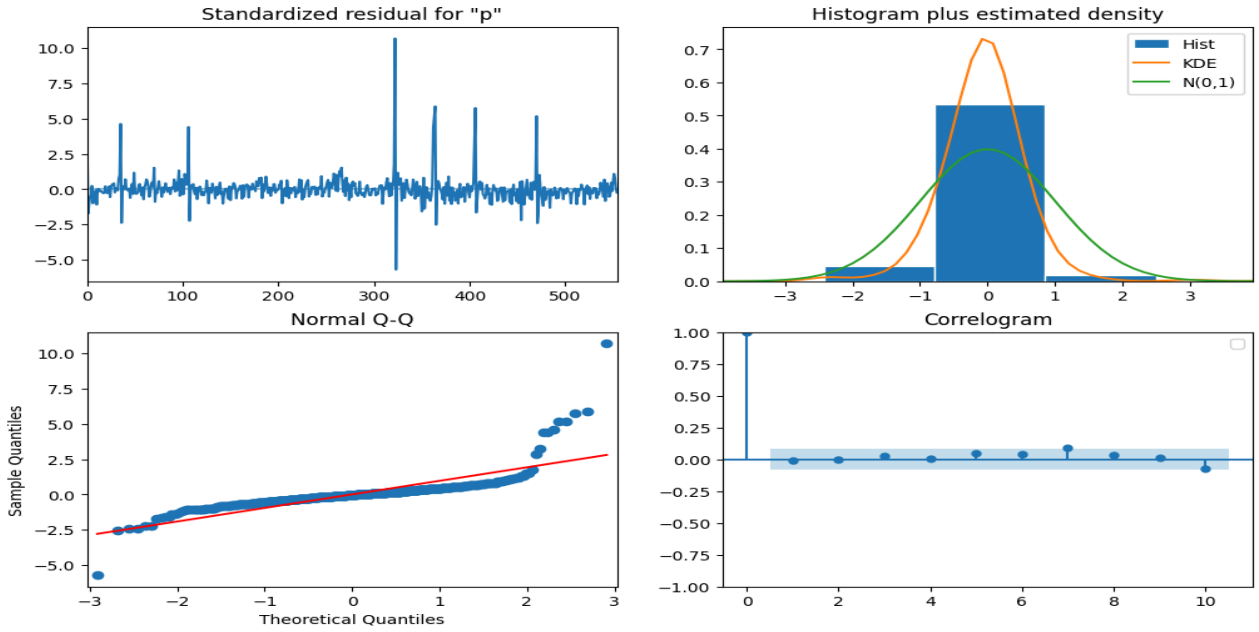


Evaluating the performance for 30 days (My way)

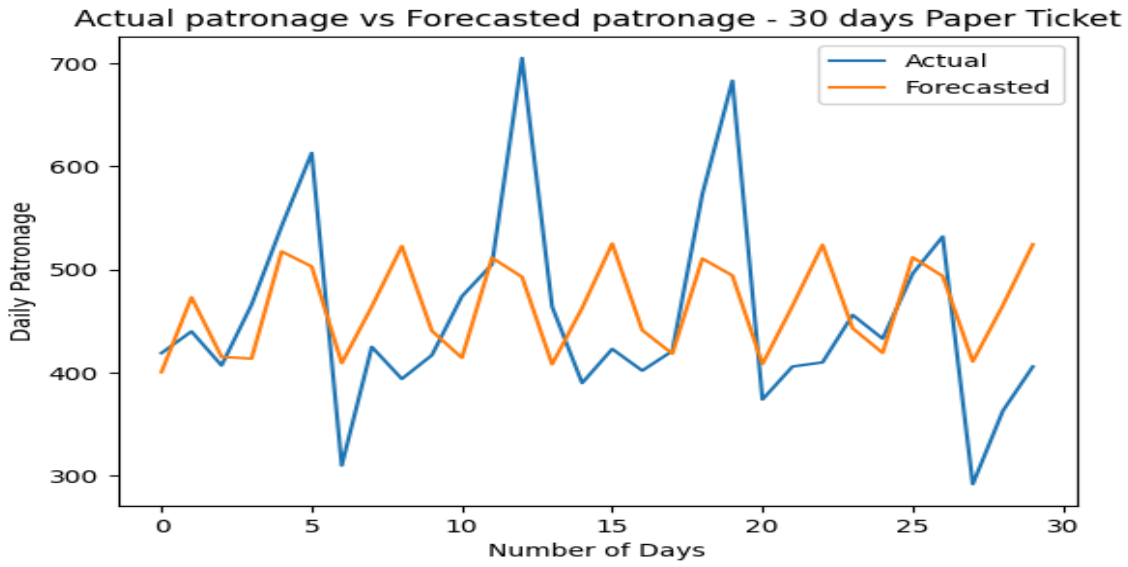



```
RMSE value is: 4639.115066248306
MAE value is: 3962.9315938828854
MAPE value is: 0.3516638206310155
MSE value is: 21521388.597892027
```

Fitting the ARIMA for the “Paper Ticket” variable

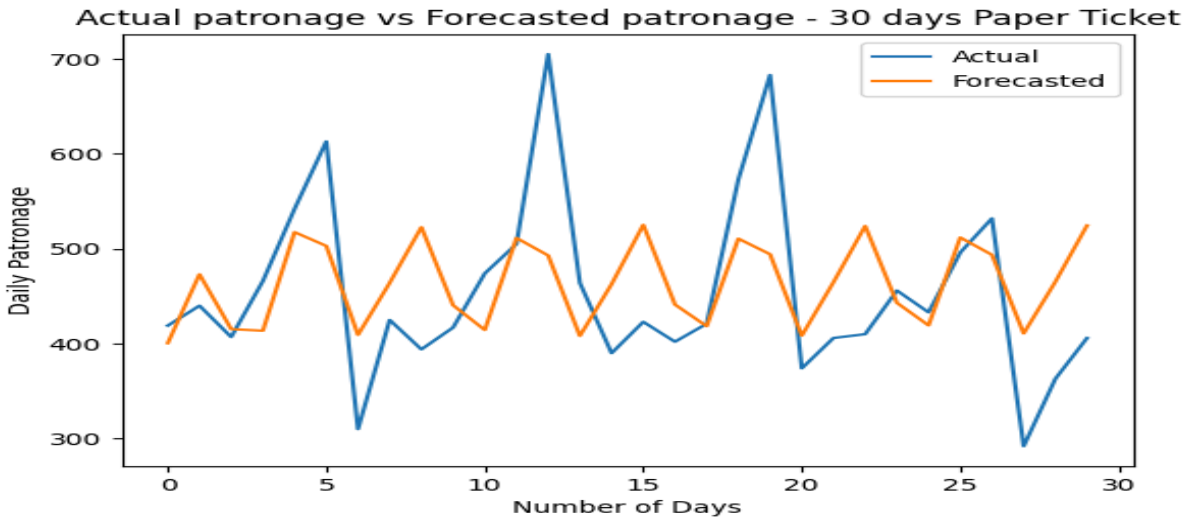


Evaluating the performance for 30 days (Paper Ticket)\




```
RMSE value is: 84.12032317825073
MAE value is: 65.55676019773212
MAPE value is: 0.1469227569373823
MSE value is: 7076.228771613348
```

Evaluating the performance for the "Total" variable by adding "My Way" and "Paper Ticket" forecast (30 days)

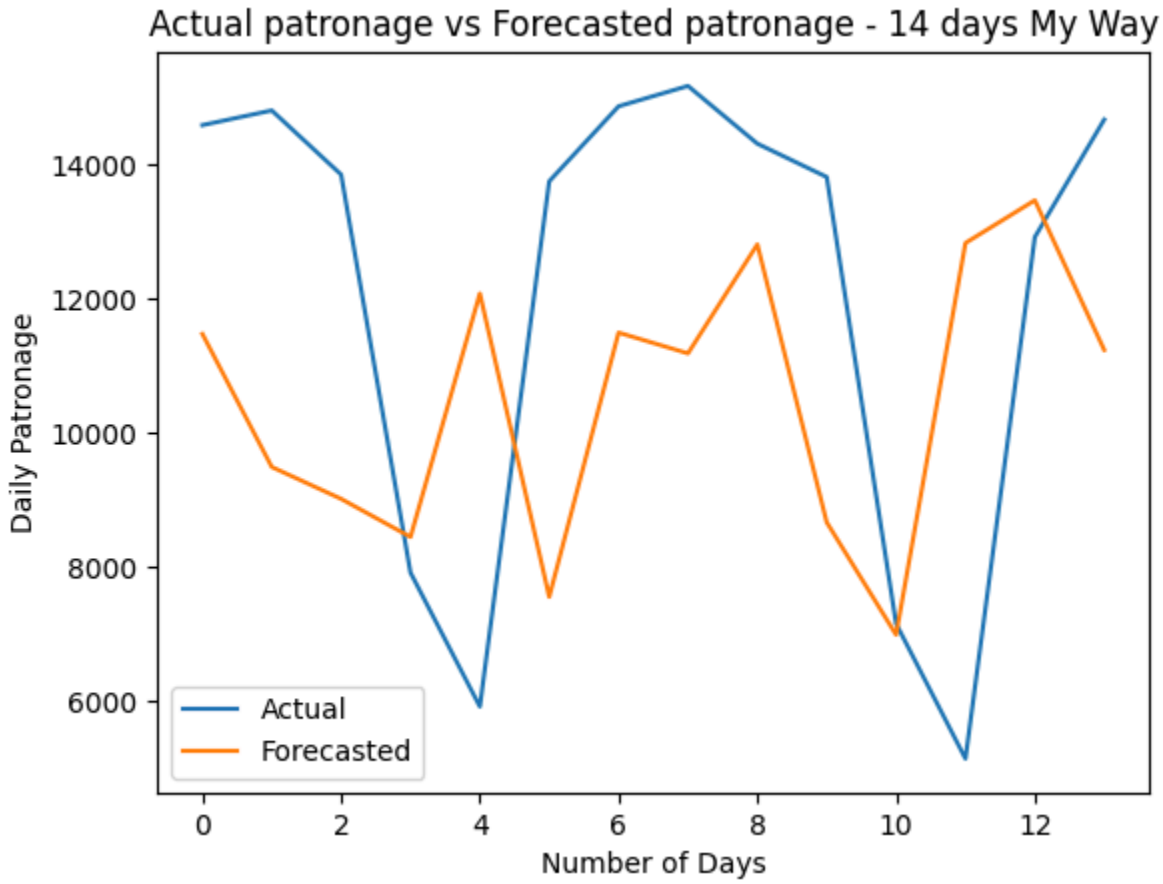



```
RMSE value is: 4632.077798486128
MAE value is: 3958.9401300480986
MAPE value is: 0.3377559963634538
MSE value is: 21456144.731228095
```

P10, P50 & P90 Errors

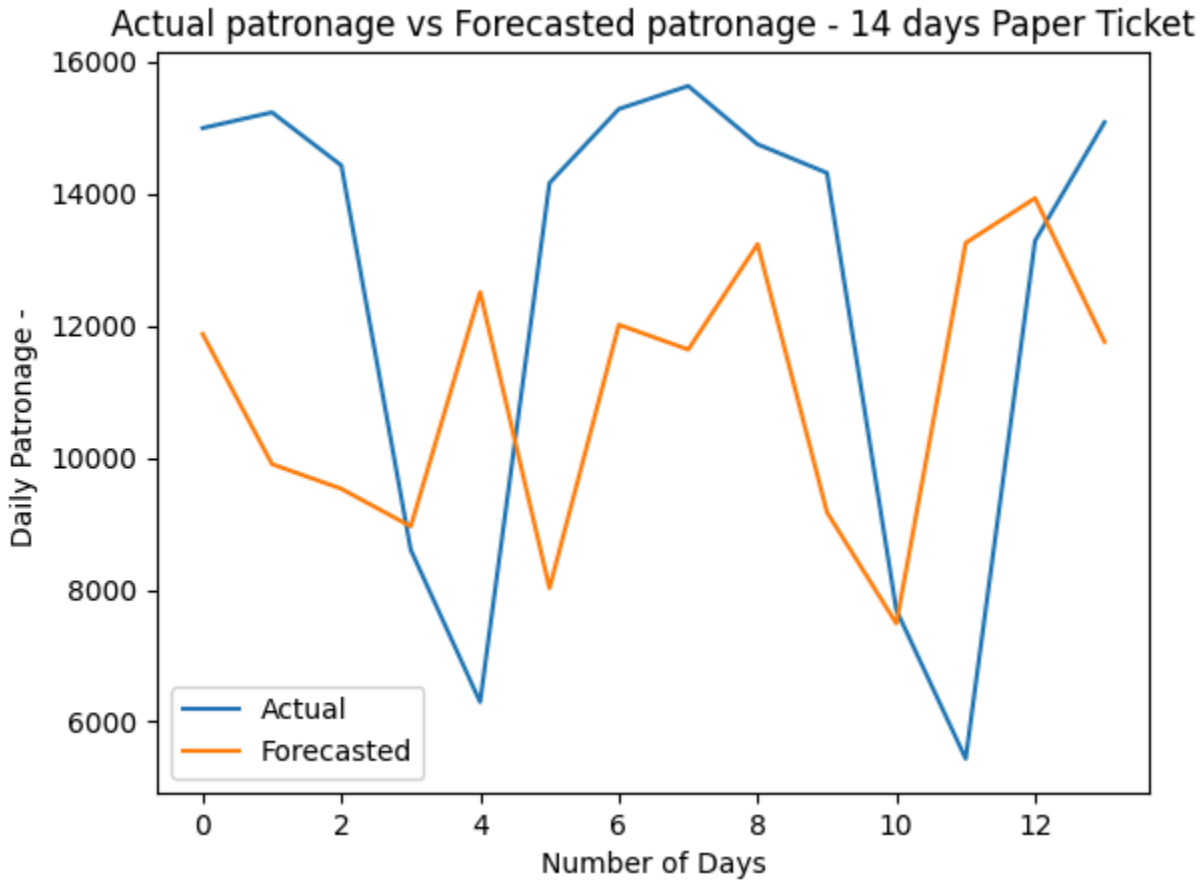
```
P10 Error % : 0.11215507212626052
P50 Error % : 0.44174318107114025
P90 Error % : 0.21937230019035275
```

Evaluating the performance for 14 days (My way)



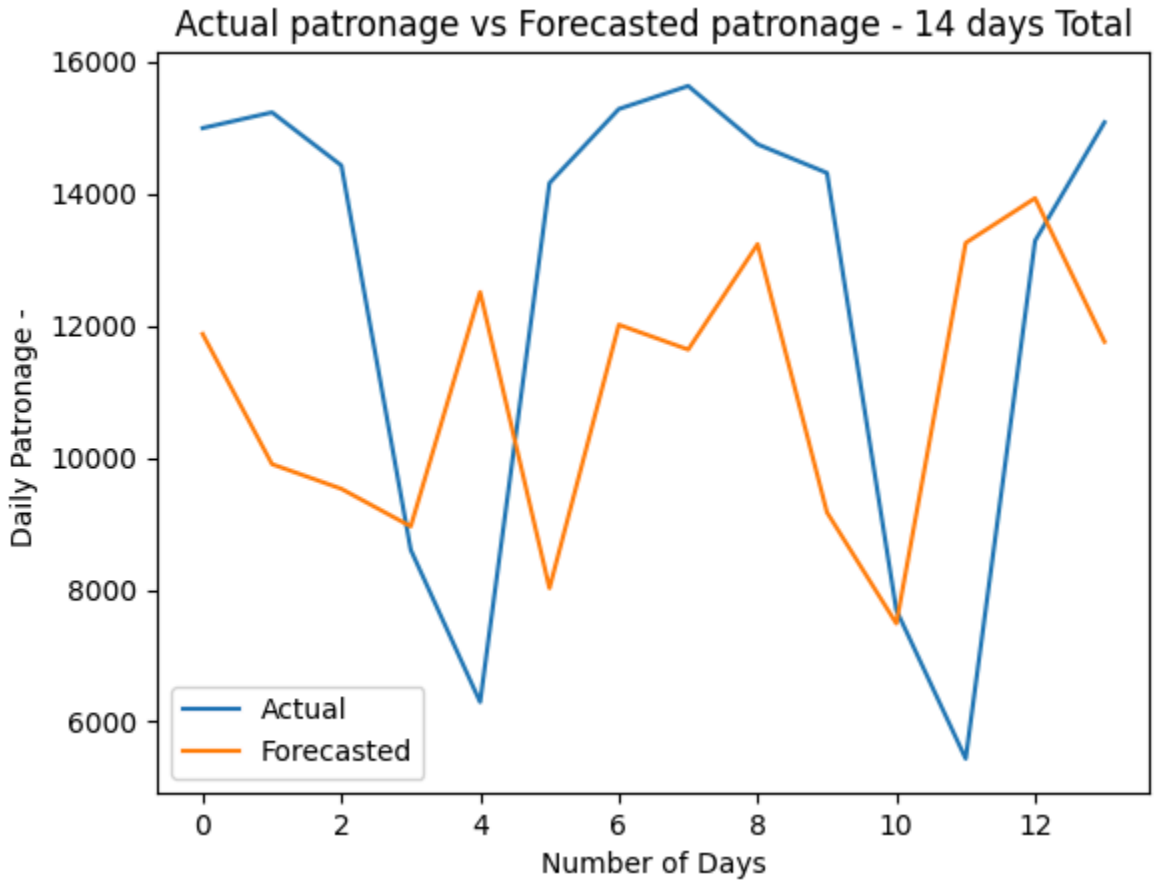

```
RMSE value is: 81.35328918402874
MAE value is: 62.60691463571128
MAPE value is: 0.1475514616639746
MSE value is: 6618.357661060209
```

Evaluating the performance for 14 days (Paper Ticket)




```
RMSE value is: 4360.799281809378
MAE value is: 3709.697337024828
MAPE value is: 0.35853431335730895
MSE value is: 19016570.376229186
```

Evaluating the performance for the "Total" variable by adding "My Way" and "Paper Ticket" forecast (14 days)

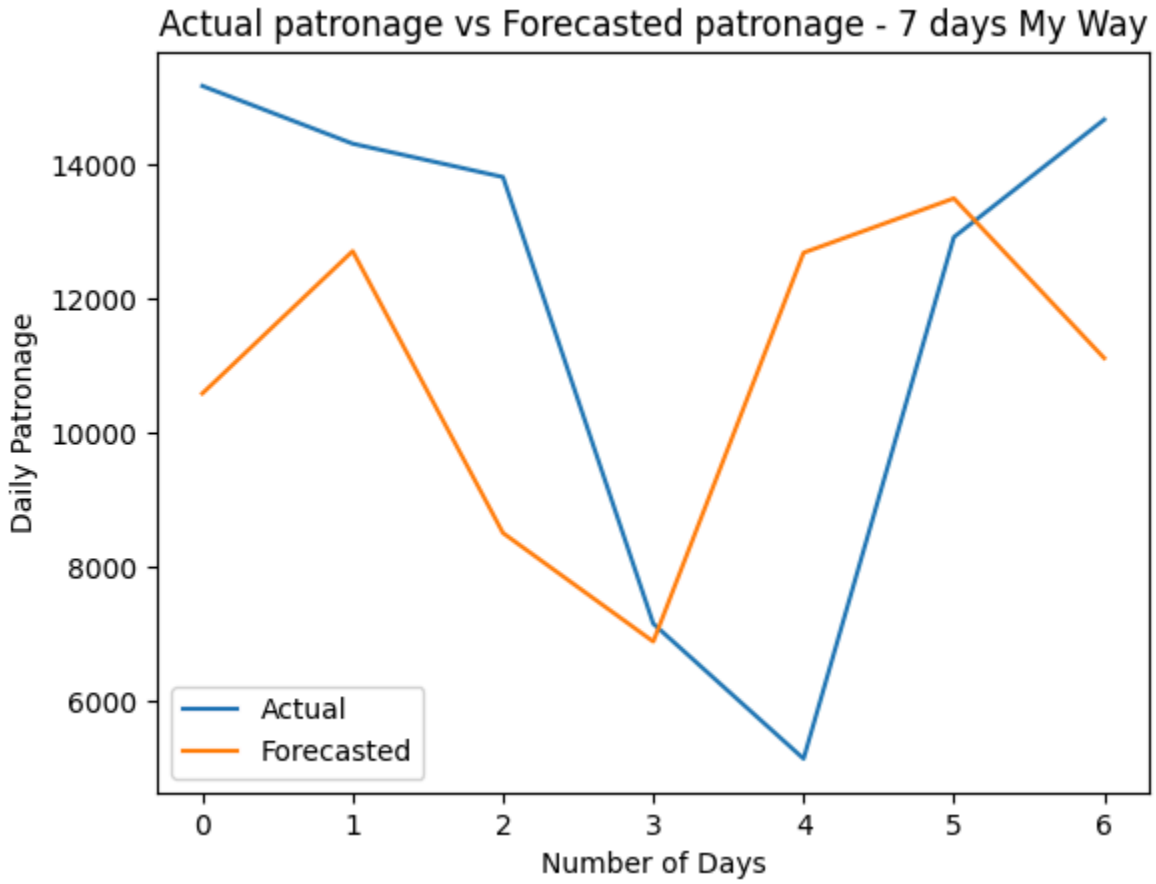



```
RMSE value is: 4360.799281809378
MAE value is: 3709.697337024828
MAPE value is: 0.35853431335730895
MSE value is: 19016570.376229186
```

P10, P50 & P90 Errors

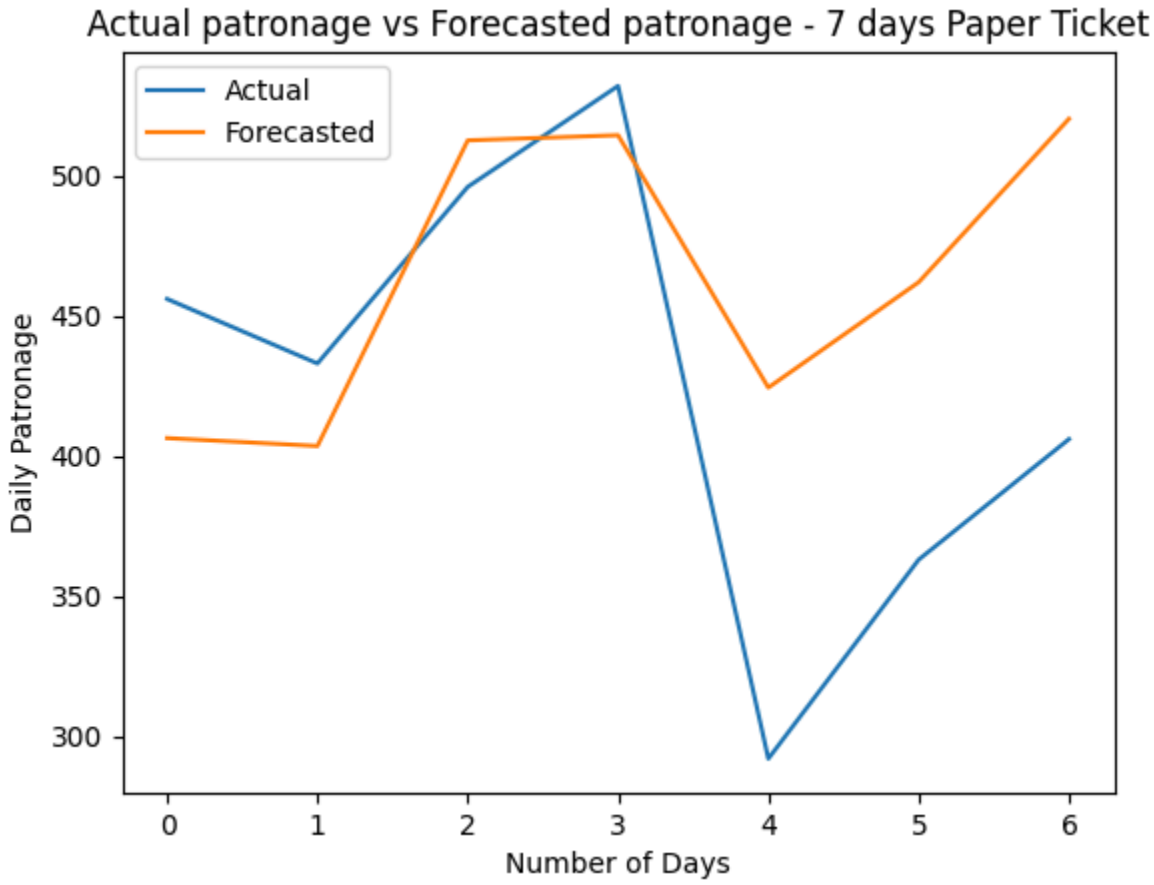
```
P10 Error % : 0.19111061540766905
P50 Error % : 0.22846022654735007
P90 Error % : 0.15231711409485837
```

Evaluating the performance for 7 days (My way)



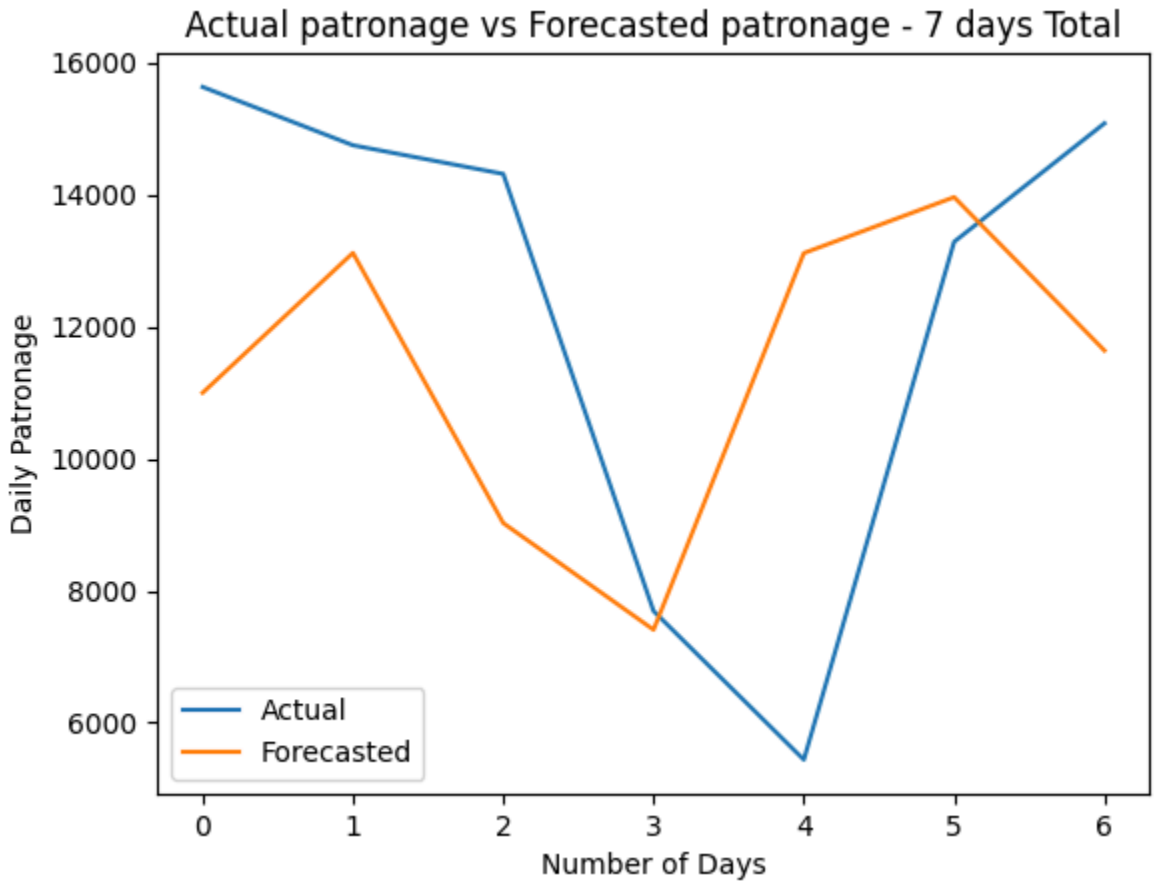

```
RMSE value is: 4167.384565681846
MAE value is: 3346.2927320490744
MAPE value is: 0.36943385183373767
MSE value is: 17367094.118283268
```

Evaluating the performance for 7 days (Paper Ticket)




```
RMSE value is: 79.54881577553834
MAE value is: 65.55368584542906
MAPE value is: 0.17868992466084932
MSE value is: 6328.014091290538
```

Evaluating the performance for the “Total” variable by adding “My Way” and “Paper Ticket” forecast (7 days)



P10, P50 & P90 Errors

```
P10 Error % : 0.18936527377440998
P50 Error % : 0.22982181117974732
P90 Error % : 0.13714330699909877
```

Section 13: Comparison of the Models

30 Days forecast

	VAR	ARIMA
P10	0.17	0.18
P50	0.36	0.22
P90	0.48	0.13

30 Days forecast

	VAR	ARIMA
MAE	3382	3374
MAPE	0.39	0.35

Clearly, ARIMA performs better than VAR

ARIMA model		
	MAPE	MAE
7 days	0.33	3958
14 days	0.35	3709
30 days	0.35	3374

Section 14: What might be the reasons behind the ARIMA's outperformance

Model Structure:

- **Focus on Individual Series:** ARIMA models are designed to capture the intricate patterns within a single time series, delving deeply into its

autoregressive (AR) and moving average (MA) components. This focused approach often leads to more accurate forecasts for that specific series.

- **Complexity of Relationships**: VAR models, on the other hand, handle multiple time series simultaneously, considering their interrelationships. This added complexity can sometimes hinder forecasting performance, especially if the relationships between the series are not strong or well-defined.

Data Characteristics:

- **Stationarity:** ARIMA models inherently address non-stationarity through differencing, making them well-suited for time series with trends or seasonality. VAR models, without explicit differencing, might struggle with non-stationary data, potentially leading to suboptimal forecasts.
- **Noise and Volatility:** ARIMA's moving average (MA) component effectively addresses random fluctuations and noise in the data, often resulting in smoother and more reliable forecasts. VAR models, lacking an explicit MA component, might be more sensitive to such noise.
- **Model Selection and Estimation:** ARIMA models typically involve fewer parameters and are often easier to identify and estimate, especially for univariate time series. VAR models, with their multiple equations and potential for overfitting, can pose greater challenges in model selection and estimation.
- **Computational Efficiency:** ARIMA models, being simpler in structure, often require less computational resources for estimation and forecasting, making them more efficient, especially for large datasets or real-time applications.

Assessment 2

Problem statement

There are 15 participants who wore accelerator meter to collect data on various actions. The accelerator meter data contains x acceleration, y acceleration, z acceleration, and label. Your task is to build a model that can classify an action based on the input accelerator meter data (x acceleration, y acceleration, z acceleration)

Objective

- To conduct an EDA for understanding the patterns in the data
- To develop a classification model to predict the target variables
- To use right error metrics, cross-validation and hyperparemeter tuning.

Exploratory Data Analysis

There are a total of 15 input files each representing a different participant

Section 15 - Checking the Shape of the data

Data_ID	Shape_of_Participant_Data
1	(162501, 5)
2	(138001, 5)
3	(102341, 5)
4	(122201, 5)
5	(160001, 5)
6	(140901, 5)
7	(163001, 5)
8	(138001, 5)
9	(166741, 5)
10	(126801, 5)
11	(104451, 5)
12	(114702, 5)
13	(67651, 5)
14	(116101, 5)
15	(103501, 5)

- Each of the files has 5 columns
- The number of records are slightly differing
- The 5 columns are - sequential_number, x_accelaration, y_accelaration, z_accelaration, and label.

- For further analysis, we have merged all the 15 separate files together

	x_acceleration	y_acceleration	z_acceleration	label
1760678	1927	2404	2081	3
451852	2066	2333	2034	4
25626	1961	2382	2097	1
1314732	2016	2533	2035	1
1458681	2004	2350	2016	1
...	...	...	...	...
1903887	2113	2731	2036	5
778481	2251	2411	1974	5
1833281	2102	2541	2023	1
467300	2218	2493	2061	4
542476	1878	2351	2002	1

  - The total number of records is around 2 million (20 lakhs).

```
all_data.shape
```

```
(1926896, 4)
```

  - The number of columns excluding sequential_number is 4 (which is an irrelevant index column)

Section 16 - Implementing Stratified Sampling

Stratified sampling is a powerful technique used in statistics and research to improve the representativeness and accuracy of samples drawn from a population. It achieves this by dividing the population into smaller subgroups, called strata, based on shared characteristics like age, gender, income, education level, or any other relevant factor. Then, a proportionate or disproportionate number of individuals are randomly selected from each stratum to form the final sample.

Dividing into Strata:

- The population is segmented into distinct subgroups based on relevant characteristics.

- These subgroups, called strata, should be mutually exclusive (each member belongs to one and only one stratum) and collectively exhaustive (cover the entire population).

Benefits of Stratified Sampling:

- **Reduced Sampling Bias:** By ensuring each stratum is represented in the sample, stratified sampling helps reduce bias and increase the representativeness of the conclusions drawn from the sample.
- **Improved Precision:** Stratified samples often have lower sampling error, leading to more precise estimates of population parameters.
- **Subgroup Analysis:** It allows for analyzing specific subpopulations within the sample, providing valuable insights into their unique characteristics and behaviours.

Since the dataset has around 2 million records, there are limitations with respect to infrastructure for training such a big population (like RAM). Therefore, we are taking a sample of approx 400,000 records for conducting this experiment using stratified sampling. The stratified sampling ensures that all the classes will have a good number of participants in the sample from the population.

We can easily train this model on all the records by removing this piece of code which can also bring more accurate analysis as a part of further enhancement.

sklearn.model_selection.StratifiedShuffleSplit

We are using the StratifiedShuffleSplit package from Python to implement this technique

```
split = StratifiedShuffleSplit(n_splits=10, test_size=0.50, random_state=42)
for incl_index, excl_index in split.split(all_data, all_data['label']):
    strat_sample_set = all_data.loc[incl_index]
    strat_removal_set = all_data.loc[excl_index]
all_data = strat_sample_set
```

	x_acceleration	y_acceleration	z_acceleration	label
422878	2021	2383	2030	1
143267	1910	2381	1987	7
91361	1893	2388	2002	7
1626205	1798	2330	1985	7
1911878	2026	2532	2027	7
...	...	...	...	...
1252756	2073	2541	2048	7
628205	1923	2349	1937	7
372344	1945	2324	1945	4
828114	2152	2331	1811	1
993283	1905	2399	2033	1

385379 rows × 4 columns

Section 17 - Analysing the target variable

First of all, we need to analyse, if is there any class label which does not correspond to any target class.

Provided that, these are the classes for each label-
1 - Working at Computer
2 - Standing Up, Walking and going updown stairs
3- Standing
4 - Walking
5 - Going up down stairs
6 - Walking and Talking with someone
7- Talking while standing

	label
1	121733
7	118712
4	71413
3	43347
5	10300
2	9576
6	9554
0	744

There is a class 0 which is not mapped to any activity. Hence, we are removing those records. Since, the proportion of the class is very low, we don't expect any information loss & since it's a part of the target, removing is better than imputing to avoid biases in further analysis.

Section 18 - Checking if there are any columns with negative values

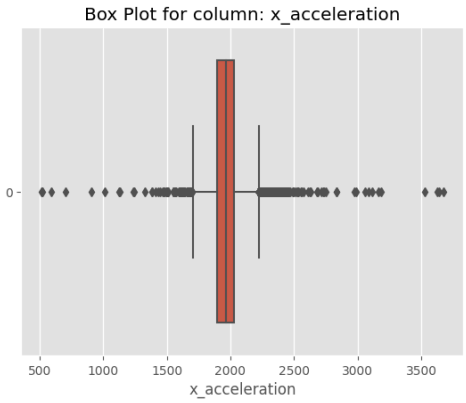
```
Is there any negative value in column: x_acceleration - False
Is there any negative value in column: y_acceleration - False
Is there any negative value in column: z_acceleration - False
```

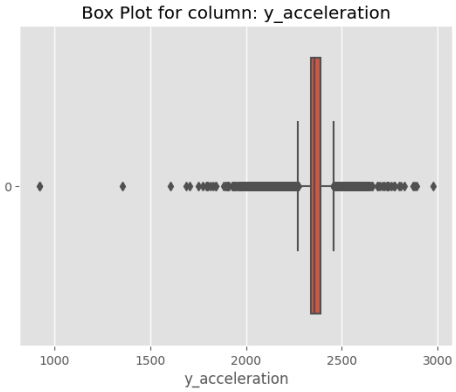
Section 19 - Checking if there are any missing values

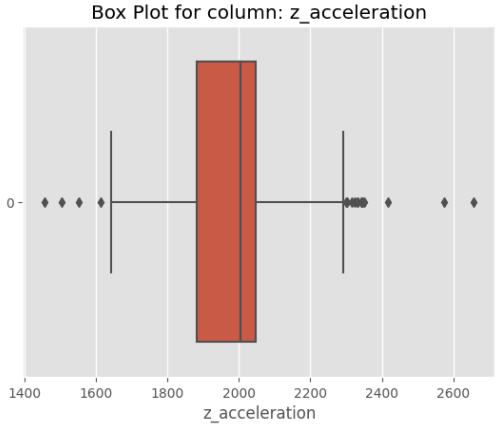
```
x_acceleration    0
y_acceleration    0
z_acceleration    0
label             0
dtype: int64
```

Section 20 - Checking and treatment of Outliers

To detect the outliers in each of the columns, the data is filtered through different labels and outliers detection technique is applied.







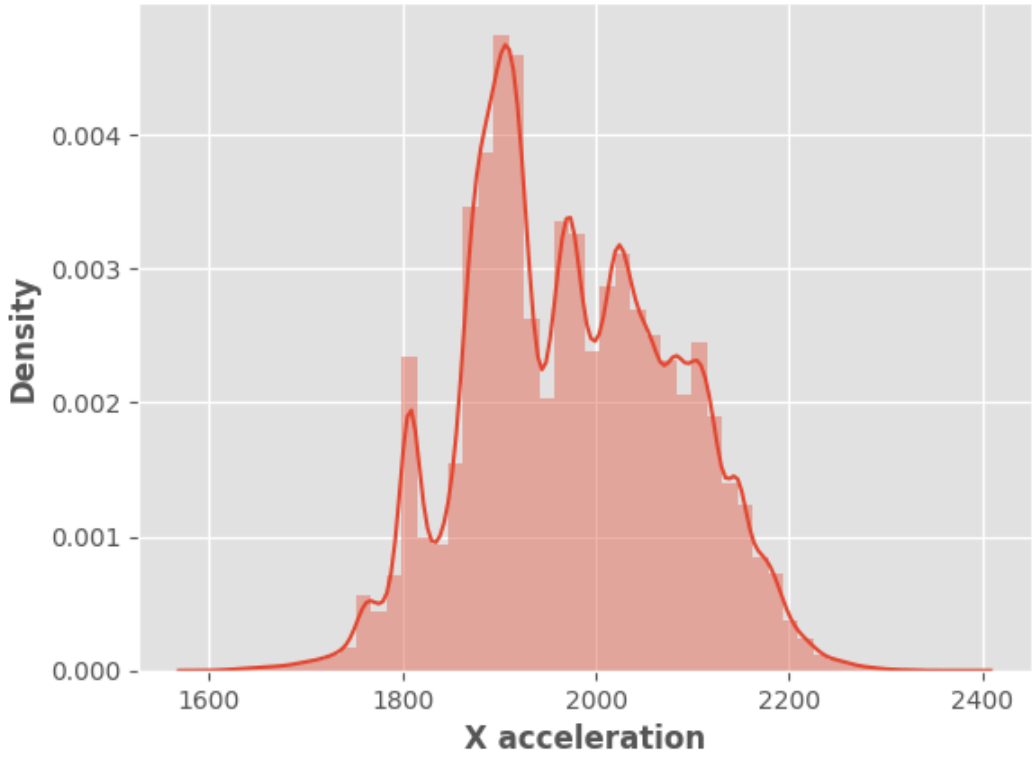
Outliers frequency			
	x_accelaration	y_accelaratio n	z_accelaration
Class 1	1069	36480	18
Class 2	22	2213	39
Class 3	25	7117	1326
Class 4	182	795	2196
Class 5	11	67	241
Class 6	0	33	14
Class 7	25	28202	817

Section 21 - Conducting Univariate analysis (Feature columns)

Analysing x_accelaration

	count	mean	std	min	25%	50%	75%	max
x_acceleration	303493.000000	1974.039000	106.949000	1594.000000	1895.000000	1968.000000	2055.000000	2383.000000

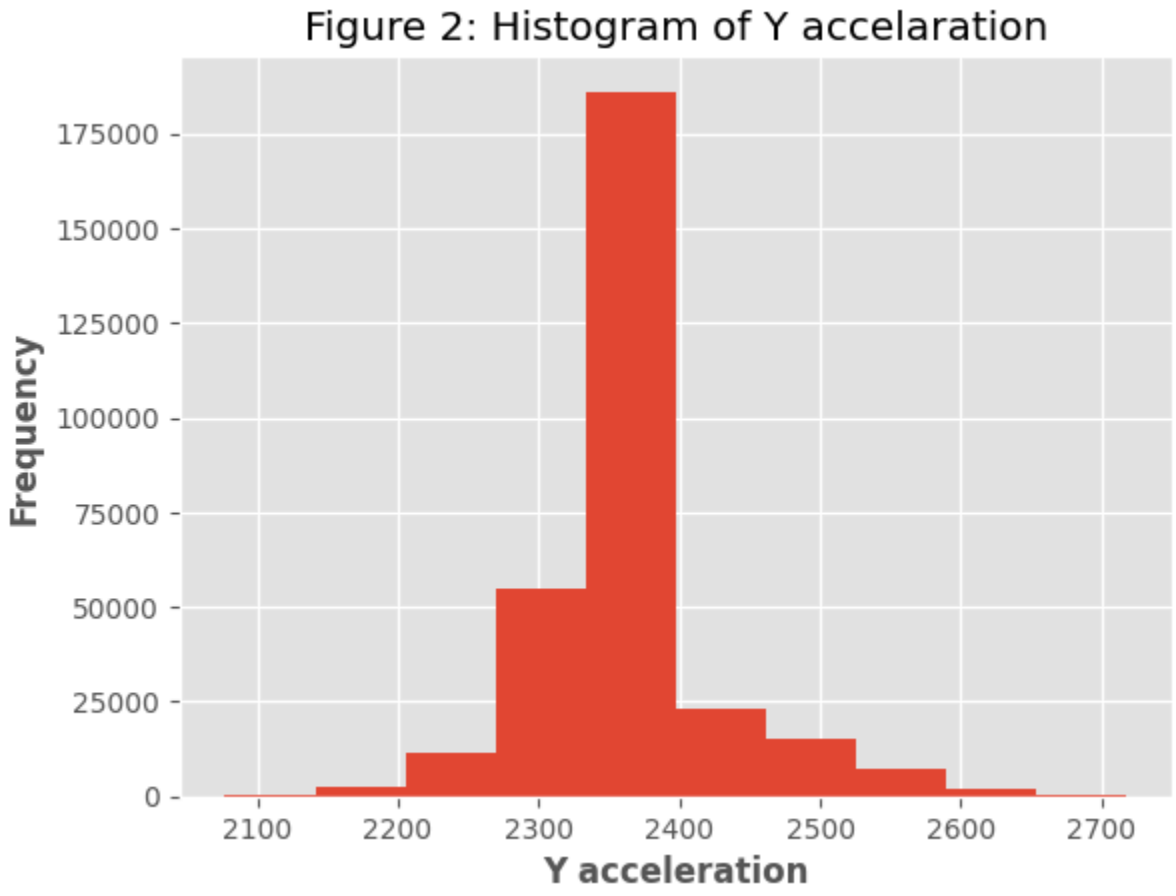
Figure 1: Histogram with kernel density estimate for X acceleration



- The column x_acceleration is explored by plotting a histogram and the kernel density estimate (KDE) is overlayed using the distplot function in seaborn.
-
- From the plot it can be observed that it is right skewed and most of the data falls to the right, or positive side, of the graph's peak. Moreover from the summary statistics, it can be inferred that the mean is greater than the median. Thus making it rightly skewed.

Analysing y_accelaration

	count	mean	std	min	25%	50%	75%	max
y_acceleration	303493.000000	2364.161203	65.617815	2077.000000	2337.000000	2358.000000	2385.000000	2718.000000

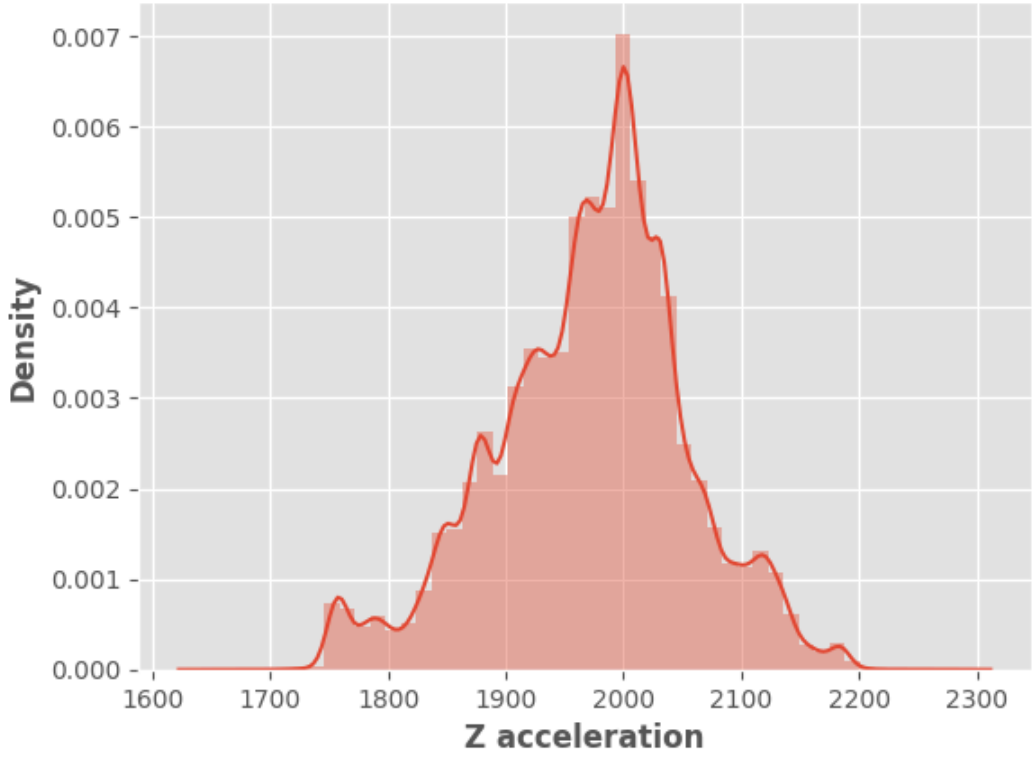


- The column y_acceleration is explored by plotting a histogram using the hist function.
- From the histogram it can be observed that it is slightly right skewed and most of the data falls around 2300 to 2400 which is the graph's peak. Moreover from the summary statistics, it can be inferred that the mean is slightly greater than the median. Thus making it rightly skewed.

Analysing the z_acceleration

	count	mean	std	min	25%	50%	75%	max
z_acceleration	303493.000000	1972.544583	82.011678	1642.000000	1922.000000	1980.000000	2024.000000	2292.000000

Figure 3: Histogram with kernel density estimate for Z acceleration

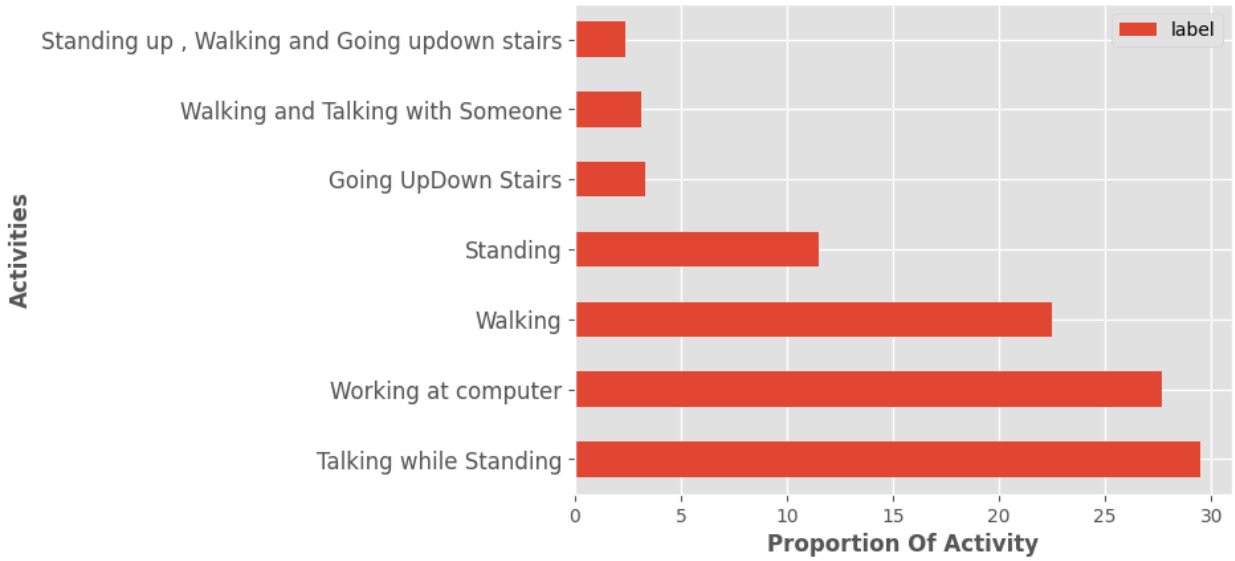


- The column z_acceleration is explored by plotting a histogram and overlaying the kernel density estimate (KDE) using the distplotfunction in seaborn.
- From the plot it can be observed that it is left skewed as the tail is longer on the left and most of the data falls to the left, or negative side, of the graph's peak. From the summary statistics, it can be inferred that the mean is to the left of the median (mean lesser than the median). Thus making it left-skewed.

Section 22 - Conducting Univariate analysis (Target column)

	label
2	2.400000
6	3.100000
5	3.300000
3	11.500000
4	22.500000
1	27.700000
7	29.500000

Figure 4: Bar Chart of Activities

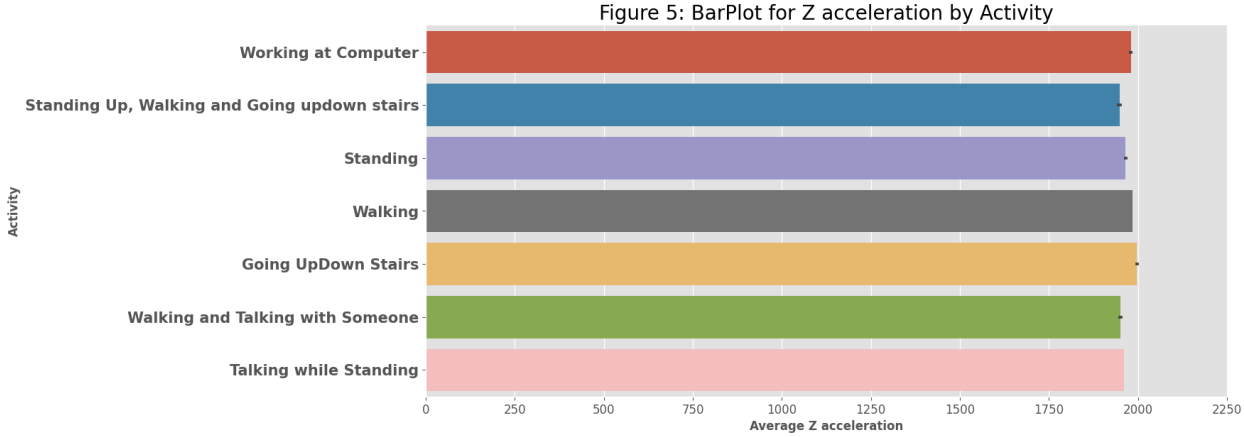


- The column labels were explored by plotting a horizontal bar chart using the plot(kind = 'barh') function.
- From the plot it can be observed that over one-fourth of the proportion of activities of the participants was Working at Computer and Talking while standing. The remaining proportion was occupied by the other 5 activities. Activity - Standing up, Walking and Going up-down stairs was the least activity and Talking while Standing was the most activity performed by the participants. This signifies that there is a class imbalance in the dataset.

Section 23 - Conducting Bivariate Analysis (Feature + Target)

Analyzing the relationship between z_acceleration and target column

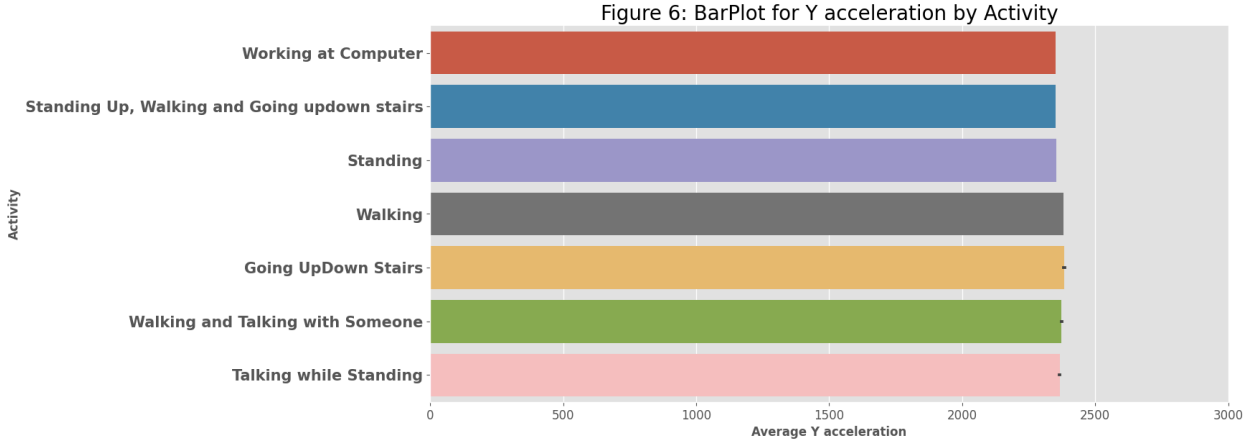
Figure 5: BarPlot for Z acceleration by Activity



- From the above barplot it can be inferred that the Average Z acceleration value is not the same for each activity.
- Going UpDown Stairs Activity has the highest average value and Standing up, walking and going updown stairs has the lowest mean value among the other activities.
- This signifies that average Z acceleration values are different for different activities.

Analysing the relationship between y_accelertation and the target column

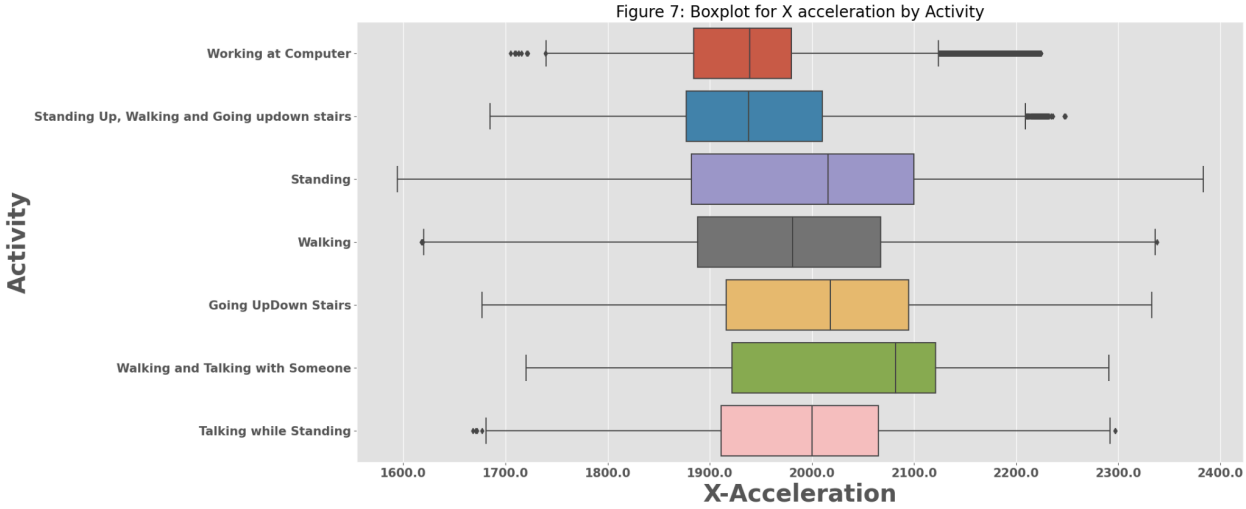
Figure 6: BarPlot for Y acceleration by Activity



- From the above barplot it can be seen that average y_acceleration looks similar for each activity. But the mean value from the summary statistics suggests that Going UpDown Stairs activity has the highest mean value and `Working at the computer has the lowest mean value. Other activities also have dissimilar values. Thus inferred that the Average Y acceleration value is not the same for each activity.
- This signifies that Y acceleration values are different for different activities.

Analysing the relationship between x_accelaration and target column

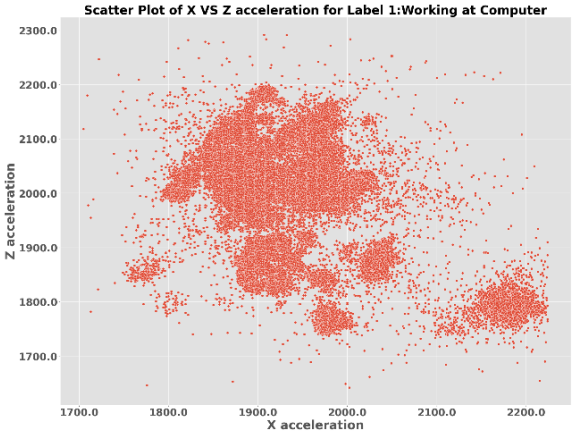
Figure 7: Boxplot for X acceleration by Activity

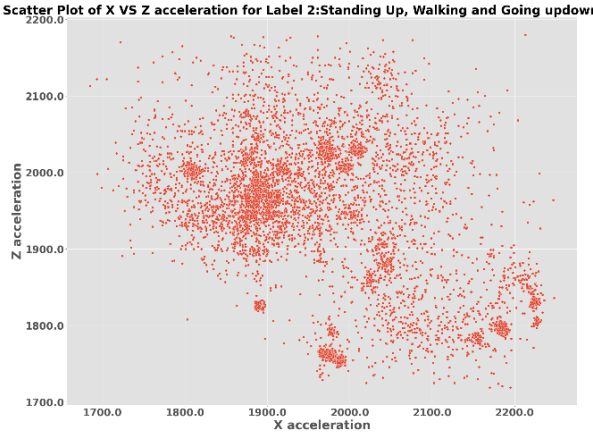


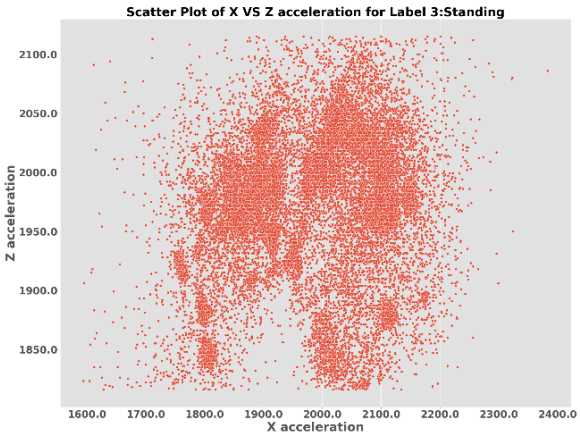
- From the above boxplots and summary statistics of x_acceleration across activities, it can be seen that apart from `Working at Computer activity, all other activities do not follow a normal distribution.
- Thus the values for the Working at Computer are found to be symmetric with outliers at the positive side of the distribution.

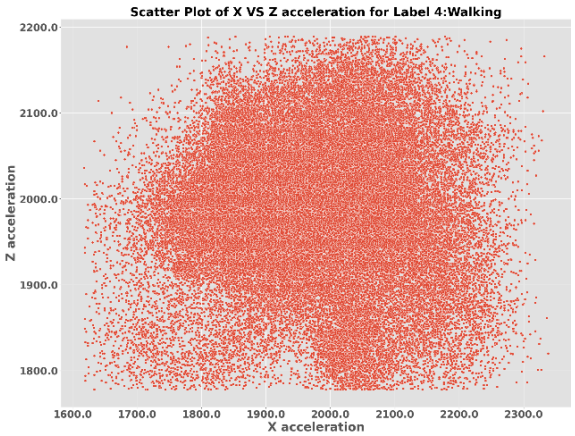
Section 24 - Conducting Bivariate Analysis (Feature + Feature)

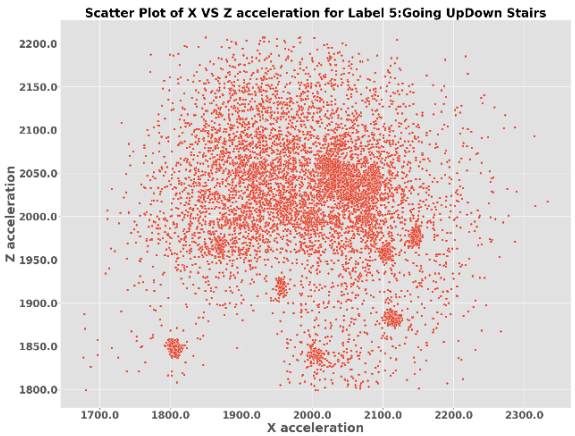
Analysing the relationship between x_accelation and z_accelartion

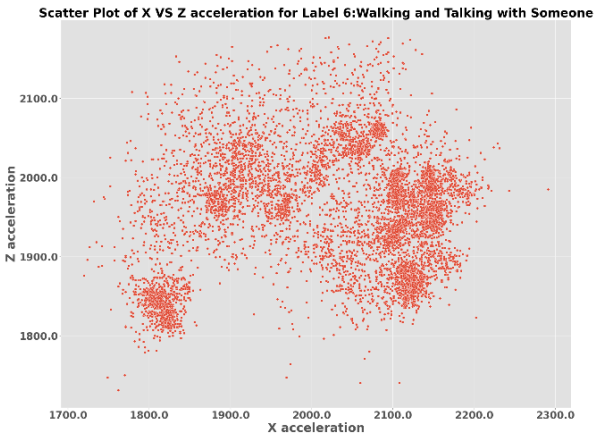

Scatter Plot of X VS Z acceleration for Label 1:Working at Computer
Z acceleration
X acceleration
2300.0
2200.0
2100.0
2000.0
1900.0
1800.0
1700.0
1700.0
1800.0
1900.0
2000.0
2100.0
2200.0

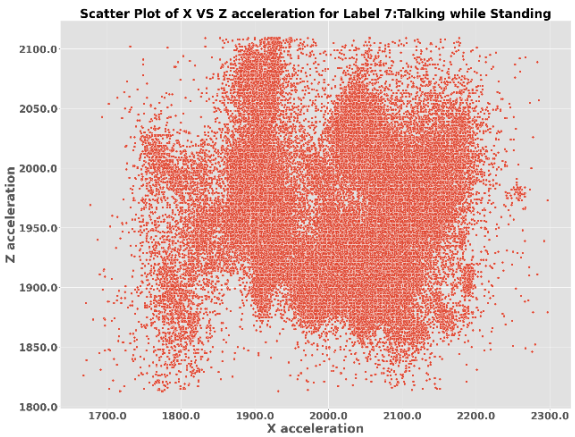

Scatter Plot of X VS Z acceleration for Label 2:Standing Up, Walking and Going updown stairs
Z acceleration
X acceleration
2200.0
2100.0
2000.0
1900.0
1800.0
1700.0
1700.0
1800.0
1900.0
2000.0
2100.0
2200.0


Scatter Plot of X VS Z acceleration for Label 3:Standing
Z acceleration
X acceleration
2100.0
2050.0
2000.0
1950.0
1900.0
1850.0
1600.0
1700.0
1800.0
1900.0
2000.0
2100.0
2200.0
2300.0
2400.0


Scatter Plot of X VS Z acceleration for Label 4:Walking
Z acceleration
X acceleration
2200.0
2100.0
2000.0
1900.0
1800.0
1600.0
1700.0
1800.0
1900.0
2000.0
2100.0
2200.0
2300.0


Scatter Plot of X VS Z acceleration for Label 5:Going UpDown Stairs
Z acceleration
X acceleration
2200.0
2150.0
2100.0
2050.0
2000.0
1950.0
1900.0
1850.0
1800.0
1700.0
1800.0
1900.0
2000.0
2100.0
2200.0
2300.0


Scatter Plot of X VS Z acceleration for Label 6:Walking and Talking with Someone
Z acceleration
X acceleration
2100.0
2000.0
1900.0
1800.0
1700.0
1800.0
1900.0
2000.0
2100.0
2200.0
2300.0

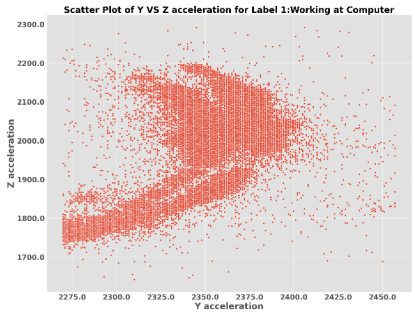

Scatter Plot of X VS Z acceleration for Label 7:Talking while Standing
Z acceleration
X acceleration
2100.0
2050.0
2000.0
1950.0
1900.0
1850.0
1800.0
1700.0
1800.0
1900.0
2000.0
2100.0
2200.0
2300.0

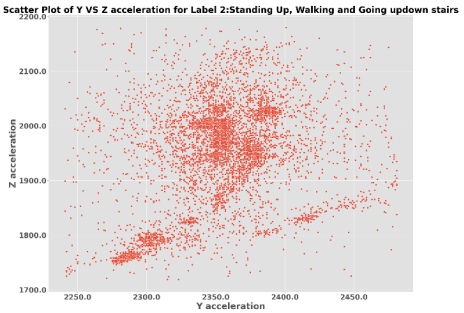
From the above scatterplots plotted X acceleration Vs Z acceleration for different activities, it can be concluded that there is no relationship between the X and Z accelerations for any of the activities. Thus X acceleration and Z acceleration values do not have much impact in classifying the type of activity.

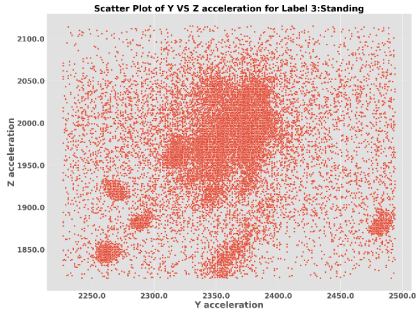
Analysing the relationship between x_acceleration and y_acceleration

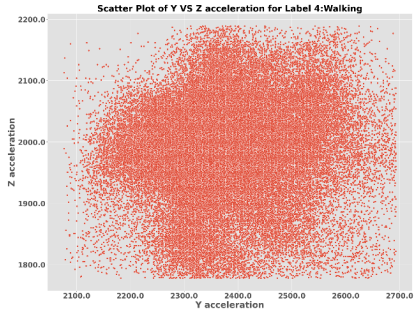
- From the above scatterplots plotted X acceleration Vs Y acceleration for different activities, it can be concluded that there is _no clear relationship between the X and Y accelerations for any of the activities.
- Thus X acceleration and Y acceleration values do not have much impact in classifying the type of activity.

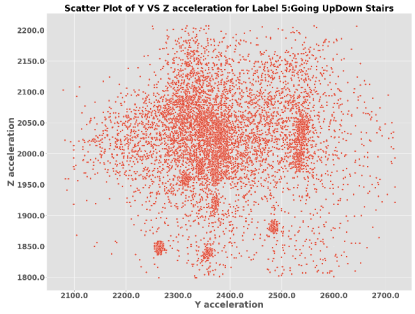
Analysing the relationship between y_acceleration vs z_acceleration

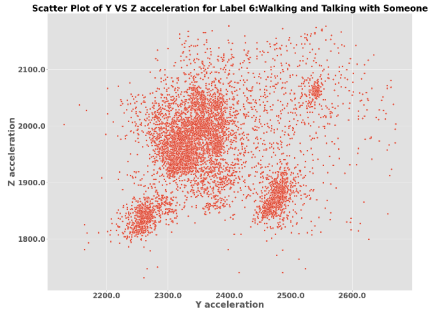


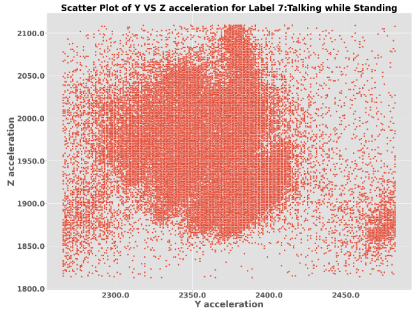












- From the above scatterplots plotted Y acceleration Vs Z acceleration for different activities, it can be concluded that there is no clear relationship between the Y and Z accelerations for any of the activities. Thus Y acceleration and Z acceleration values do not have much impact in classifying the type of activity.

Section 25 - Predictive modelling: Model 1 - K Nearest Neighbour

Feature selection

Feature selection is performed by the Simple Hill climbing technique.

Simple Hill Climbing is a straightforward optimization technique aimed at identifying the best subset of features for a machine learning model. It works by starting with an initial set of features and then iteratively making small changes to improve the performance of the model, based on a pre-defined evaluation metric.

Here's how it works:

1. **Initial State:** Begin with a randomly chosen subset of features or an existing feature set you want to optimize.
2. **Neighbor Exploration:** Generate neighbouring states by adding, removing, or swapping individual features from the current set.
3. **Evaluation:** Evaluate the performance of the model on each neighbouring state using a designated metric, such as accuracy, precision, or recall.
4. **Selection**: If any neighbouring state performs better than the current state, replace the current state with the best-performing neighbour.
5. **Termination:** Repeat steps 2-4 until no further improvement is found or a maximum number of iterations is reached.

```
Score with 1 selected features: 0.39228758457077584
Score with 2 selected features: 0.6014519813724629
Score with 3 selected features: 0.7288133731658026
```

- From the simple hill climbing technique with default parameters for KNeighborsClassifier, the best accuracy score is obtained when all the three features i.e, x_acceleration, y_acceleration and z_acceleration are selected.
- Thus all three features are selected.

Train-Test Split

- Splitting the data into train and test partitions with a 70:30 ratio using stratification.
- As there is a class imbalance in the dataset stratify is set to target(stratify = target). This makes sure that the proportion of target values in the sample remains the same as the original data.

```
from sklearn.model_selection import train_test_split
D_train , D_test , t_train , t_test = train_test_split(data,target,
                                                       test_size=0.30,
                                                       random_state=999,
                                                       stratify = target)
```

Cross Validation & Hyperparameter Tuning

For cross-validation, the K-fold technique is used with K=5

Number of Neighbors (n_neighbors): To make the classifier less susceptible to "noise" in the data higher k values are used. Thus to make sure to get an optimal value of k the following k values are used for hyperparameter tuning:
[1, 5, 50, 100]

Power parameter for the Minkowski metric (p)- As the dataset has only 3 dimensional vectors trying for a higher value of might work better. Hence following p values are used for hyperparameter tuning.
p=1 (Manhattan Distance)
p=2 (Euclidean Distance)
p=5 (Minkowski Distance)

Weight function (weights): As the data is uncalibrated considering all points in each neighbourhood as weighted equally will not better option. Thus both uniform and distance weight functions are used for hyperparameter tuning.

```
Hyperparameter Tuning  22  - Number of neighbors: 100 , Metric: minkowski , p: 2 , Weights: distance
Fold CV Scores:
[fold 0] score: 0.74266
[fold 1] score: 0.74313
[fold 2] score: 0.74290
[fold 3] score: 0.74579
[fold 4] score: 0.74367
The Mean Cross Validation Score is: 0.7436324695803619
Hyperparameter Tuning  23  - Number of neighbors: 100 , Metric: minkowski , p: 5 , Weights: uniform
Fold CV Scores:
[fold 0] score: 0.74539
[fold 1] score: 0.74692
[fold 2] score: 0.74645
[fold 3] score: 0.74843
[fold 4] score: 0.74551
The Mean Cross Validation Score is: 0.7465414577890748
Hyperparameter Tuning  24  - Number of neighbors: 100 , Metric: minkowski , p: 5 , Weights: distance
Fold CV Scores:
[fold 0] score: 0.74165
[fold 1] score: 0.74306
[fold 2] score: 0.74318
[fold 3] score: 0.74528
[fold 4] score: 0.74313
The Mean Cross Validation Score is: 0.7432606086281155
The best score obtained is  0.7514462566781991
```

Among the various scores obtained the one with the highest mean cross-validation score is selected and the corresponding set of hyperparameter values is selected. From the hyperparameter tuning the following optimal set of parameters for KNN is obtained:

Number of Neighbors: 50
Power parameter for the Minkowski metric: 5
Weight function: uniform
The best accuracy score obtained for the above set of parameters for KNN is **0.75**

Section 26 - Predictive modelling: Model 1 - Decision Tree

For cross-validation, K-fold technique is used with K=5

Feature selection

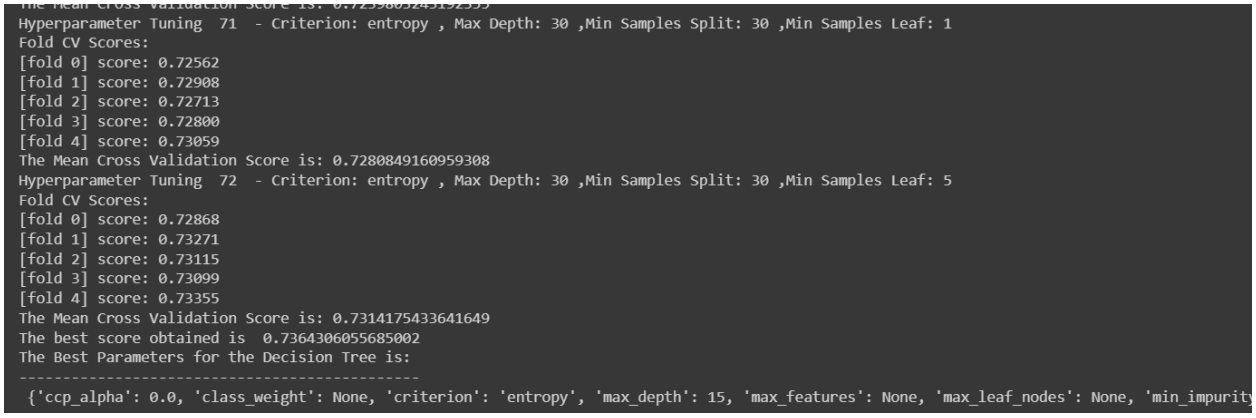
Similar to KNN, Here also we have used Simple hill-climbing algorithm and all of the 3 features are shortlisted for further statistical procedures

Train-Test Split

Similar to KNN, Splitting the data into train and test partitions with a 70:30 ratio using stratification.

Hyper Parameter Tuning

- **Impurity Measure (criterion)**- Decision tree nodes are split by using impurity. The following Impurity measurements are used for hyperparameter tuning. 'gini' and 'entropy'
- **Maximum Depth of tree (max_depth)**- The maximum depth is specified to avoid overfitting as higher depth will allow the model to learn relations very specific to a particular sample. The following values are used for max_depth in hyperparameter tuning - [5,10,15,20,25,30]
- **Minimum Samples for a node split (min_samples_split)-** The Minimum Samples Split is specified to control overfitting. Too high values lead to under-fitting, therefore the following values are tuned using cross-validation - [10,20,30]
- **Minimum samples for a terminal node (min_samples_leaf)-** The Minimum Samples leaf is specified to control overfitting.As the dataset has imbalanced class counts lower values are chosen -[1,5]



Among the various scores obtained, the one with the highest mean cross-validation score is selected and the corresponding set of hyperparameter values is selected. From the hyperparameter tuning the following optimal set of parameters for the Decision Tree is obtained:

- Impurity Measure entropy
- Maximum Depth of tree 15
- Minimum Samples for a node split 30
- Minimum samples for a terminal node 5
- The best accuracy score obtained for the above set of parameters for Decision Tree is 0.73

Section 27: Evaluation of KNN & DT models

Classification report of KNN

	precision	recall	f1-score	support
1	0.84	0.92	0.88	25250
2	0.56	0.09	0.16	2191
3	0.60	0.52	0.56	10464
4	0.72	0.76	0.74	20472
5	0.49	0.12	0.19	2994
6	0.60	0.25	0.35	2852
7	0.75	0.85	0.80	26825
accuracy			0.75	91048
macro avg	0.65	0.50	0.53	91048
weighted avg	0.73	0.75	0.73	91048

- From the classification report, it can be observed that neither precision or recall tells the whole story.
- For a few labels there is excellent precision with bad recall, or alternately, terrible precision with excellent recall. Hence F1-Score is the best metric as it combines precision and recall.
- The F1-Score is highest for Label 1 - Working at Computer and lowest for Label 2 - Standing Up, Walking and Going down stairs.
- The weighted average of the F1-score for the KNN model with optimal parameters is 0.73.

Classification report of Decision Tree

	precision	recall	f1-score	support
1	0.85	0.91	0.88	25250
2	0.42	0.09	0.14	2191
3	0.56	0.50	0.53	10464
4	0.69	0.75	0.72	20472
5	0.39	0.13	0.19	2994
6	0.54	0.20	0.30	2852
7	0.74	0.83	0.79	26825
accuracy			0.74	91048
macro avg	0.60	0.49	0.51	91048
weighted avg	0.71	0.74	0.72	91048

- From the classification report, it can be observed that neither precision nor recall tells a clear story.
- For a few labels there is excellent precision with bad recall, or alternately, terrible precision with excellent recall. Hence F1-Score is the best metric as it combines precision and recall.
- The F1-Score is highest for Label 1 - Working at Computer and lowest for Label 2 - Standing Up, Walking and Going updown stairs.
- The weighted average of the F1-score for the Decision Tree model with optimal parameters is 0.72.

Section 28: Final Inference

- By comparing the model's accuracy score, confusion matrix and classification report, KNN and DT are almost par with each other. We can go forward with 2 models or else if it is needed to shortlist one, then KNN would be better since its F1 and accuracy are slightly better than DT. Also, the difference is expected to go up when new data points are added.
- Also, as the dataset has only continuous variables the decision tree will not be a better option as the decision tree loses information when it splits variables in different nodes.
- Thus KNearest Neighbor classifier is recommended over the Decision Tree.
- In addition to the KNN model built, the performance can be further enhanced by performing stratified sampling or rebalancing to avoid imbalance in the dataset. In addition, the scaling of the descriptive features would lead to improved performance.
- Apart from KNN and decision trees more complex Machine Learning Algorithms such as Random Forest Classifier and Neural Networks would give the best performance and predictive power.