FACULTY OF COMPUTING
UTM Johor Bahru

MCSD 1143: SUPPLY CHAIN ANALYTICS

PROJECT

Lecturer: Dr. Noorfa Haszlinna bte Mustaffa

Date of Submission: 19th May 2024

No.	Group Member	Matrics No
1	Syeinrita Devi A/P Anbealagan	MCS221022
2	Fan Chin Wei	MCS221024
3	Zhang Qi Wei	MCS221013
4	Muhamad Afiq Asyraf Bin Md Ishak	MCS221026

TABLE OF CONTENTS

TABLE OF FIGURES

Introduction

Efficient supply chain management relies heavily on accurate demand forecasting, allowing businesses to maintain optimal inventory levels, reduce waste, and ensure timely production and distribution. In industries where consumer demand is highly variable, like the airline industry, the ability to forecast accurately becomes even more crucial. Airlines must manage resources such as aircraft, fuel, and staff, all while adapting to seasonal fluctuations and broader economic trends. Successful demand forecasting can lead to smoother operations, reduced costs, and an overall better customer experience.

In this project, we focus on forecasting demand by analyzing a time series dataset and applying various statistical and machine learning methods. Our goal is to evaluate several approaches to determine the most effective forecasting method for the chosen data.

Problem Background

In the airline industry, the supply chain management (SCM) process encompasses a wide range of activities—from managing ticket bookings to scheduling flights and handling passenger services. Accurate demand forecasting is a critical component, as it directly impacts the airline's ability to manage resources efficiently and ensure high levels of customer satisfaction. The need for precise demand forecasting stems from the fact that misalignment between expected demand and actual needs can lead to significant operational disruptions and financial losses.

Given the importance of precise demand forecasting in the airline industry, this project aims to evaluate several time series forecasting methods to determine which is best suited for the "Air Passengers" dataset. An accurate forecast allows airlines to plan their operations more effectively, ensuring that resources are appropriately allocated, flights are scheduled efficiently, and passenger services are optimized. Misalignment in forecasts can lead to overbooking, underutilization of flights, and other operational challenges, ultimately affecting customer satisfaction and the airline's bottom line. Through this project, we seek to identify the most effective forecasting method, contributing to the optimization of SCM in the airline industry.

Project Objective

Given the critical role of demand forecasting in the airline industry, this project aims to evaluate a variety of forecasting methods to determine which is most suitable for the "Air Passengers" dataset. By accurately predicting demand, airlines can better manage ticket bookings, flight scheduling, and passenger services, thereby enhancing operational efficiency and customer satisfaction. The forecasting methods under consideration include:

- Moving Average: A simple smoothing technique to identify underlying trends.
- Exponential Smoothing: Includes variations like Simple Exponential Smoothing, Holt's linear trend, and Winter's model, each designed to capture trends and seasonality.
- Box-Jenkins Method: Focuses on Autoregressive Integrated Moving Average (ARIMA) and Seasonal ARIMA (SARIMA), well-suited for complex time series data.

Project Approach

To assess the effectiveness of each forecasting method, we will calculate common error metrics, such as Mean Squared Error (MSE), Mean Absolute Deviation (MAD), and Mean Absolute Percentage Error (MAPE). These metrics will help quantify the accuracy of the forecasts and identify the most reliable method.

Data Source

The dataset for this project, known as the "Air Passengers", is sourced from Kaggle, a popular platform for data science datasets. Kaggle is widely recognized for providing a vast array of datasets across various domains, enabling data scientists and researchers to practise their skills and explore real-world data. The platform also serves as a collaborative community, allowing users to share insights, code, and methodologies for addressing complex data-driven problems. Then, this dataset is extracted in CSV format for easy access and analysis using Python-based tools and libraries. This enables a thorough investigation of supply chain analytics and time series forecasting techniques. Link of data is https://www.kaggle.com/datasets/rakannimer/air-passengers.

Data Background

The "Air Passengers" dataset contains monthly totals of international airline passengers from January 1949 to December 1960. This historical data set has become a classic resource for time series analysis, thanks to its clearly identifiable seasonal patterns and consistent upward trend. By examining this dataset, we can uncover critical insights into the behavior of international air travel over more than a decade, providing a robust foundation for developing demand forecasting models.

The data contains the following components:

- Month: This column represents the month and year of each observation, in the format YYYY-MM-DD. The dataset spans from January 1949 to December 1960, resulting in 144 records. Although the day part of the date is not used for analysis, the full date format maintains consistency and compatibility with time series operations in various programming environments. The data type for this column is datetime, allowing for accurate time-based operations, such as calculating differences between dates and extracting specific components like month or year.
- #Passengers: This column indicates the total count of international airline passengers for each corresponding month. The values range from a minimum of 104 passengers to a maximum of 622 passengers over the entire dataset. The data type for this column is integer, as it represents discrete counts of passengers.

The dataset is considered demand data because it captures the number of passengers (demand) for flights each month reflecting demand for air travel. Demand data is crucial for understanding how many resources (such as flights and seats) are required to meet the needs of customers. By analyzing these passenger counts, airlines can predict future demand, manage capacity, optimize scheduling, and make informed business decisions to cater to fluctuating travel patterns and preferences.

Project Deliverables

The key deliverables for this project are:

- Data Analysis: An in-depth examination of the Air Passengers dataset to understand its trends, seasonality, and underlying patterns.
- Model Development: Implementation of the various forecasting methods outlined above.
- Forecasting: Generation of forecasts for 4 to 6 periods ahead using different forecasting techniques.
- Performance Evaluation: Assessment of each method's accuracy using error metrics like MSE, MAD, and MAPE.
- Recommendations: Suggestions for the optimal method for demand forecasting, based on the analysis and results.

Through this project, we aim to provide insights into the most effective methods for demand forecasting in the airline industry and offer practical recommendations for the time series data. As a whole, by predicting future demands accurately, airlines can adjust their capacities, manage staffing levels, and optimize fuel consumption, thereby enhancing operational efficiency and customer service.

Tools Used

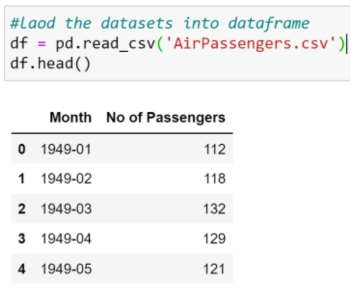
For this project, Python is used as the primary programming language for data analysis and forecasting. Python's extensive ecosystem of libraries and packages makes it a versatile tool for time series analysis and Microsoft Excel is used for data analysis and modelling due to its user-friendly interface and wide range of functions. We will use pandas for data manipulation and analysis, allowing for efficient data organization and cleaning. NumPy will assist with numerical operations and array processing, while matplotlib is used for visualization, providing capabilities for plotting data trends and patterns. Additionally, the statsmodels package is crucial for implementing statistical models, including those required for time series forecasting like ARIMA and exponential smoothing.

Microsoft Word is employed for documenting the project's findings and preparing the final report. Its robust formatting tools and support for embedded charts and graphics make it an ideal platform for presenting results in a clear and organized manner. For the visual presentation of the report, Canva will be used to create engaging slides and video presentations. By combining Python and Microsoft Excel for data analysis, Microsoft Word for reporting and Canva for video presentation, we can ensure a comprehensive approach to both the technical and presentation aspects of the project.

Data Exploration

In this project, we performed an initial data exploration to understand the structure and characteristics of the "Air Passengers" dataset, which contains monthly totals of international airline passengers from January 1949 to December 1960.

Data Import



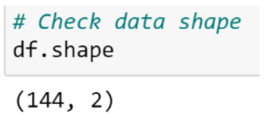
```
#laod the datasets into dataframe
df = pd.read_csv('AirPassengers.csv')
df.head()
```

```
     Month  No of Passengers
0  1949-01               112
1  1949-02               118
2  1949-03               132
3  1949-04               129
4  1949-05               121
```

Figure 1: Data Import

The data was imported into a pandas DataFrame for analysis in Python.

Data Shape



```
# Check data shape
df.shape
```

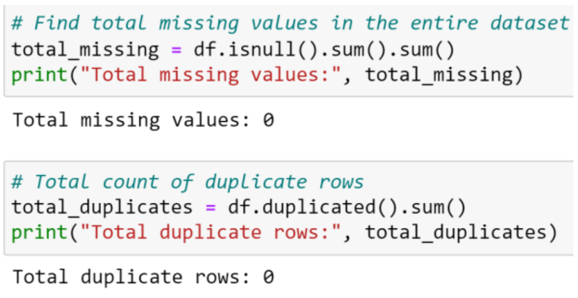
```
(144, 2)
```

Figure 2: Data Shape

The shape of the dataset is (144, 2), indicating there are 144 records (rows) and 2 attributes (columns). The columns represent:

- Month: The date in YYYY-MM-DD format, with only the month and year components used for analysis.
- No. of Passengers: The total count of international airline passengers for the corresponding month.

Missing & Duplicate Values



```
# Find total missing values in the entire dataset
total_missing = df.isnull().sum().sum()
print("Total missing values:", total_missing)
```

```
Total missing values: 0
```

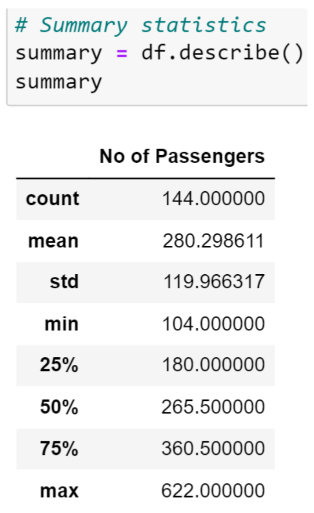
```
# Total count of duplicate rows
total_duplicates = df.duplicated().sum()
print("Total duplicate rows:", total_duplicates)
```

```
Total duplicate rows: 0
```

Figure 3: Missing & Duplicate Values

The dataset has no missing values, indicating that each month in the given time frame has a corresponding passenger count, resulting in a complete and consistent time series. The dataset has no duplicate rows, showing that each record is unique. This absence of duplicates suggests there are no repeated months or passenger counts, preserving the integrity of the data.

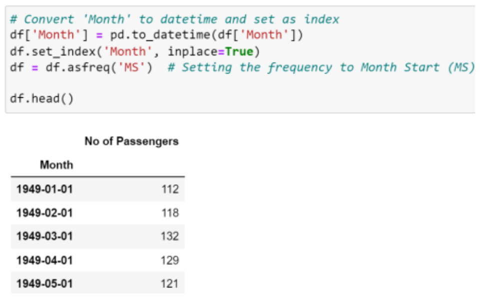
Data Summary



```
# Summary statistics
summary = df.describe()
summary
```

	No of Passengers
count	144.000000
mean	280.298611
std	119.966317
min	104.000000
25%	180.000000
50%	265.500000
75%	360.500000
max	622.000000

The summary of the data reveals that the mean passenger count is 280.298611, indicating the average number of monthly passengers over the 12-year period. The standard deviation is relatively high, suggesting significant variation in passenger counts across the dataset. The data has a wide range, from a minimum of 104 to a maximum of 622, indicating a steady upward trend with some seasonal fluctuations.



```
# Convert 'Month' to datetime and set as index
df['Month'] = pd.to_datetime(df['Month'])
df.set_index('Month', inplace=True)
df = df.asfreq('MS')  # Setting the frequency to Month Start (MS)

df.head()
```

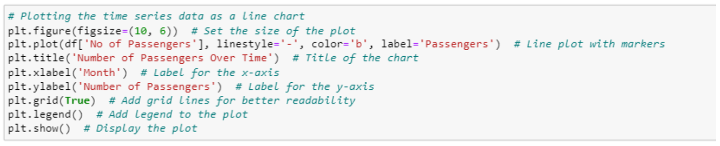
Month	No of Passengers
1949-01-01	112
1949-02-01	118
1949-03-01	132
1949-04-01	129
1949-05-01	121

Figure 4: Data Type

.

The 'Month' column is converted from a string to a datetime format and then sets it as the DataFrame index. The frequency of the DataFrame is explicitly set to "Month Start," ensuring that the data points align with the beginning of each month. This step is crucial for maintaining consistency and enabling accurate time-based operations in the dataset.

Time Series Plot



```
# Plotting the time series data as a line chart
plt.figure(figsize=(10, 6))  # Set the size of the plot
plt.plot(df['No of Passengers'], linestyle='-', color='b', label='Passengers')  # Line plot with markers
plt.title('Number of Passengers Over Time')  # Title of the chart
plt.xlabel('Month')  # Label for the x-axis
plt.ylabel('Number of Passengers')  # Label for the y-axis
plt.grid(True)  # Add grid lines for better readability
plt.legend()  # Add legend to the plot
plt.show()  # Display the plot
```

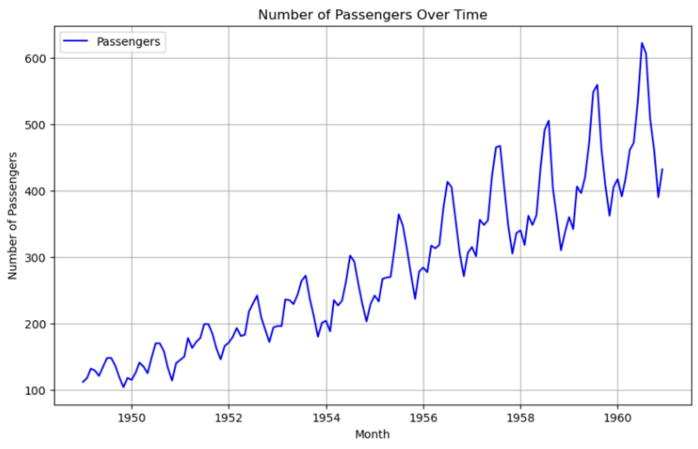


Figure 5: Air Passengers Count by Time Plot

A line graph was plotted to visually represent the monthly no of passengers over time. The x-axis represents the date, and the y-axis represents the count of passengers. The most apparent characteristic is the steady upward trend in the number of passengers over the 12-year period. This trend suggests a general growth in international air travel during this timeframe, indicating a broader increase in demand for airline services. Beyond the general upward trend, there's a clear seasonal pattern. Passenger counts typically peak in the summer months (June to August) and down during the winter months (December to February). This pattern aligns with common travel behavior, with more people traveling during summer vacations and fewer during winter. The line chart also shows consistent year-to-year growth. For example, the peaks in summer months tend to be higher each successive year, suggesting that the airline industry was experiencing significant expansion during the 1950s and early 1960s. This analysis provides a foundational understanding of the data, highlighting the importance of considering both trends and seasonality when forecasting future demand.

Time Series Decomposition

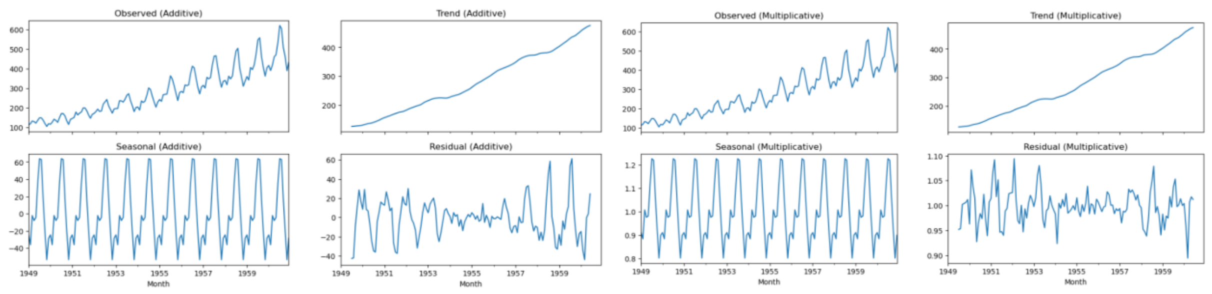


Figure 6: Decomposition

Time series decomposition helps us understand the underlying components of a time series: trend, seasonality, and residuals. In this case, the decomposition has been done using both additive and multiplicative models to analyze the "Air Passengers" dataset.

Additive Decomposition

- Trend: The upward trend is clear and consistent across the entire time series, indicating a steady increase in the number of passengers over time.
- Seasonality: The seasonal pattern in the additive model ranges from -40 to 60, with peaks and troughs occurring regularly each year. This suggests that the seasonal effect has a relatively constant amplitude over time.
- Residuals: The residuals (the part of the time series that can't be explained by the trend or seasonality) also range from -40 to 60, indicating a relatively wide dispersion. This can imply some noise or variability in the data that isn't captured by the trend and seasonal components.

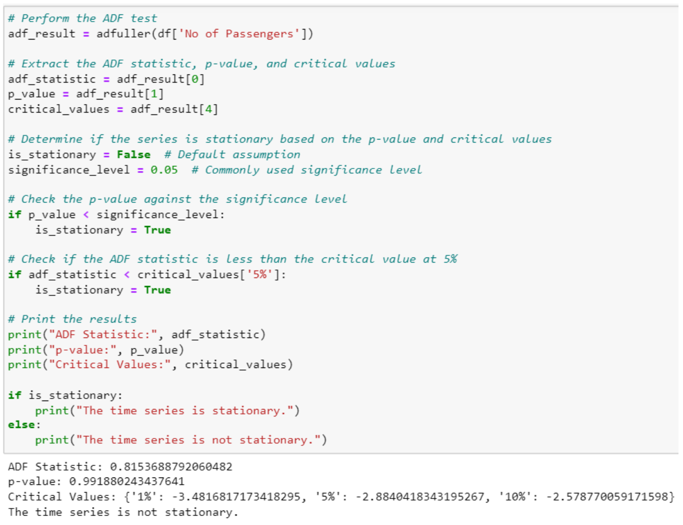
Multiplicative Decomposition

- Trend: The trend in the multiplicative model aligns with the additive decomposition, showing a steady upward growth.
- Seasonality: In the multiplicative model, the seasonal pattern fluctuates between 0.8 and 1.2. This indicates that the seasonal effect varies proportionally with the trend, suggesting that the amplitude of the seasonal fluctuations grows as the overall level of the series increases.
- Residuals: The residuals in the multiplicative model range from 0.8 to 1.1. These residuals appear to be less variable compared to the additive model, suggesting that the

multiplicative approach might better capture the proportional nature of the seasonal pattern.

Both additive and multiplicative decompositions confirm an upward trend in the time series. The seasonal patterns are evident in both models, with consistent peaks and troughs corresponding to seasonal travel behavior. The range of residuals suggests that the multiplicative model might better account for proportional changes in seasonality and residuals compared to the additive model. Given this decomposition analysis, a multiplicative model might be more suitable for this dataset, as it seems to capture the proportional relationship between the trend and seasonality. This could be useful in forecasting, where it's important to understand not only the underlying trend but also how the seasonal effects might change over time.

Stationarity Test



```python
# Perform the ADF test
adf_result = adfuller(df['No of Passengers'])

# Extract the ADF statistic, p-value, and critical values
adf_statistic = adf_result[0]
p_value = adf_result[1]
critical_values = adf_result[4]

# Determine if the series is stationary based on the p-value and critical values
is_stationary = False  # Default assumption
significance_level = 0.05  # Commonly used significance level

# Check the p-value against the significance level
if p_value < significance_level:
    is_stationary = True

# Check if the ADF statistic is less than the critical value at 5%
if adf_statistic < critical_values['5%']:
    is_stationary = True

# Print the results
print("ADF Statistic:", adf_statistic)
print("p-value:", p_value)
print("Critical Values:", critical_values)

if is_stationary:
    print("The time series is stationary.")
else:
    print("The time series is not stationary.")
```

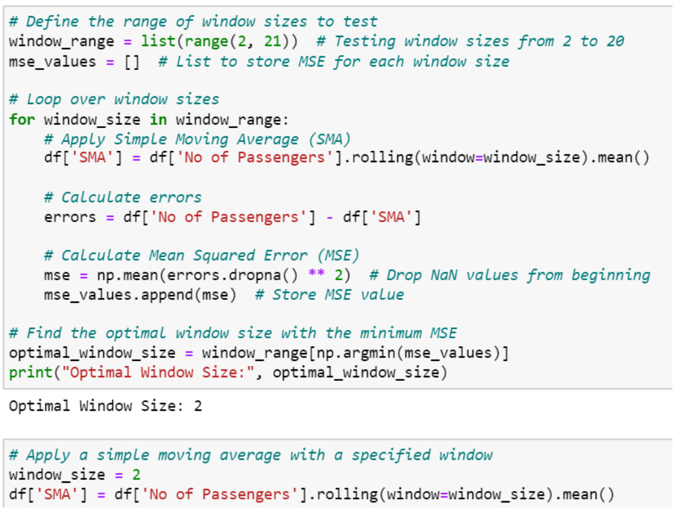
```
ADF Statistic: 0.8153688792060482
p-value: 0.991880243437641
Critical Values: {'1%': -3.4816817173418295, '5%': -2.8840418343195267, '10%': -2.578770059171598}
The time series is not stationary.
```

Figure 7: Stationarity Test

A stationarity test, such as the Augmented Dickey-Fuller (ADF) test, is used to determine if a time series is stationary. Stationarity in a time series means that its statistical properties, like mean and variance, do not change over time. Stationary data is a key assumption for many time series forecasting methods, especially ARIMA and SARIMA based models. This is the probability of observing the given result if the time series is not stationary (i.e., under the null hypothesis). A p-value below a certain threshold (commonly 0.05) suggests rejecting the null hypothesis, indicating stationarity. The ADF Statistic is 0.815, which is greater than all critical values. The p-value is 0.992, which is far above the common threshold of 0.05. Since the ADF Statistic is higher than the critical values, and the p-value is significantly greater than 0.05, we cannot reject the null hypothesis. This indicates that the time series is not stationary. The fact that the dataset is not stationary confirms the earlier observation of an upward trend and seasonality. Non-stationary data can lead to unreliable forecasts if not properly transformed, so it's important to make the series stationary before applying certain forecasting models.

Modelling – Python

Moving Average (MA)



```
# Define the range of window sizes to test
window_range = list(range(2, 21))  # Testing window sizes from 2 to 20
mse_values = []  # List to store MSE for each window size

# Loop over window sizes
for window_size in window_range:
    # Apply Simple Moving Average (SMA)
    df['SMA'] = df['No of Passengers'].rolling(window=window_size).mean()

    # Calculate errors
    errors = df['No of Passengers'] - df['SMA']

    # Calculate Mean Squared Error (MSE)
    mse = np.mean(errors.dropna() ** 2)  # Drop NaN values from beginning
    mse_values.append(mse)  # Store MSE value

# Find the optimal window size with the minimum MSE
optimal_window_size = window_range[np.argmin(mse_values)]
print("Optimal Window Size:", optimal_window_size)
```

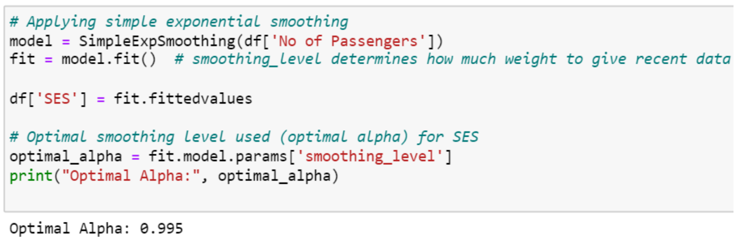
```
Optimal Window Size: 2
```

```
# Apply a simple moving average with a specified window
window_size = 2
df['SMA'] = df['No of Passengers'].rolling(window=window_size).mean()
```

Figure 8: Fit using MA

Moving Average involves calculating the average of a subset of data points (known as a window) as it moves through the dataset. The size of the window determines how much smoothing occurs: a larger window leads to more smoothing, while a smaller window retains more of the original data's variability. To find the optimal window size, one approach is to experiment with different window sizes and evaluate the error metrics, such as the Mean Squared Error (MSE). The goal is to identify the window size that yields the lowest MSE, indicating the best balance between smoothing and preserving the data's underlying patterns. In this analysis, a moving average model was constructed with varying window sizes, and the resulting MSE was calculated for each. The optimal window size was found to be 2, indicating that averaging every two consecutive data points yielded the best smoothing while minimizing the MSE, thus the window size 2 is used to model the data.

Simple Exponential Smoothing (SES)



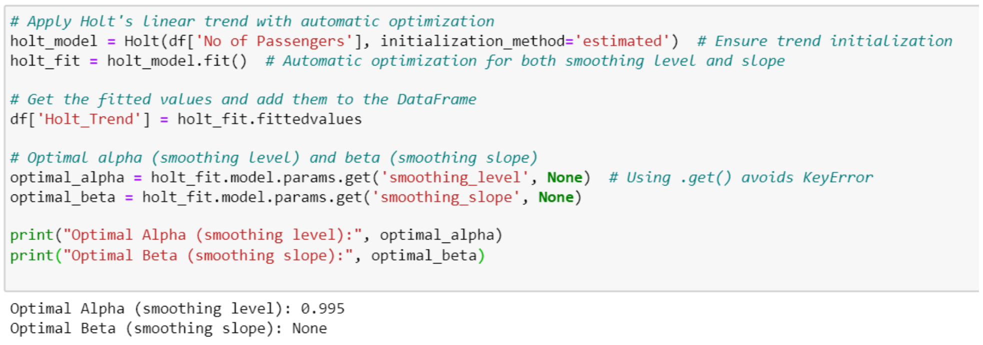
```
# Applying simple exponential smoothing
model = SimpleExpSmoothing(df['No of Passengers'])
fit = model.fit()  # smoothing_level determines how much weight to give recent data

df['SES'] = fit.fittedvalues

# Optimal smoothing level used (optimal alpha) for SES
optimal_alpha = fit.model.params['smoothing_level']
print("Optimal Alpha:", optimal_alpha)
```

```
Optimal Alpha: 0.995
```

Figure 9: Fit using SES

Simple Exponential Smoothing (SES) is a time series forecasting method that uses a single smoothing parameter, alpha, to determine the weighting applied to recent observations. An alpha value close to 1 means that recent data points carry more weight, while an alpha closer to 0 indicates that historical data has more influence. To find the optimal alpha for an SES model, parameter tuning can be conducted, which involves testing different alpha values and evaluating error metrics like Mean Squared Error (MSE). However, in this project, the SES model was fitted without explicitly stating the alpha, allowing Python to automatically determine the optimal value based on the data. This approach led to an optimal alpha of 0.995, suggesting that the model strongly emphasizes recent data to forecast future trends. This high alpha value indicates that the time series data might exhibit significant recent variability or rapid changes, warranting a stronger focus on the latest information. With the optimal alpha identified, the SES model can be used to fit the time series data, offering a forecast that aligns with its characteristics.

Holt’s Model



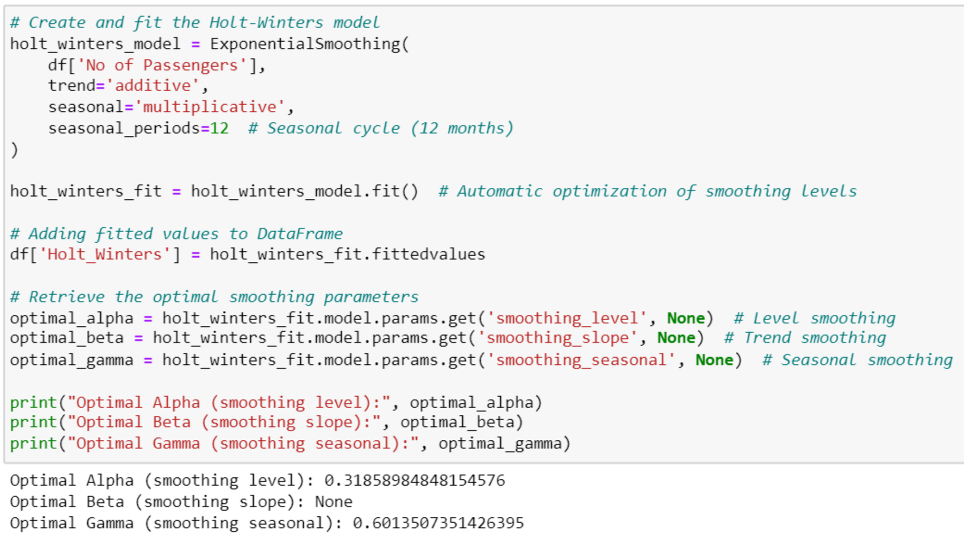
```
# Apply Holt's linear trend with automatic optimization
holt_model = Holt(df['No of Passengers'], initialization_method='estimated')  # Ensure trend initialization
holt_fit = holt_model.fit()  # Automatic optimization for both smoothing level and slope

# Get the fitted values and add them to the DataFrame
df['Holt_Trend'] = holt_fit.fittedvalues

# Optimal alpha (smoothing level) and beta (smoothing slope)
optimal_alpha = holt_fit.model.params.get('smoothing_level', None)  # Using .get() avoids KeyError
optimal_beta = holt_fit.model.params.get('smoothing_slope', None)

print("Optimal Alpha (smoothing level):", optimal_alpha)
print("Optimal Beta (smoothing slope):", optimal_beta)
```

```
Optimal Alpha (smoothing level): 0.995
Optimal Beta (smoothing slope): None
```

Figure 10: Fit using Holt

In time series forecasting, Holt's linear trend method (also known as Holt's model) extends Simple Exponential Smoothing (SES) by including a trend component in addition to the level. This approach introduces two parameters: alpha (smoothing level) and beta (smoothing slope). Alpha controls the weight given to the most recent data points, while beta adjusts the strength of the trend over time. In this project, Holt's model was fitted without explicitly specifying the alpha and beta parameters. This allowed Python to automatically find the optimal values based on the data to minimize error metrics like Mean Squared Error (MSE). Here are the optimal parameters found for Holt's model: The optimal alpha of 0.995 indicates that the smoothing relies heavily on recent data, suggesting that the level of the series is primarily driven by recent observations. The absence of an optimal beta suggests that a linear trend component might not be significant in this dataset, or that the trend is minimal compared to the overall level and variability. This could imply that while the data has an upward trend, it may not require explicit smoothing or adjustment for slope. With these optimal parameters, Holt's model is fitted to the time series data, allowing for a forecast that captures the overall trend without overfitting to specific patterns or noise.

Winter's Model


```
# Create and fit the Holt-Winters model
holt_winters_model = ExponentialSmoothing(
    df['No of Passengers'],
    trend='additive',
    seasonal='multiplicative',
    seasonal_periods=12  # Seasonal cycle (12 months)
)

holt_winters_fit = holt_winters_model.fit()  # Automatic optimization of smoothing levels

# Adding fitted values to DataFrame
df['Holt_Winters'] = holt_winters_fit.fittedvalues

# Retrieve the optimal smoothing parameters
optimal_alpha = holt_winters_fit.model.params.get('smoothing_level', None)  # Level smoothing
optimal_beta = holt_winters_fit.model.params.get('smoothing_slope', None)  # Trend smoothing
optimal_gamma = holt_winters_fit.model.params.get('smoothing_seasonal', None)  # Seasonal smoothing

print("Optimal Alpha (smoothing level):", optimal_alpha)
print("Optimal Beta (smoothing slope):", optimal_beta)
print("Optimal Gamma (smoothing seasonal):", optimal_gamma)
```

```
Optimal Alpha (smoothing level): 0.31858984848154576
Optimal Beta (smoothing slope): None
Optimal Gamma (smoothing seasonal): 0.6013507351426395
```

Figure 11: Fit using Winter

Winter's method, also known as Holt-Winters' model, is an extension of Holt's model that includes a seasonal component in addition to the level and trend. This approach introduces three smoothing parameters: alpha (smoothing level), beta (smoothing slope), and gamma (smoothing seasonal). These parameters determine the relative weight given to recent data in calculating the level, trend, and seasonality of the time series, respectively. In this project, the Winter model was fitted without explicitly stating the alpha, beta, and gamma parameters, allowing Python to automatically determine the optimal values based on the dataset. This approach uses optimization techniques to find the parameter values that minimize a specific error metric, typically Mean Squared Error (MSE).The alpha value of approximately 0.31 indicates that the smoothing of the level relies on a balance between recent and historical data. The absence of an optimal beta (smoothing slope) implies that the trend component is minimal or stable over time, indicating a consistent underlying growth pattern. The gamma value of approximately 0.6014 suggests that the seasonal component carries significant weight, indicating pronounced seasonality in the data. With these optimal values, the Winter model was fitted to the time series data, allowing for a robust forecast that captures level, seasonal patterns, and trend.

Box-Jenkins Method

The Box-Jenkins method is a systematic approach to modeling and forecasting time series data. It was developed by statisticians George Box and Gwilym Jenkins and has become a cornerstone in time series analysis, especially with the introduction of ARIMA (Autoregressive Integrated Moving Average) models. This method involves a series of steps to identify, estimate, and validate the most suitable model for a given time series. The method is flexible, accommodating both non-seasonal and seasonal patterns, and can be applied to a wide range of time series data. The rigorous approach to model validation ensures that the resulting forecasts are robust and accurate, making it a preferred method for time series analysis and forecasting in various fields. The steps are as follows:

- Model Identification: This step involves examining the data to determine the appropriate model structure. Tools like the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) are used to estimate the potential autoregressive and moving average components, along with the degree of differencing needed for stationarity.

- Parameter Estimation: Once the model structure is identified, the next step is to estimate the parameters of the ARIMA or SARIMA model. This can be done using maximum likelihood estimation or other statistical techniques.
- Model Validation: After estimating the parameters, the model is validated by examining the residuals (the difference between observed and predicted values). The residuals should exhibit random behavior, indicating that the model has successfully captured the underlying patterns in the data.
- Model Selection: This step involves selecting the best model based on information criteria like Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC). These criteria help determine the most parsimonious model that adequately explains the data.
- Forecasting: With the best model identified and validated, it can be used for forecasting future values in the time series.

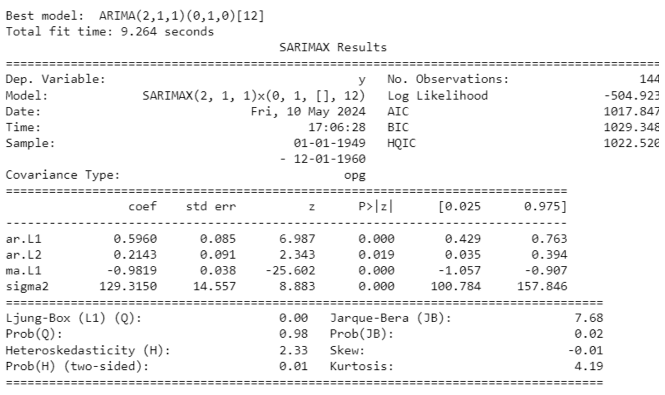
Seasonal Auto-Regressive Integrated Moving Average (SARIMA)

Seasonal Autoregressive Integrated Moving Average (SARIMA) is a widely used time series forecasting model that extends the classic ARIMA (Autoregressive Integrated Moving Average) model by incorporating seasonal components. SARIMA is particularly effective when dealing with time series data that exhibit clear seasonal patterns or cycles. It uses a combination of autoregression, differencing, and moving averages to capture trends and fluctuations, adding seasonal differencing and seasonal autoregressive and moving average terms to accommodate periodic variations. SARIMA has a set of parameters that determine the model's behavior. These parameters are:

p: Number of non-seasonal autoregressive (AR) terms.
d: Number of non-seasonal differencing (I, or integrated) terms.
q: Number of non-seasonal moving average (MA) terms.
P: Number of seasonal autoregressive terms.
D: Number of seasonal differencing terms.
Q: Number of seasonal moving average terms.
s: The length of the seasonal cycle (e.g., 12 for monthly data with an annual cycle).

In this project, two approaches were used to identify the best SARIMA model:

1. Auto.ARIMA: The first approach used the auto.arima function in Python to automatically determine the optimal parameters for the SARIMA model. This method evaluates different combinations of parameters and selects the best model based on information criteria such as AIC (Akaike Information Criterion).



```
Best model:  ARIMA(2,1,1)(0,1,0)[12]
Total fit time: 9.264 seconds
                                      SARIMAX Results
==========================================================================================
Dep. Variable:                                   y   No. Observations:                 144
Model:             SARIMAX(2, 1, 1)x(0, 1, [], 12)   Log Likelihood                -504.923
Date:                             Fri, 10 May 2024   AIC                           1017.847
Time:                                     17:06:28   BIC                           1029.348
Sample:                                 01-01-1949   HQIC                          1022.520
                                      - 12-01-1960
Covariance Type:                               opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.5960      0.085      6.987      0.000       0.429       0.763
ar.L2          0.2143      0.091      2.343      0.019       0.035       0.394
ma.L1         -0.9819      0.038    -25.602      0.000      -1.057      -0.907
sigma2       129.3150     14.557      8.883      0.000     100.784     157.846
===================================================================================
Ljung-Box (L1) (Q):                   0.00   Jarque-Bera (JB):                 7.68
Prob(Q):                              0.98   Prob(JB):                         0.02
Heteroskedasticity (H):               2.33   Skew:                            -0.01
Prob(H) (two-sided):                  0.01   Kurtosis:                         4.19
===================================================================================
```

Figure 12: Fit using autoarima

Using auto.arima, the best model found for this dataset was ARIMA(2,1,1)(0,1,0)[12], indicating SARIMA model, a seasonal differencing term with a periodicity of 12 and fitted using the data.

2. Manual Parameter Estimation: The second approach involved manual estimation of the parameters by analyzing the Partial Autocorrelation Function (PACF) and Autocorrelation Function (ACF). This method requires a deeper understanding of time series patterns and allows for more customized model fitting. In time series analysis, the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) are essential tools for examining the underlying structure and identifying patterns in the data. When manually estimating parameters for ARIMA or SARIMA models, these plots help determine the model order by identifying significant correlations and lag structures.

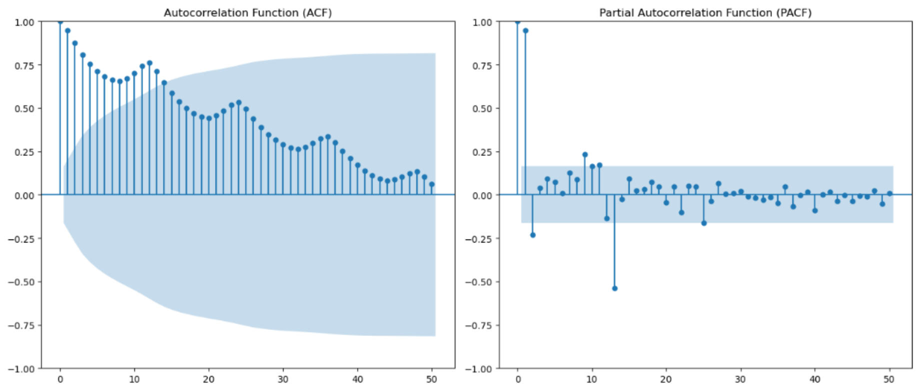


Figure 13: Original ACF & PACF

The ACF measures the correlation between a time series and its lags. A spike in the ACF at specific lags can indicate the presence of autocorrelation or seasonality. In a SARIMA model, periodicity in the ACF can suggest a seasonal pattern. The PACF measures the correlation between a time series and its lags, accounting for the influence of intervening lags. It is useful for identifying the autoregressive order (p) in ARIMA/SARIMA models. In the context of seasonal time series, periodicity in the ACF plot is a key indicator of seasonality. A peak in the ACF at every 12th lag, for example, suggests a 12-period seasonal pattern, which is common in monthly data with an annual cycle. This observation can guide the manual parameter estimation process for SARIMA models.

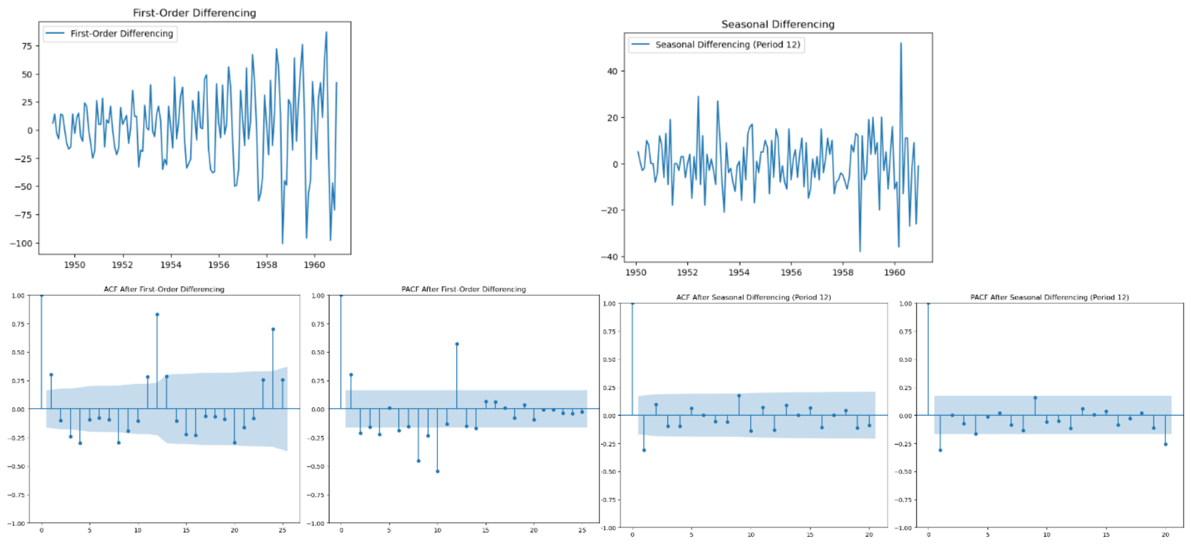
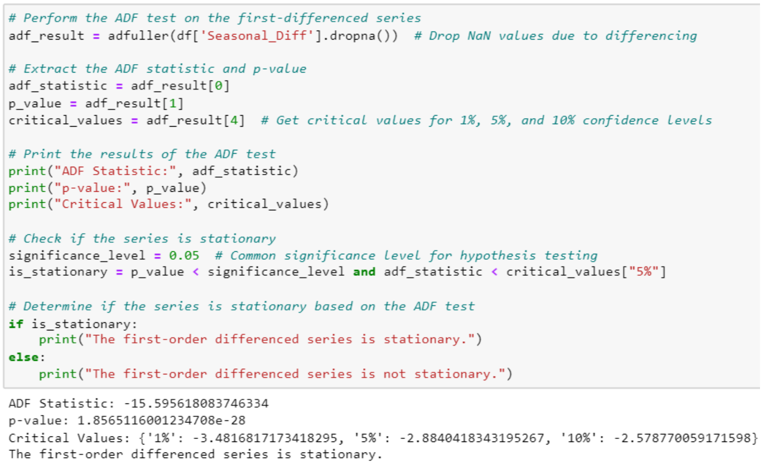


Figure 14: Differenced ACF & PACF

To effectively model non-stationary time series data, it's crucial to apply differencing to stabilize the series. Differencing removes trends and other forms of non-stationarity, making the data suitable for time series analysis and forecasting. When seasonal patterns are also present, seasonal differencing can help eliminate periodic fluctuations.

To make a time series stationary, the first step is to apply non-seasonal differencing. This process involves subtracting a previous observation from the current observation, thereby removing trends and stabilizing the mean. The differenced data represents the change between consecutive observations, which helps reduce variations due to non-stationarity. Once the data is differenced, the Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) are plotted to examine the new structure of the time series. This step helps to identify significant correlations and provides insight into the autoregressive and moving average components, where to estimate the parameter for SARIMA models.

If the time series still exhibits seasonality after first differencing, which we can observe in the ACF and PACF of 1st differencing, there is a spike at each 12th lag, the next step is to apply seasonal differencing. This step is like non-seasonal differencing but involves subtracting values from observations at a specific seasonal lag (e.g., 12 months apart). This helps remove periodic patterns and further stabilizes the time series. This process involves subtracting an observation from its equivalent in the previous seasonal cycle. In this monthly data with an annual cycle, the observation from 12 months prior is subtracted to remove seasonal effects and help to estimate the parameters of SARIMA.

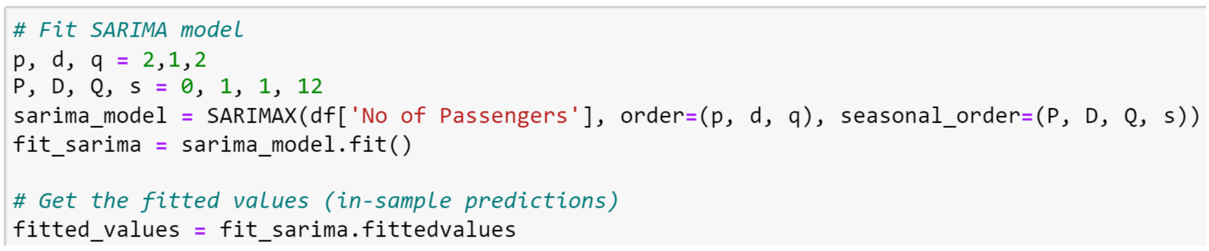


```python
# Perform the ADF test on the first-differenced series
adf_result = adfuller(df['Seasonal_Diff'].dropna())  # Drop NaN values due to differencing

# Extract the ADF statistic and p-value
adf_statistic = adf_result[0]
p_value = adf_result[1]
critical_values = adf_result[4]  # Get critical values for 1%, 5%, and 10% confidence levels

# Print the results of the ADF test
print("ADF Statistic:", adf_statistic)
print("p-value:", p_value)
print("Critical Values:", critical_values)

# Check if the series is stationary
significance_level = 0.05  # Common significance level for hypothesis testing
is_stationary = p_value < significance_level and adf_statistic < critical_values["5%"]

# Determine if the series is stationary based on the ADF test
if is_stationary:
    print("The first-order differenced series is stationary.")
else:
    print("The first-order differenced series is not stationary.")
```

```
ADF Statistic: -15.595618083746334
p-value: 1.8565116001234708e-28
Critical Values: {'1%': -3.4816817173418295, '5%': -2.8840418343195267, '10%': -2.578770059171598}
The first-order differenced series is stationary.
```

Figure 15: Stationarity Test After Differencing

Once non-seasonal and seasonal differencing are applied to a time series, the next step is to check if the data has become stationary using ADF test. The ADF statistic of 15.595618083746334 is significantly lower than all critical values, indicating strong stationarity. The extremely low p-value (1.8565116001234708e-28) suggests a high level of confidence in rejecting the null hypothesis, reinforcing that the series is stationary. The ADF statistic is well below all critical values, supporting the conclusion of stationarity. Since the data is stationary now, we can use the ACF and PACF plots to estimate the appropriate parameters for SARIMA. The ACF can help identify potential moving average components, while the PACF can indicate autoregressive components. Seasonal patterns in the ACF and PACF will guide the choice of seasonal parameters.



```
# Fit SARIMA model
p, d, q = 2,1,2
P, D, Q, s = 0, 1, 1, 12
sarima_model = SARIMAX(df['No of Passengers'], order=(p, d, q), seasonal_order=(P, D, Q, s))
fit_sarima = sarima_model.fit()

# Get the fitted values (in-sample predictions)
fitted_values = fit_sarima.fittedvalues
```

Figure 16: Fit using SARIMA Manual

Parameters for Non-Seasonal Components

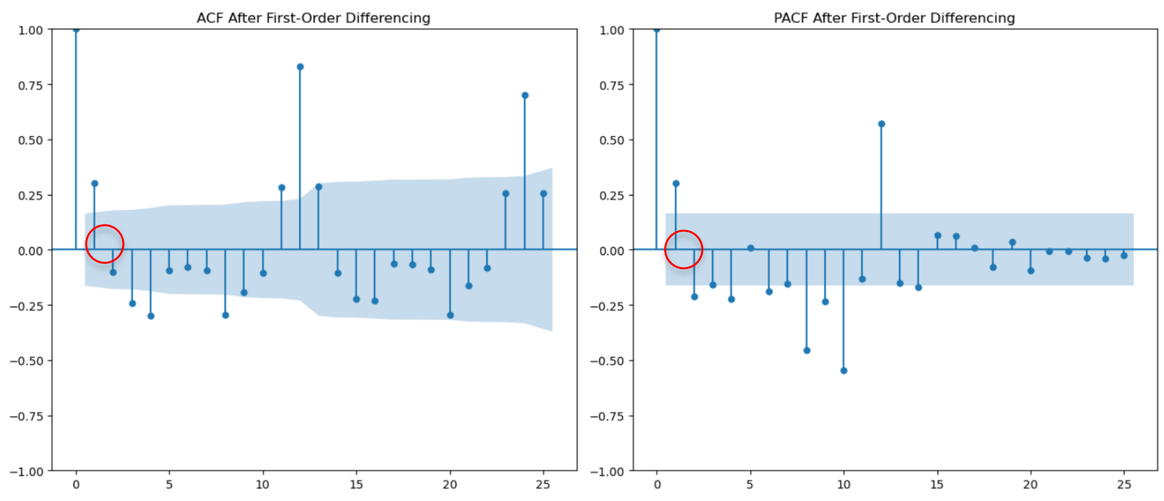


Figure 17: 1st Differencing ACF & PACF

- p (Autoregressive Order): This parameter represents the number of autoregressive terms in the non-seasonal component. It indicates how many past values are used to predict the current value. A PACF plot helps determine p. A significant spike at the 2nd lag in the PACF suggests a second-order autoregressive structure. Thus, a choice of p = 2 indicates two AR terms.

- d (Differencing Order): This parameter indicates the number of times the series must be differenced to achieve stationarity. Given that the data is stationary after first-order differencing, the chosen value of d = 1 reflects this. It suggests that one differencing operation was sufficient to stabilize the series.

- q (Moving Average Order): This parameter represents the number of moving average terms in the non-seasonal component. The ACF plot helps determine q. Since there's a significant spike at the 2nd lag in the ACF, it indicates a second order moving average structure. Thus, a choice of q = 2 aligns with this observation.

Parameters for Seasonal Components

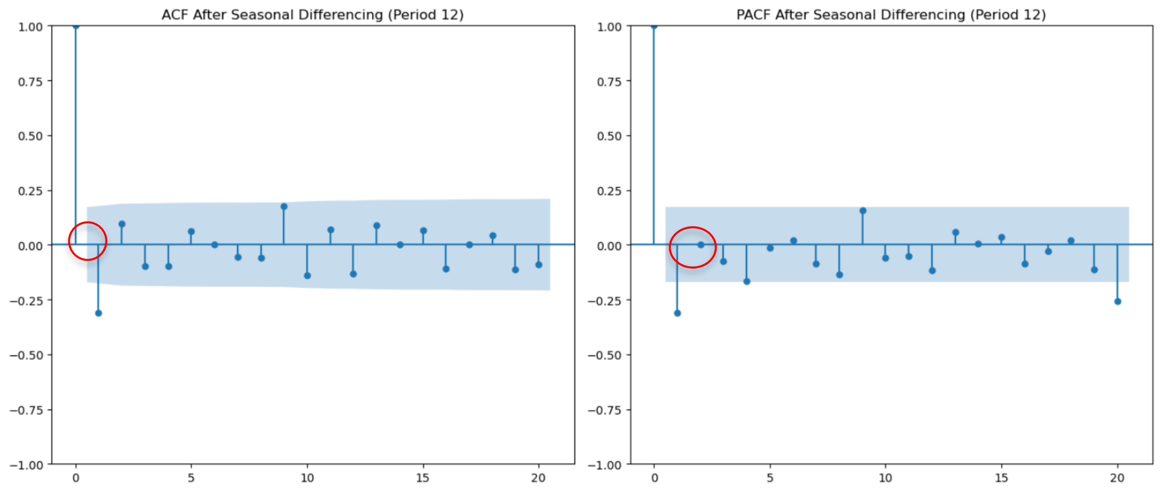


Figure 18: Seasonal Differencing ACF & PACF

- P (Seasonal Autoregressive Order): This parameter represents the number of autoregressive terms in the seasonal component. ACF and PACF plots can both inform this choice. Since the seasonal pattern in the ACF doesn't show a strong AR structure, we choose P = 0, indicating no seasonal AR terms.

- D (Seasonal Differencing Order): This parameter indicates the number of times the series must be differenced to remove seasonal patterns. A significant drop-off in the ACF after seasonal differencing indicates successful removal of seasonality. A value of D = 1 suggests that one seasonal differencing operation was sufficient.

- Q (Seasonal Moving Average Order): This parameter represents the number of moving average terms in the seasonal component. Since the ACF shows a significant spike at the seasonal lag, at the 12th lag for this monthly data, it indicates a seasonal moving average component. A value of Q = 1 aligns with this observation.

- s (Seasonal Periodicity): This parameter indicates the length of the seasonal cycle. s = 12 reflects a 12-month seasonal cycle, typical for monthly data with annual patterns like the Air Passenger data chosen.

The data is fitted using the estimated parameters for further analysis.

Modelling – Microsoft Excel

To model the time series data in Microsoft Excel, we used the optimal parameters identified through our analysis in Python. This approach allows for a consistent baseline, ensuring a fair comparison between models developed in different platforms.

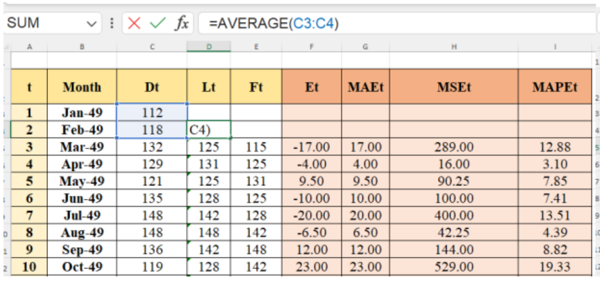
Moving Average (MA)

In Microsoft Excel, the moving average calculation is performed using a window size of 2, similar to what was done in Python. The Lt and forecasts are then calculated using the following formulas:

$$L_t = \frac{(D_t + D_{t-1})}{2}$$

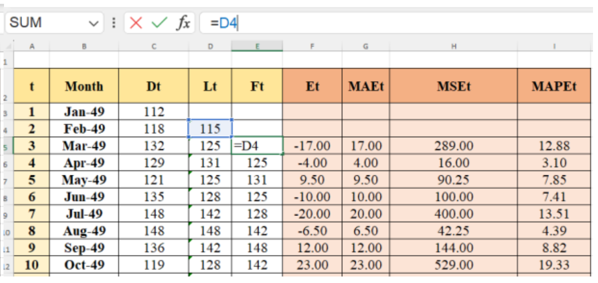
$$F_{t+1} = L_t$$

$$F_{t+n} = L_t$$



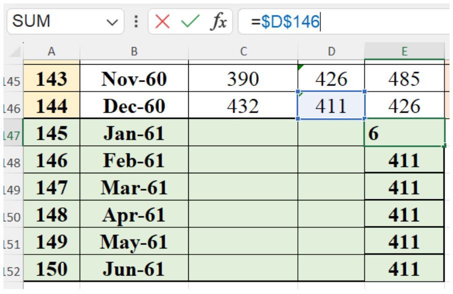
SUM =AVERAGE(C3:C4)

t	Month	Dt	Lt	Ft	Et	MAEt	MSEt	MAPEt
1	Jan-49	112						
2	Feb-49	118	C4)					
3	Mar-49	132	125	115	-17.00	17.00	289.00	12.88
4	Apr-49	129	131	125	-4.00	4.00	16.00	3.10
5	May-49	121	125	131	9.50	9.50	90.25	7.85
6	Jun-49	135	128	125	-10.00	10.00	100.00	7.41
7	Jul-49	148	142	128	-20.00	20.00	400.00	13.51
8	Aug-49	148	148	142	-6.50	6.50	42.25	4.39
9	Sep-49	136	142	148	12.00	12.00	144.00	8.82
10	Oct-49	119	128	142	23.00	23.00	529.00	19.33



SUM =D4

t	Month	Dt	Lt	Ft	Et	MAEt	MSEt	MAPEt
1	Jan-49	112						
2	Feb-49	118	115					
3	Mar-49	132	125	=D4	-17.00	17.00	289.00	12.88
4	Apr-49	129	131	125	-4.00	4.00	16.00	3.10
5	May-49	121	125	131	9.50	9.50	90.25	7.85
6	Jun-49	135	128	125	-10.00	10.00	100.00	7.41
7	Jul-49	148	142	128	-20.00	20.00	400.00	13.51
8	Aug-49	148	148	142	-6.50	6.50	42.25	4.39
9	Sep-49	136	142	148	12.00	12.00	144.00	8.82
10	Oct-49	119	128	142	23.00	23.00	529.00	19.33



SUM =$D$146

A	B	C	D	E
143	Nov-60	390	426	485
144	Dec-60	432	411	426
145	Jan-61			6
146	Feb-61			411
147	Mar-61			411
148	Apr-61			411
149	May-61			411
150	Jun-61			411

Figure 19: MA Excel

Simple Exponential Smoothing (SES)

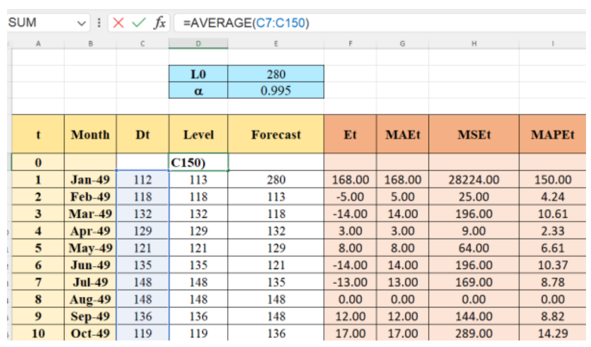
In Microsoft Excel, for Simple Exponential Smoothing (SES), the same alpha value, 0.995 used in Python is applied. This alpha value determines the smoothing factor, controlling the weight given to recent observations in generating forecasts. Using this alpha value, the initial level (L0), Lt and forecast can be calculated using the following formulas:

$$L_0 = \frac{\sum_{i=1}^{144} D_i}{144}$$

$$L_t = \alpha D_{t+1} + (1-\alpha)L_t$$

$$F_{t+1} = L_t$$

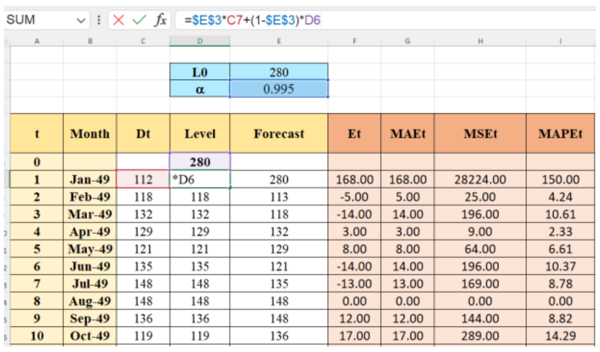
$$F_{t+n} = L_t$$



SUM =AVERAGE(C7:C150)

L0	280
α	0.995

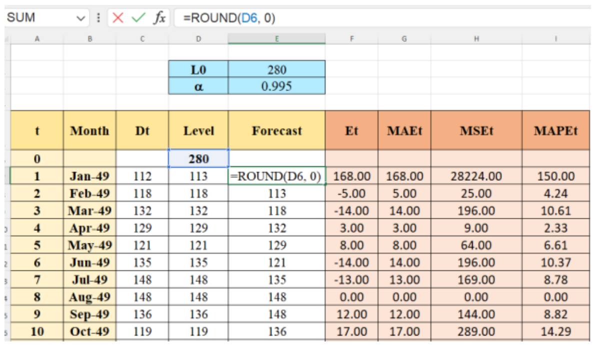
t	Month	Dt	Level	Forecast	Et	MAEt	MSEt	MAPEt
0			C150)					
1	Jan-49	112	113	280	168.00	168.00	28224.00	150.00
2	Feb-49	118	118	113	-5.00	5.00	25.00	4.24
3	Mar-49	132	132	118	-14.00	14.00	196.00	10.61
4	Apr-49	129	129	132	3.00	3.00	9.00	2.33
5	May-49	121	121	129	8.00	8.00	64.00	6.61
6	Jun-49	135	135	121	-14.00	14.00	196.00	10.37
7	Jul-49	148	148	135	-13.00	13.00	169.00	8.78
8	Aug-49	148	148	148	0.00	0.00	0.00	0.00
9	Sep-49	136	136	148	12.00	12.00	144.00	8.82
10	Oct-49	119	119	136	17.00	17.00	289.00	14.29



SUM =$E$3*C7+(1-$E$3)*D6

L0	280
α	0.995

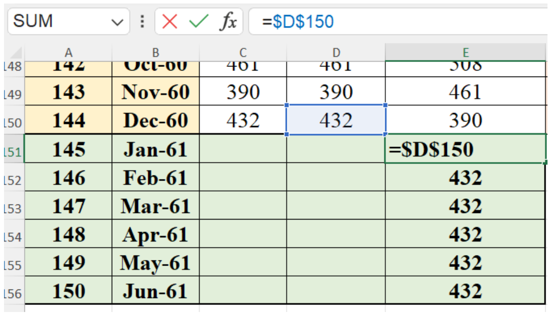
t	Month	Dt	Level	Forecast	Et	MAEt	MSEt	MAPEt
0			280					
1	Jan-49	112	*D6	280	168.00	168.00	28224.00	150.00
2	Feb-49	118	118	113	-5.00	5.00	25.00	4.24
3	Mar-49	132	132	118	-14.00	14.00	196.00	10.61
4	Apr-49	129	129	132	3.00	3.00	9.00	2.33
5	May-49	121	121	129	8.00	8.00	64.00	6.61
6	Jun-49	135	135	121	-14.00	14.00	196.00	10.37
7	Jul-49	148	148	135	-13.00	13.00	169.00	8.78
8	Aug-49	148	148	148	0.00	0.00	0.00	0.00
9	Sep-49	136	136	148	12.00	12.00	144.00	8.82
10	Oct-49	119	119	136	17.00	17.00	289.00	14.29



SUM =ROUND(D6, 0)

L0	280
α	0.995

t	Month	Dt	Level	Forecast	Et	MAEt	MSEt	MAPEt
0			280					
1	Jan-49	112	113	=ROUND(D6, 0)	168.00	168.00	28224.00	150.00
2	Feb-49	118	118	113	-5.00	5.00	25.00	4.24
3	Mar-49	132	132	118	-14.00	14.00	196.00	10.61
4	Apr-49	129	129	132	3.00	3.00	9.00	2.33
5	May-49	121	121	129	8.00	8.00	64.00	6.61
6	Jun-49	135	135	121	-14.00	14.00	196.00	10.37
7	Jul-49	148	148	135	-13.00	13.00	169.00	8.78
8	Aug-49	148	148	148	0.00	0.00	0.00	0.00
9	Sep-49	136	136	148	12.00	12.00	144.00	8.82
10	Oct-49	119	119	136	17.00	17.00	289.00	14.29



SUM =$D$150

	A	B	C	D	E
148	142	Oct-60	461	461	508
149	143	Nov-60	390	390	461
150	144	Dec-60	432	432	390
151	145	Jan-61			=$D$150
152	146	Feb-61			432
153	147	Mar-61			432
154	148	Apr-61			432
155	149	May-61			432
156	150	Jun-61			432

Figure 20: SES Excel

Holt's Model

In Microsoft Excel, the Holt's Model is implemented using the same parameters as used in Python. The initial level (L0) is determined by applying the Regression tool, where the time series data is regressed against time (X) and demand of passengers (Y). From the regression summary, the intercept term provides the value for L0, which is found to be 88 after rounding off. Additionally, the slope coefficient represents the trend component (T0), which is calculated to be 3. On the other hand, the values of L0 and T0 can be found using the trendline option in the graphing section too.

Once L0 and T0 are obtained, the following formulas is applied to compute Lt (smoothed level), Tt (trend component), and Ft (forecast):

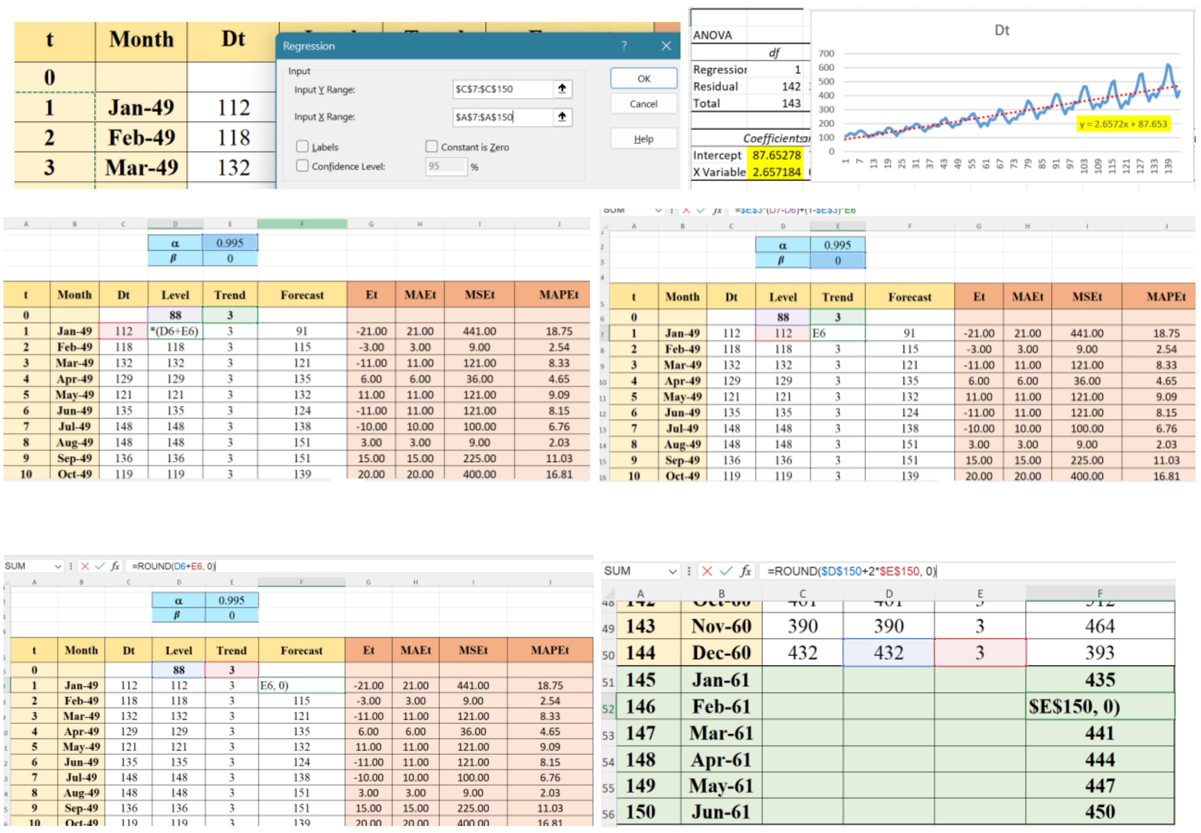
$$D_t = 3t + 88, L_0 = 88, T_0 = 3$$

$$L_{t+1} = \alpha D_{t+1} + (1 - \alpha)(L_t + T_t)$$

$$T_{t+1} = \beta(L_{t+1} - L_t) + (1-\beta)T_t$$

$$F_{t+1} = L_t + T_t$$

$$F_{t+n} = L_t + nT_t$$



t	Month	Dt
0		
1	Jan-49	112
2	Feb-49	118
3	Mar-49	132

Regression

Input

Input Y Range: $C$7:$C$150

Input X Range: $A$7:$A$150

☐ Labels ☐ Constant is Zero

☐ Confidence Level: 95 %

ANOVA	
	df
Regression	1
Residual	142
Total	143
	Coefficients
Intercept	87.65278
X Variable	2.657184

α	0.995
β	0

t	Month	Dt	Level	Trend	Forecast	Et	MAEt	MSEt	MAPEt
0			88	3					
1	Jan-49	112	*(D6+E6)	3	91	-21.00	21.00	441.00	18.75
2	Feb-49	118	118	3	115	-3.00	3.00	9.00	2.54
3	Mar-49	132	132	3	121	-11.00	11.00	121.00	8.33
4	Apr-49	129	129	3	135	6.00	6.00	36.00	4.65
5	May-49	121	121	3	132	11.00	11.00	121.00	9.09
6	Jun-49	135	135	3	124	-11.00	11.00	121.00	8.15
7	Jul-49	148	148	3	138	-10.00	10.00	100.00	6.76
8	Aug-49	148	148	3	151	3.00	3.00	9.00	2.03
9	Sep-49	136	136	3	151	15.00	15.00	225.00	11.03
10	Oct-49	119	119	3	139	20.00	20.00	400.00	16.81

SUM =$E$3*(D7-D6)+(1-$E$3)*E6

α	0.995
β	0

t	Month	Dt	Level	Trend	Forecast	Et	MAEt	MSEt	MAPEt
0			88	3					
1	Jan-49	112	112	E6	91	-21.00	21.00	441.00	18.75
2	Feb-49	118	118	3	115	-3.00	3.00	9.00	2.54
3	Mar-49	132	132	3	121	-11.00	11.00	121.00	8.33
4	Apr-49	129	129	3	135	6.00	6.00	36.00	4.65
5	May-49	121	121	3	132	11.00	11.00	121.00	9.09
6	Jun-49	135	135	3	124	-11.00	11.00	121.00	8.15
7	Jul-49	148	148	3	138	-10.00	10.00	100.00	6.76
8	Aug-49	148	148	3	151	3.00	3.00	9.00	2.03
9	Sep-49	136	136	3	151	15.00	15.00	225.00	11.03
10	Oct-49	119	119	3	139	20.00	20.00	400.00	16.81

SUM =ROUND(D6+E6, 0)

α	0.995
β	0

t	Month	Dt	Level	Trend	Forecast	Et	MAEt	MSEt	MAPEt
0			88	3					
1	Jan-49	112	112	3	E6, 0)	-21.00	21.00	441.00	18.75
2	Feb-49	118	118	3	115	-3.00	3.00	9.00	2.54
3	Mar-49	132	132	3	121	-11.00	11.00	121.00	8.33
4	Apr-49	129	129	3	135	6.00	6.00	36.00	4.65
5	May-49	121	121	3	132	11.00	11.00	121.00	9.09
6	Jun-49	135	135	3	124	-11.00	11.00	121.00	8.15
7	Jul-49	148	148	3	138	-10.00	10.00	100.00	6.76
8	Aug-49	148	148	3	151	3.00	3.00	9.00	2.03
9	Sep-49	136	136	3	151	15.00	15.00	225.00	11.03
10	Oct-49	119	119	3	139	20.00	20.00	400.00	16.81

SUM =ROUND($D$150+2*$E$150, 0)

A	B	C	D	E	F
143	Nov-60	390	390	3	464
144	Dec-60	432	432	3	393
145	Jan-61				435
146	Feb-61				$E$150, 0)
147	Mar-61				441
148	Apr-61				444
149	May-61				447
150	Jun-61				450

Figure 21: Holt Excel

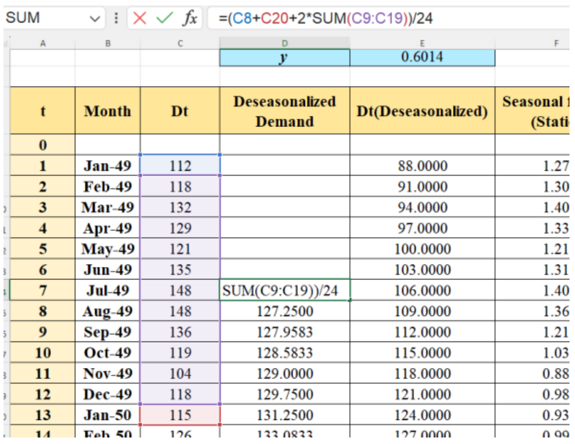
Winter's Model

In Microsoft Excel, the Winter's Model is implemented using the same parameters as in Python. However, before applying the model, a static method is employed to derive the static seasonal factors.

The initial step entails deseasonalizing the demand data with a period (p) of 12, given that the seasonal pattern recurs every 12 months. We select t=7 to ensure that t-(p/2) does not yield zero, as 12/2 equals 6. This selection ensures a smoother deseasonalization process. Since p is even, the following formula is used:

$$\bar{D}_t = \frac{D_{t-\left(\frac{p}{2}\right)} + D_{t+\left(\frac{p}{2}\right)} + \sum_{i=t+1-\left(\frac{p}{2}\right)}^{t-1+\left(\frac{p}{2}\right)} 2D_i}{2(p)}$$

$$\bar{D}_7 = \frac{D_1 + D_{13} + \sum_2^{12} 2D_i}{24}$$

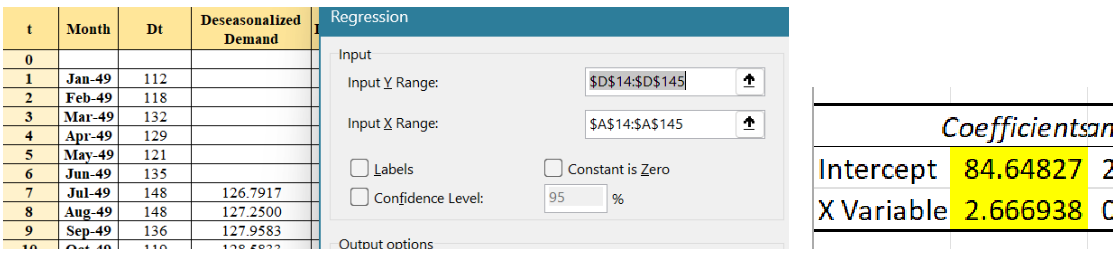


SUM =(C8+C20+2*SUM(C9:C19))/24

			y	0.6014	
t	**Month**	**Dt**	**Deseasonalized Demand**	**Dt(Deseasonalized)**	**Seasonal (Stati**
0					
1	**Jan-49**	112		88.0000	1.27
2	**Feb-49**	118		91.0000	1.30
3	**Mar-49**	132		94.0000	1.40
4	**Apr-49**	129		97.0000	1.33
5	**May-49**	121		100.0000	1.21
6	**Jun-49**	135		103.0000	1.31
7	**Jul-49**	148	SUM(C9:C19))/24	106.0000	1.40
8	**Aug-49**	148	127.2500	109.0000	1.36
9	**Sep-49**	136	127.9583	112.0000	1.21
10	**Oct-49**	119	128.5833	115.0000	1.03
11	**Nov-49**	104	129.0000	118.0000	0.88
12	**Dec-49**	118	129.7500	121.0000	0.98
13	**Jan-50**	115	131.2500	124.0000	0.93
14	**Feb-50**	126	133.0833	127.0000	0.99

Figure 22: Deseasonalised Demand

Once the deseasonalized demand is calculated, the initial level (L0) and trend component (T0) can be determined. After rounding off, L0 is found to be 85, and T0 is 3.



t	**Month**	**Dt**	**Deseasonalized Demand**
0			
1	**Jan-49**	112	
2	**Feb-49**	118	
3	**Mar-49**	132	
4	**Apr-49**	129	
5	**May-49**	121	
6	**Jun-49**	135	
7	**Jul-49**	148	126.7917
8	**Aug-49**	148	127.2500
9	**Sep-49**	136	127.9583
10	**Oct-49**	119	128.5833

Regression

Input

Input Y Range: $D$14:$D$145

Input X Range: $A$14:$A$145

☐ Labels ☐ Constant is Zero

☐ Confidence Level: 95 %

Output options

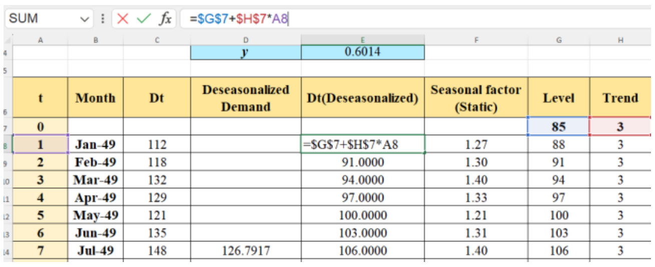
	Coefficients
Intercept	84.64827
X Variable	2.666938

Figure 23: L0 & T0 Values

The deseasonalized Dt and Static seasonal factor is found using formulas below:

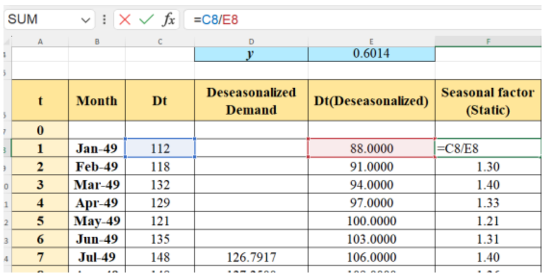
$$\overline{D}_t = 85 + 3t$$

$$\bar{S}_t = \frac{D_t}{\overline{D}_t}$$



=$G$7+$H$7*A8

			y	0.6014			
t	Month	Dt	Deseasonalized Demand	Dt(Deseasonalized)	Seasonal factor (Static)	Level	Trend
0						85	3
1	Jan-49	112		=$G$7+$H$7*A8	1.27	88	3
2	Feb-49	118		91.0000	1.30	91	3
3	Mar-49	132		94.0000	1.40	94	3
4	Apr-49	129		97.0000	1.33	97	3
5	May-49	121		100.0000	1.21	100	3
6	Jun-49	135		103.0000	1.31	103	3
7	Jul-49	148	126.7917	106.0000	1.40	106	3



=C8/E8

			y	0.6014	
t	Month	Dt	Deseasonalized Demand	Dt(Deseasonalized)	Seasonal factor (Static)
0					
1	Jan-49	112		88.0000	=C8/E8
2	Feb-49	118		91.0000	1.30
3	Mar-49	132		94.0000	1.40
4	Apr-49	129		97.0000	1.33
5	May-49	121		100.0000	1.21
6	Jun-49	135		103.0000	1.31
7	Jul-49	148	126.7917	106.0000	1.40

Figure 24: Deseasonalised Dt

To execute the Winter's model, one computes Lt, Tt, and Ft using specific formulas bel. Regarding the seasonal factor of Winter, for periods p1 to p12, static seasonal factors are utilized. However, from period 13 onwards, the following formulas are employed.

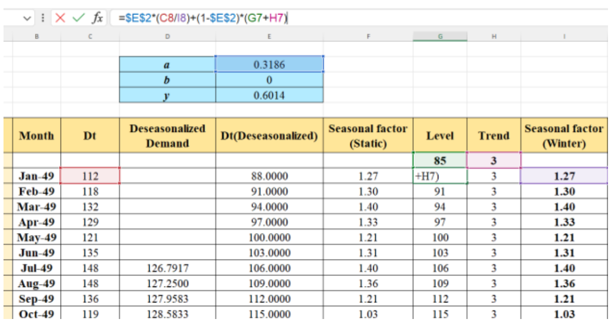
$$L_{t+1} = \alpha\left(\frac{D_{t+1}}{S_{t+1}}\right) + (1-\alpha)(L_t + T_t)$$

$$T_{t+1} = \beta(L_{t+1} - L_t) + (1-\beta)T_t$$

$$S_{t+p+1} = \gamma(\frac{D_{t+1}}{L_{t+1}}) + (1-\gamma)S_{t+!}$$

$$F_{t+1} = (L_t + T_t)(S_{t+1})$$

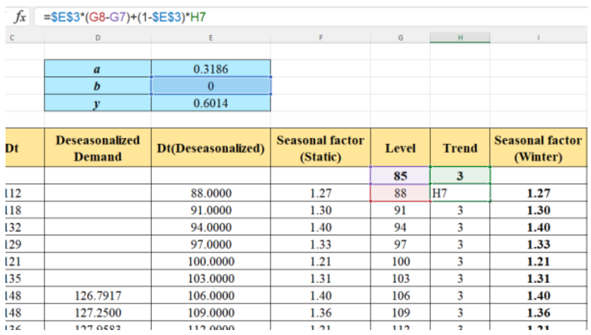
$$F_{t+n} = (L_t + nT_t)(S_{t+n})$$



=$E$2*(C8/I8)+(1-$E$2)*(G7+H7)

a	0.3186
b	0
y	0.6014

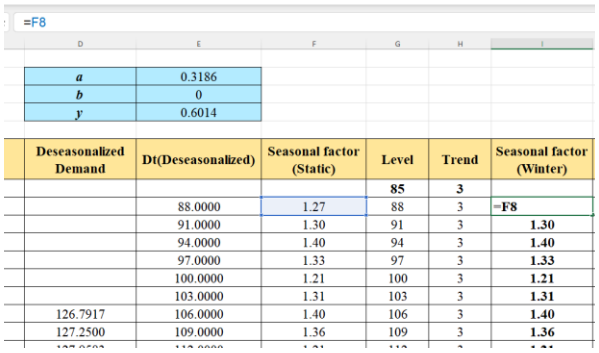
Month	Dt	Deseasonalized Demand	Dt(Deseasonalized)	Seasonal factor (Static)	Level	Trend	Seasonal factor (Winter)
					85	3	
Jan-49	112		88.0000	1.27	+H7)	3	1.27
Feb-49	118		91.0000	1.30	91	3	1.30
Mar-49	132		94.0000	1.40	94	3	1.40
Apr-49	129		97.0000	1.33	97	3	1.33
May-49	121		100.0000	1.21	100	3	1.21
Jun-49	135		103.0000	1.31	103	3	1.31
Jul-49	148	126.7917	106.0000	1.40	106	3	1.40
Aug-49	148	127.2500	109.0000	1.36	109	3	1.36
Sep-49	136	127.9583	112.0000	1.21	112	3	1.21
Oct-49	119	128.5833	115.0000	1.03	115	3	1.03



=$E$3*(G8-G7)+(1-$E$3)*H7

a	0.3186
b	0
y	0.6014

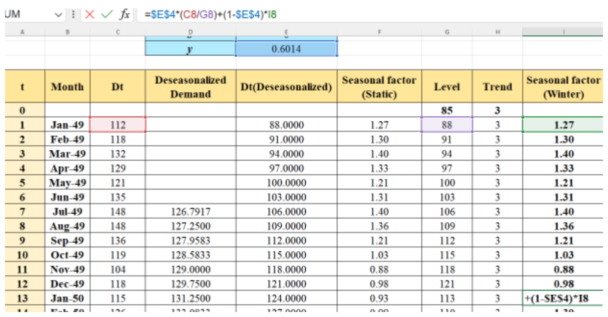
Dt	Deseasonalized Demand	Dt(Deseasonalized)	Seasonal factor (Static)	Level	Trend	Seasonal factor (Winter)
				85	3	
112		88.0000	1.27	88	H7	1.27
118		91.0000	1.30	91	3	1.30
132		94.0000	1.40	94	3	1.40
129		97.0000	1.33	97	3	1.33
121		100.0000	1.21	100	3	1.21
135		103.0000	1.31	103	3	1.31
148	126.7917	106.0000	1.40	106	3	1.40
148	127.2500	109.0000	1.36	109	3	1.36



=F8

a	0.3186
b	0
y	0.6014

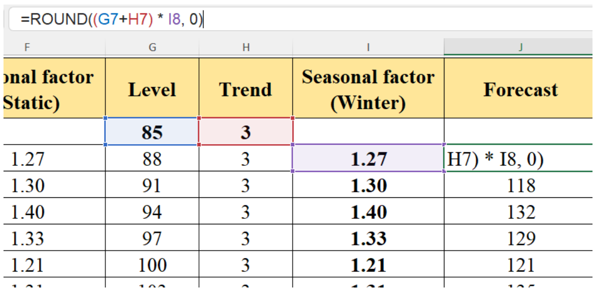
Deseasonalized Demand	Dt(Deseasonalized)	Seasonal factor (Static)	Level	Trend	Seasonal factor (Winter)
			85	3	
	88.0000	1.27	88	3	=F8
	91.0000	1.30	91	3	1.30
	94.0000	1.40	94	3	1.40
	97.0000	1.33	97	3	1.33
	100.0000	1.21	100	3	1.21
	103.0000	1.31	103	3	1.31
126.7917	106.0000	1.40	106	3	1.40
127.2500	109.0000	1.36	109	3	1.36



=$E$4*(C8/G8)+(1-$E$4)*I8

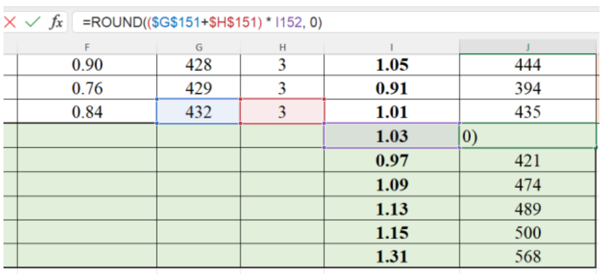
y	0.6014

t	Month	Dt	Deseasonalized Demand	Dt(Deseasonalized)	Seasonal factor (Static)	Level	Trend	Seasonal factor (Winter)
0						85	3	
1	Jan-49	112		88.0000	1.27	88	3	1.27
2	Feb-49	118		91.0000	1.30	91	3	1.30
3	Mar-49	132		94.0000	1.40	94	3	1.40
4	Apr-49	129		97.0000	1.33	97	3	1.33
5	May-49	121		100.0000	1.21	100	3	1.21
6	Jun-49	135		103.0000	1.31	103	3	1.31
7	Jul-49	148	126.7917	106.0000	1.40	106	3	1.40
8	Aug-49	148	127.2500	109.0000	1.36	109	3	1.36
9	Sep-49	136	127.9583	112.0000	1.21	112	3	1.21
10	Oct-49	119	128.5833	115.0000	1.03	115	3	1.03
11	Nov-49	104	129.0000	118.0000	0.88	118	3	0.88
12	Dec-49	118	129.7500	121.0000	0.98	121	3	0.98
13	Jan-50	115	131.2500	124.0000	0.93	113	3	+(1-$E$4)*I8



=ROUND((G7+H7) * I8, 0)

nal factor Static)	Level	Trend	Seasonal factor (Winter)	Forecast
	85	3		
1.27	88	3	1.27	H7) * I8, 0)
1.30	91	3	1.30	118
1.40	94	3	1.40	132
1.33	97	3	1.33	129
1.21	100	3	1.21	121



=ROUND(($G$151+$H$151) * I152, 0)

0.90	428	3	1.05	444
0.76	429	3	0.91	394
0.84	432	3	1.01	435
			1.03	0)
			0.97	421
			1.09	474
			1.13	489
			1.15	500
			1.31	568

Figure 25: Winter in Excel

Forecast Values - Comparison

In Excel, forecasts for Moving Average (MA), Simple Exponential Smoothing (SES), Holt's Model, and Winter's Model were generated for 6 months. Similarly, forecasts for MA, SES, Holt's Model, Winter's Model, and SARIMA were generated using Python. The forecasted values are summarized in the table below and visualized through charts for better comparison:

Table 1: Forecast Values Comparison

Model / Date	Excel				Python				
	MA	SES	Holt	Winter	MA	SES	Holt	Winter	SARIMA
Jan-61	411	432	435	449	411	432	434	445	445
Feb-61	411	432	438	421	411	432	436	418	421
Mar-61	411	432	441	474	411	432	438	465	453
Apr-61	411	432	444	489	411	432	440	495	491
May-61	411	432	447	500	411	432	442	505	503
Jun-61	411	432	450	568	411	432	444	573	566

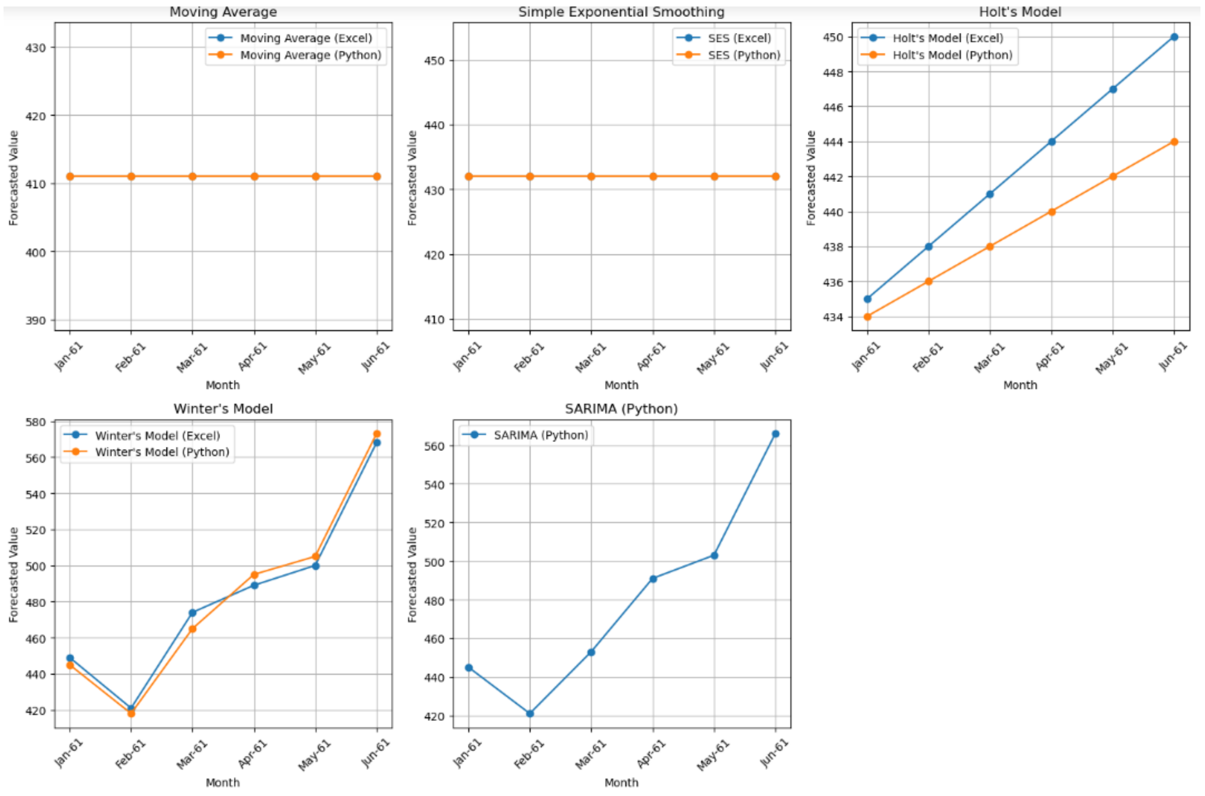


Figure 26: Forecast Comparison Charts

The comparison between the forecasts generated in Excel and Python reveals interesting insights. For Moving Average (MA) and Simple Exponential Smoothing (SES) models, the forecasted values are exactly the same in both Excel and Python. This alignment underscores the consistency and accuracy of the forecasting methods across both platforms.

However, discrepancies are observed in the forecasts generated for Holt's Model and Winter's Model. The differences could be attributed to the rounding off of parameters L0 (initial level), T0 (initial trend), and Ft (seasonal factor) in Excel. Since Excel often rounds off numerical values, slight variations may occur in the forecasted values, leading to discrepancies between the two platforms.

This interpretation underscores the importance of parameter precision in forecasting models, as even minor variations can influence the forecasted outcomes. While both Excel and Python offer powerful tools for forecasting, attention to detail in parameter estimation is crucial for ensuring consistency and accuracy across platforms.

Performance Measures

In the performance measure section, the aim is to evaluate the accuracy and reliability of forecasting models. This assessment is crucial for understanding how well models perform in predicting the values based on historical data. Several key metrics are employed, including Mean Absolute Deviation (MAD), Mean Absolute Percentage Error (MAPE), and Mean Squared Error (MSE).

Forecast error quantifies the disparity between predicted and actual values in forecasting models. The formula for E_t is:

$$E_t = F_t - D_t$$

MAD measures the average magnitude of errors between predicted and actual values, providing a straightforward assessment of the model's accuracy without considering the direction of errors. The formula for MAD is:

$$A_t = |E_t|$$

$$MAD = \frac{1}{n}\sum_{i=1}^{n} A_t$$

MAPE calculates the average percentage difference between predicted and actual values, making it useful for understanding the relative accuracy of forecasts. The formula for MAPE is:

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{E_t}{D_t}\right|$$

MSE quantifies the average squared difference between predicted and actual values, giving more weight to larger errors. It provides a measure of the model's predictive power and sensitivity to outliers. The formula for MSE is:

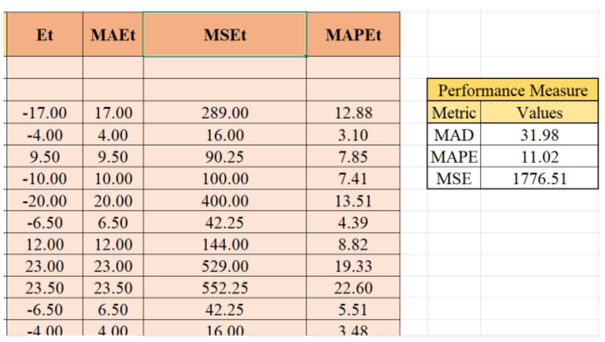
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(E_t)^2$$

Where E_t is the forecast error, A_t is the absolute forecast error, $D_t$is the actual demand value and n is the total number of observations.

The error, MAD, MAPE, MSE are calculated based on the formulas stated above for each model for both Microsoft Excel and Python.

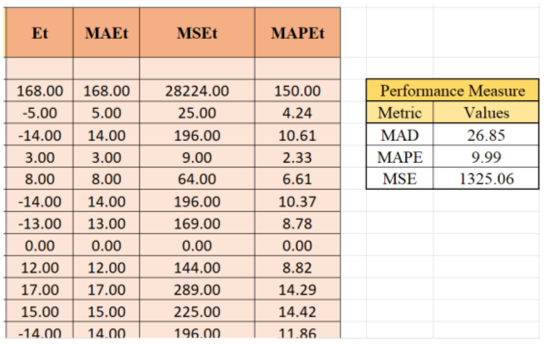
Microsoft Excel

Values



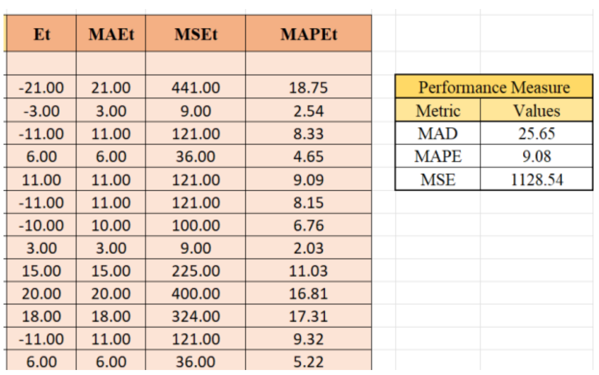
Et	MAEt	MSEt	MAPEt
-17.00	17.00	289.00	12.88
-4.00	4.00	16.00	3.10
9.50	9.50	90.25	7.85
-10.00	10.00	100.00	7.41
-20.00	20.00	400.00	13.51
-6.50	6.50	42.25	4.39
12.00	12.00	144.00	8.82
23.00	23.00	529.00	19.33
23.50	23.50	552.25	22.60
-6.50	6.50	42.25	5.51
-4.00	4.00	16.00	3.48

Performance Measure	
Metric	Values
MAD	31.98
MAPE	11.02
MSE	1776.51



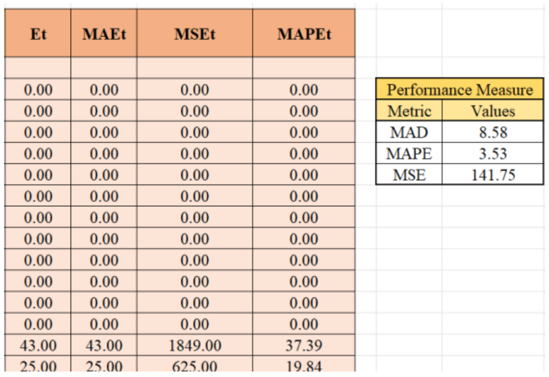
Et	MAEt	MSEt	MAPEt
168.00	168.00	28224.00	150.00
-5.00	5.00	25.00	4.24
-14.00	14.00	196.00	10.61
3.00	3.00	9.00	2.33
8.00	8.00	64.00	6.61
-14.00	14.00	196.00	10.37
-13.00	13.00	169.00	8.78
0.00	0.00	0.00	0.00
12.00	12.00	144.00	8.82
17.00	17.00	289.00	14.29
15.00	15.00	225.00	14.42
-14.00	14.00	196.00	11.86

Performance Measure	
Metric	Values
MAD	26.85
MAPE	9.99
MSE	1325.06



Et	MAEt	MSEt	MAPEt
-21.00	21.00	441.00	18.75
-3.00	3.00	9.00	2.54
-11.00	11.00	121.00	8.33
6.00	6.00	36.00	4.65
11.00	11.00	121.00	9.09
-11.00	11.00	121.00	8.15
-10.00	10.00	100.00	6.76
3.00	3.00	9.00	2.03
15.00	15.00	225.00	11.03
20.00	20.00	400.00	16.81
18.00	18.00	324.00	17.31
-11.00	11.00	121.00	9.32
6.00	6.00	36.00	5.22

Performance Measure	
Metric	Values
MAD	25.65
MAPE	9.08
MSE	1128.54



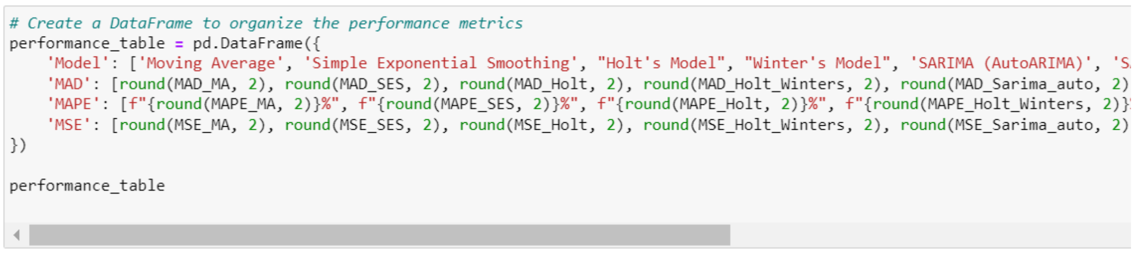
Et	MAEt	MSEt	MAPEt
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00
43.00	43.00	1849.00	37.39
25.00	25.00	625.00	19.84

Performance Measure	
Metric	Values
MAD	8.58
MAPE	3.53
MSE	141.75

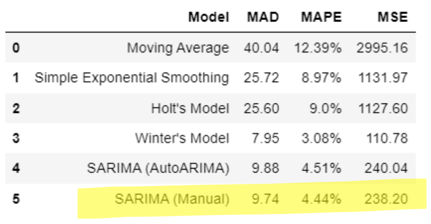
Figure 27: Performance Measure Excel

Python

Values



```
# Create a DataFrame to organize the performance metrics
performance_table = pd.DataFrame({
    'Model': ['Moving Average', 'Simple Exponential Smoothing', "Holt's Model", "Winter's Model", 'SARIMA (AutoARIMA)', 'S
    'MAD': [round(MAD_MA, 2), round(MAD_SES, 2), round(MAD_Holt, 2), round(MAD_Holt_Winters, 2), round(MAD_Sarima_auto, 2)
    'MAPE': [f"{round(MAPE_MA, 2)}%", f"{round(MAPE_SES, 2)}%", f"{round(MAPE_Holt, 2)}%", f"{round(MAPE_Holt_Winters, 2)}
    'MSE': [round(MSE_MA, 2), round(MSE_SES, 2), round(MSE_Holt, 2), round(MSE_Holt_Winters, 2), round(MSE_Sarima_auto, 2)
})

performance_table
```



	Model	MAD	MAPE	MSE
0	Moving Average	40.04	12.39%	2995.16
1	Simple Exponential Smoothing	25.72	8.97%	1131.97
2	Holt's Model	25.60	9.0%	1127.60
3	Winter's Model	7.95	3.08%	110.78
4	SARIMA (AutoARIMA)	9.88	4.51%	240.04
5	SARIMA (Manual)	9.74	4.44%	238.20

Figure 28: Performance Measure Python

Since the MAD, MAPE and MSE values are lower for SARIMA manual compared with SARIMA with autoarima function, the manual model will be used for graphical presentation and also for further analysis.

Microsoft Excel

Graphs

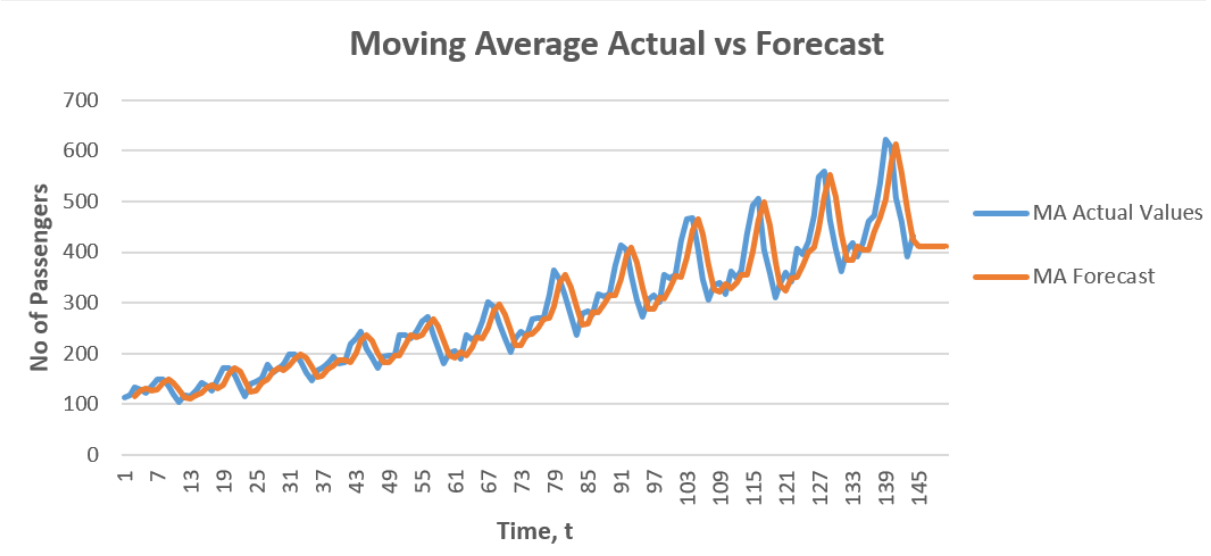


Figure 29: MA 6-month Forecast Graph Excel

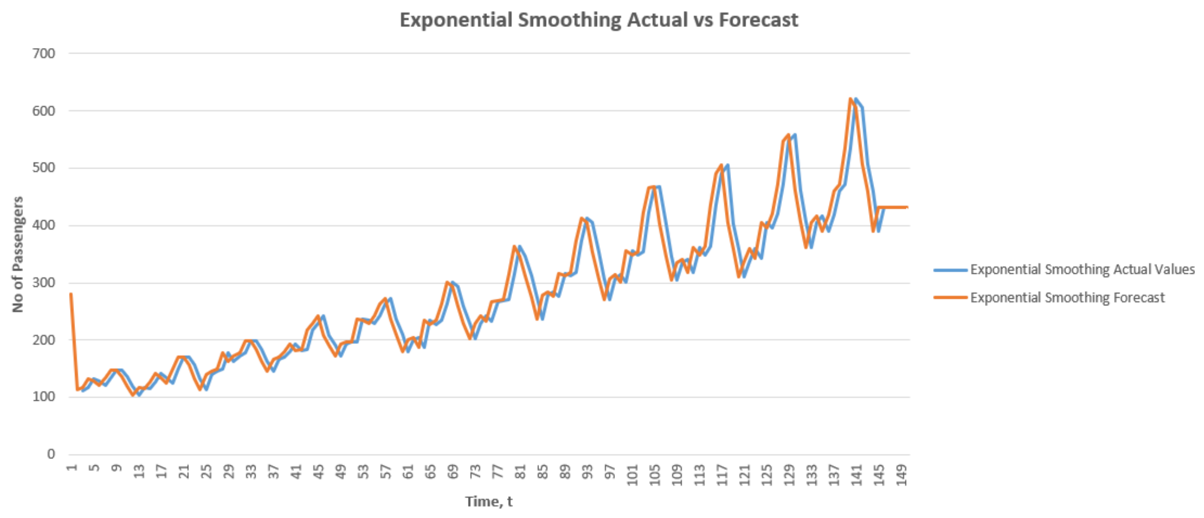


Figure 30: SES 6-month Forecast Graph Excel

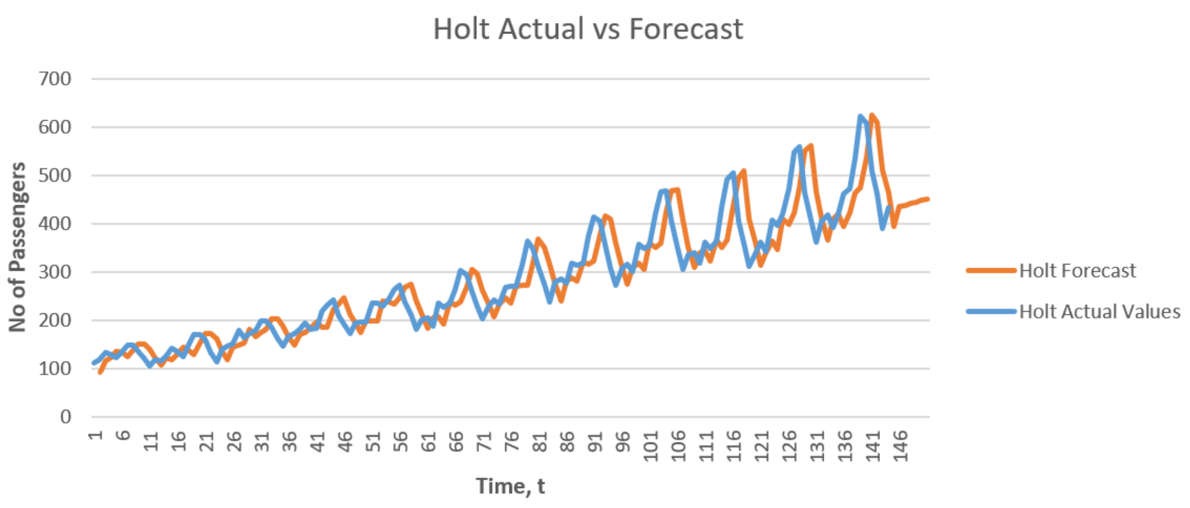


Figure 31: Holt 6-month Forecast Graph Excel

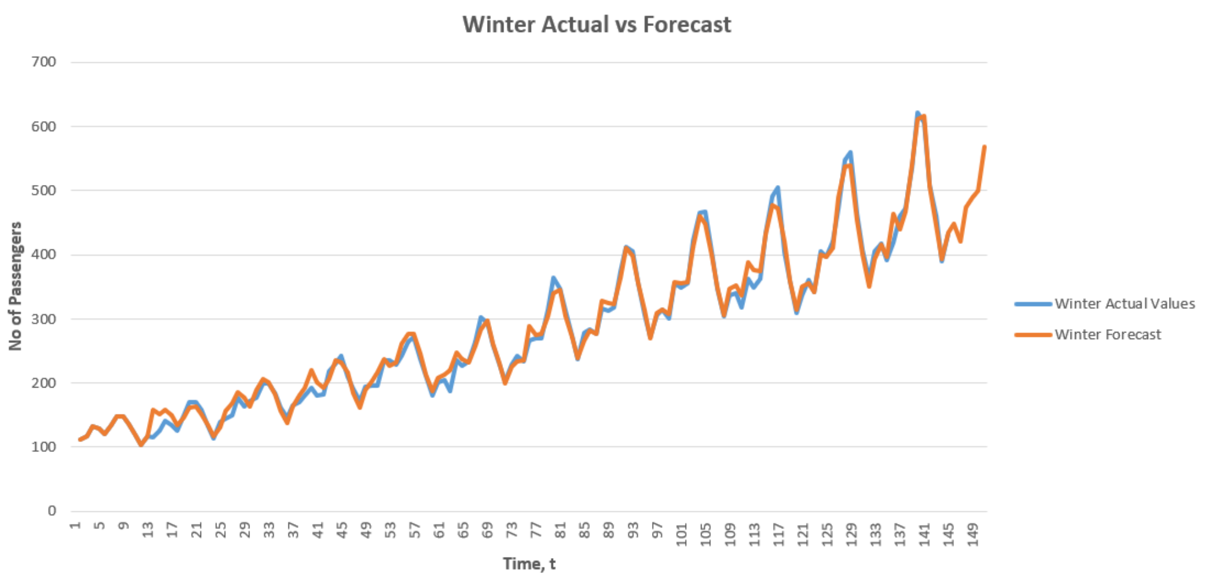


Figure 32: Winter 6-month Forecast Graph Excel

Python

Graphs

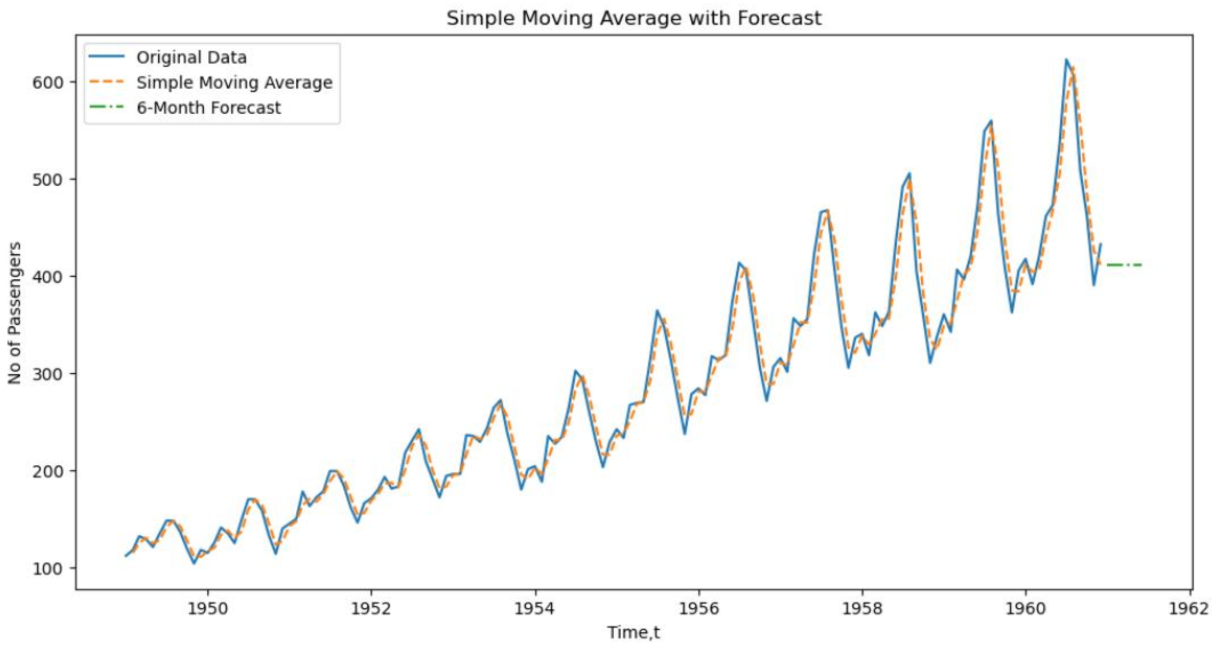


Figure 33: MA 6-month Forecast Graph Python

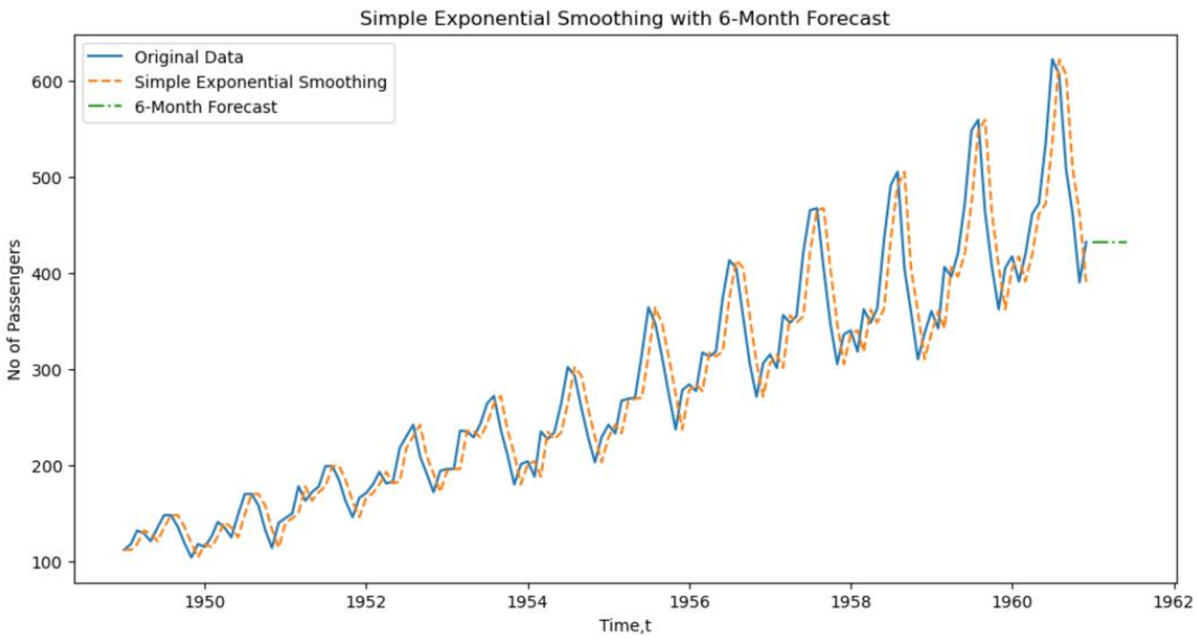


Figure 34: SES 6-month Forecast Graph Python

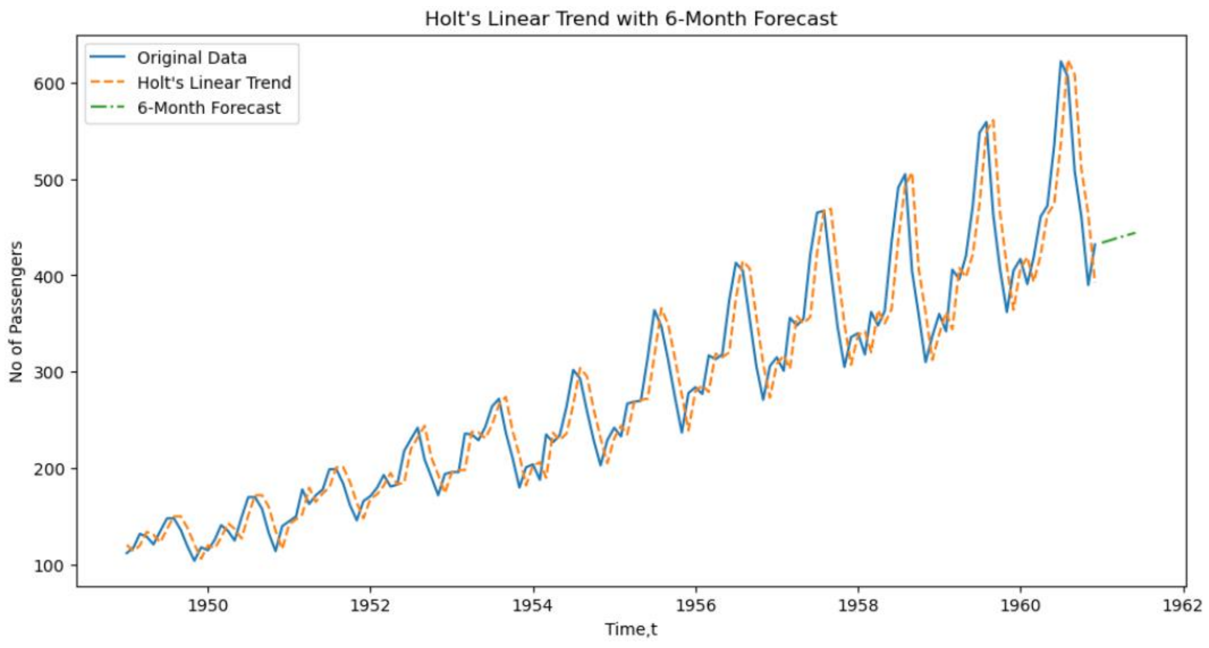


Figure 35: Holt 6-month Forecast Graph Python

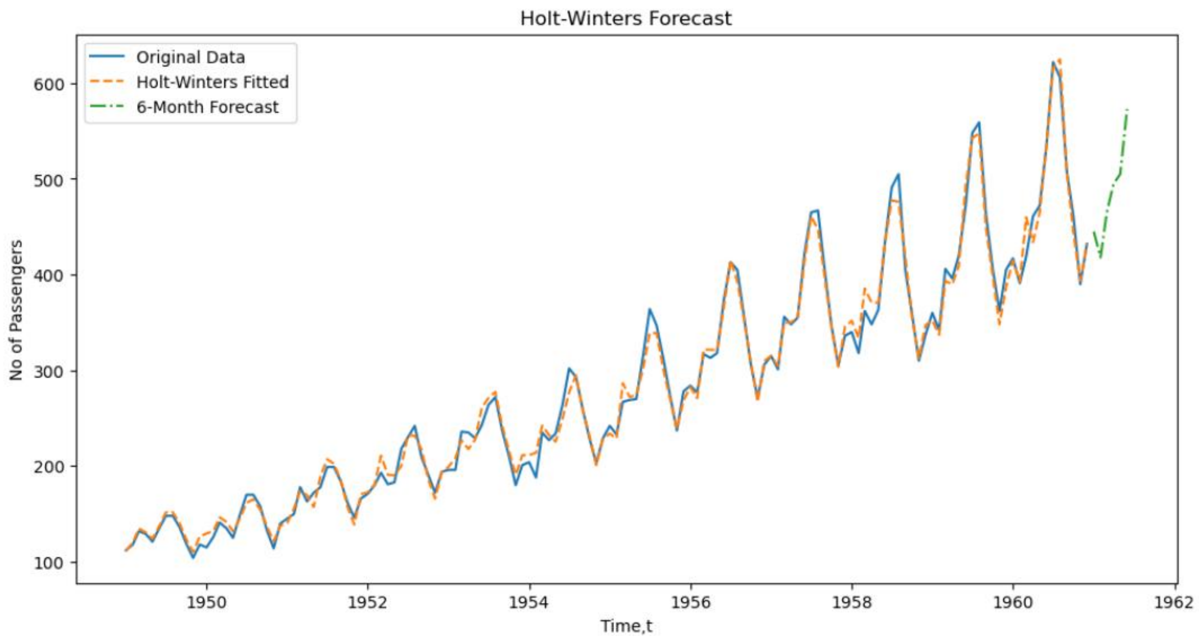


Figure 36: Winter 6-month Forecast Graph Python

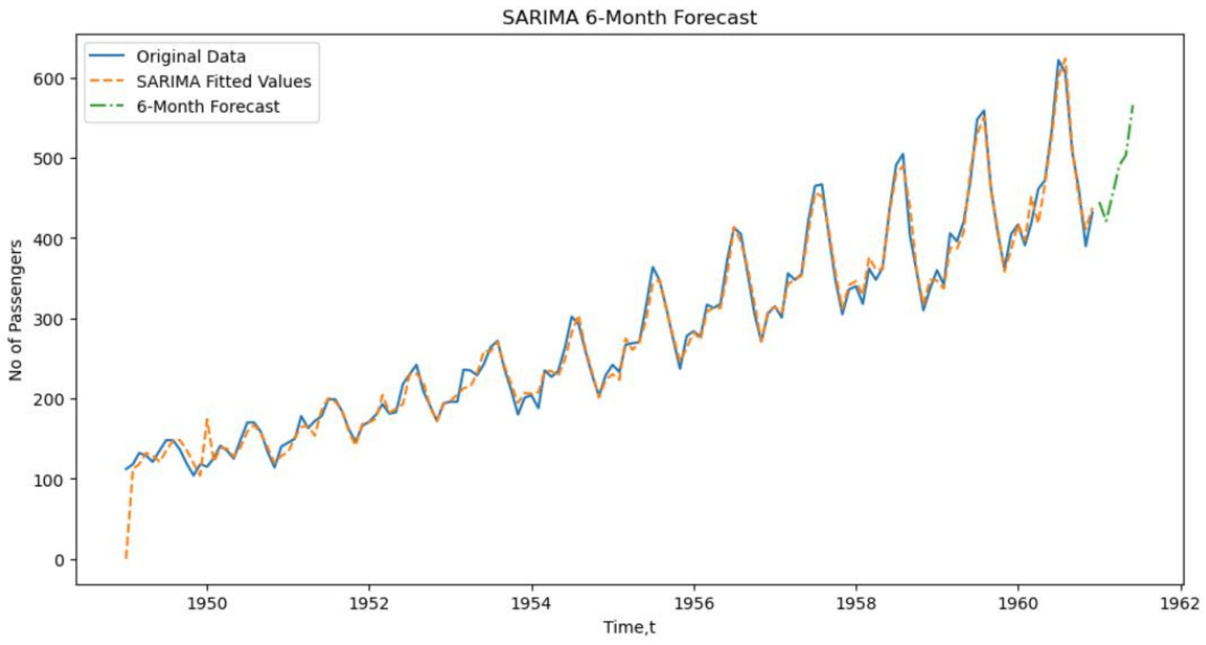
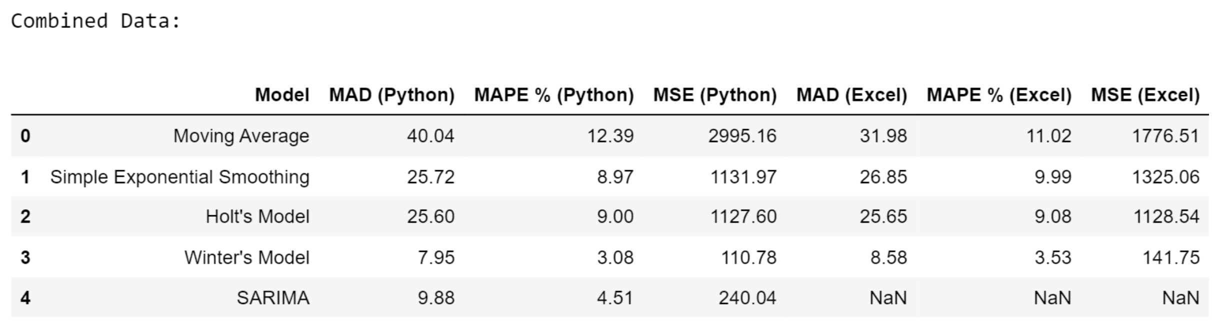


Figure 37: SARIMA 6-month Forecast Graph Python

Based on the forecasted values and graphical representation, it's apparent that the Moving Average (MA), Simple Exponential Smoothing (SES), and Holt's Model deviate from the observed seasonal pattern, unlike Winter's Model and SARIMA, which appear to align with the pattern. To validate this observation further, a comparison of the models using both Excel and Python is warranted to determine the best-performing approach across both platforms.

Comparison



```
Combined Data:
```

	Model	MAD (Python)	MAPE % (Python)	MSE (Python)	MAD (Excel)	MAPE % (Excel)	MSE (Excel)
0	Moving Average	40.04	12.39	2995.16	31.98	11.02	1776.51
1	Simple Exponential Smoothing	25.72	8.97	1131.97	26.85	9.99	1325.06
2	Holt's Model	25.60	9.00	1127.60	25.65	9.08	1128.54
3	Winter's Model	7.95	3.08	110.78	8.58	3.53	141.75
4	SARIMA	9.88	4.51	240.04	NaN	NaN	NaN

Figure 38: Performance Measure Comparison Table

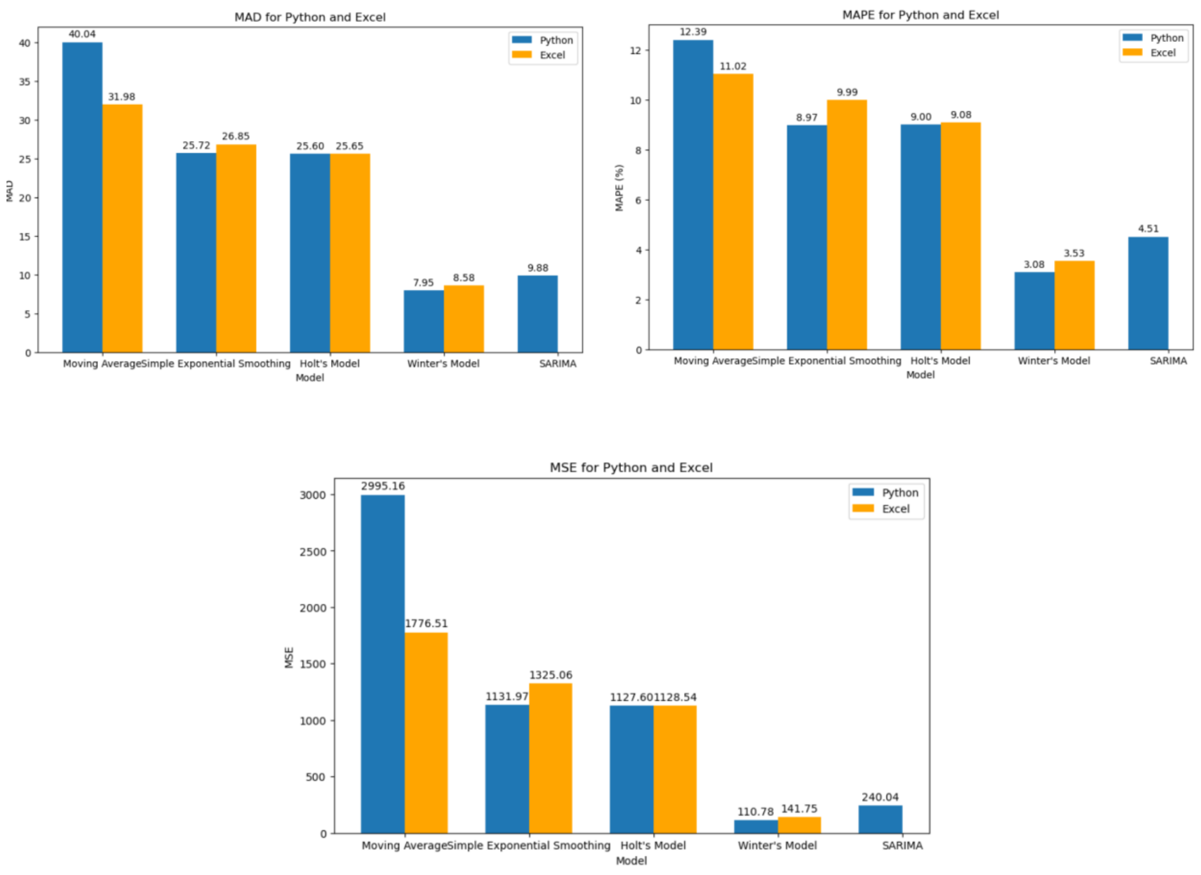


Figure 39: Performance Measure Comparison Charts

Furthermore, the comparison of performance measure values generated using both Python and Excel indicates minimal deviation between the two. Any differences observed could likely be attributed to the rounding off of parameters like L0, T0, and Ft in Excel. However, these differences are relatively insignificant, as depicted by the clustered bar chart for each performance measure. Overall, both approaches demonstrate robust performance.

Based on the comparison table and performance charts for each measure between Excel and Python, it's evident that the Winter's Model consistently outperforms other models, exhibiting the lowest MA, MAPE, and MSE values in both Excel and Python implementations.

Best Model – Winter’s Model

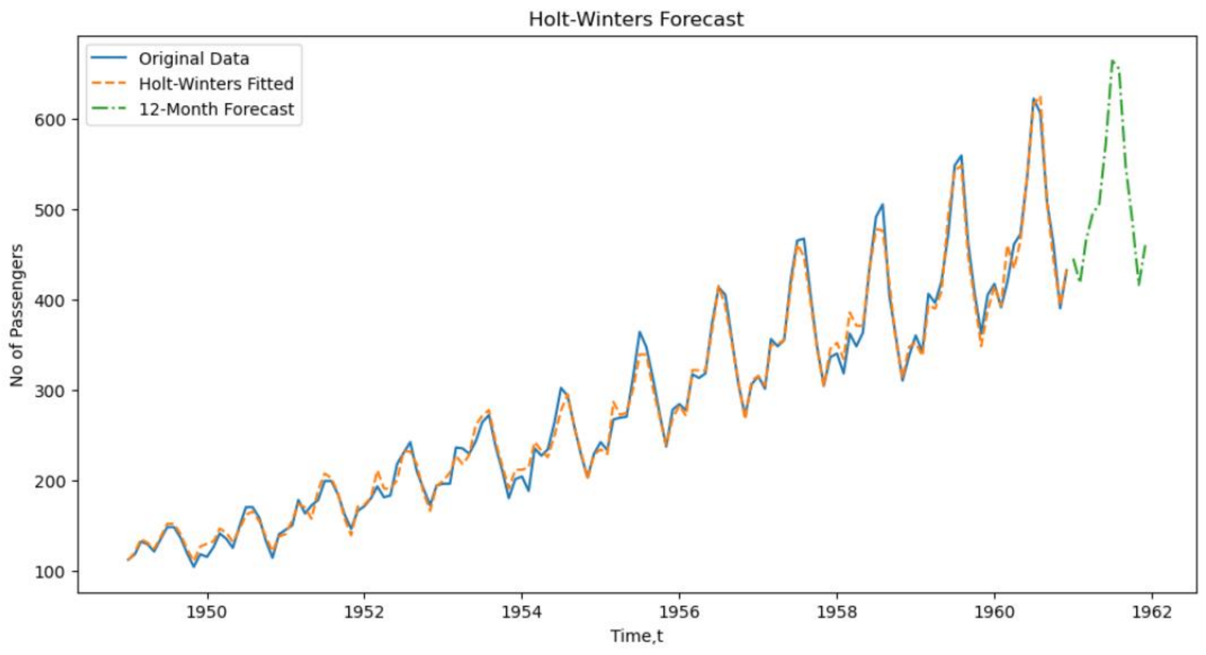


Figure 40: Best Model 12-month Forecast

Utilizing the Holts-Winter model, considered the best performing model, the airline can forecast further into the future. For instance, a one-year forecast provides insight into the expected passenger demand, reflecting patterns observed in previous years. The forecasted values for the 12-month period are as follows:

- January 1961: 445.0
- February 1961: 418.0
- March 1961: 465.0
- April 1961: 495.0
- May 1961: 505.0
- June 1961: 573.0
- July 1961: 664.0
- August 1961: 655.0
- September 1961: 547.0
- October 1961: 488.0
- November 1961: 416.0
- December 1961: 460.0

Interpretation of Forecast

Analysing the forecasted values, we discern distinct patterns across the year, reflecting seasonal dynamics and external factors shaping passenger demand.

In the initial months, typically spanning January to March, passenger numbers appear relatively subdued, possibly due to factors such as post-holiday lulls, unfavorable weather conditions like winter season, and reduced travel activity following the festive season. As we transition into spring and summer, encompassing the months of April to August, a notable uptick in forecasted values emerges. This surge can be attributed to a variety of factors, including school holidays, summer vacations, and an overall increase in leisure travel as individuals seek warmer destinations and outdoor activities.

The high point of passenger demand is often observed in July and August, coinciding with peak vacation periods and heightened tourism activity. During these months, travellers flock to popular destinations for leisure, leading to a surge in airline bookings. Subsequently, as summer draws to a close and we approach the latter half of the year, forecasted values gradually taper off. This decline may stem from factors such as the conclusion of summer vacations, back-to-school preparations, and a shift towards more routine travel patterns.

These fluctuations in forecasted values underscore the intricate interplay of various factors influencing passenger behavior and travel trends. Seasonal variations, holidays, major events, economic conditions, and promotional activities all play pivotal roles in shaping demand patterns.

Recommendations

In the recommendation section, actionable strategies and long-term approaches are outlined to capitalize on forecasted demand fluctuations and optimize operational performance. These recommendations are tailored to guide airlines in effectively managing both low and high demand periods, fostering sustainable growth, and enhancing the overall customer experience.

During periods of low demand:

- Offer discounts and targeted promotions: Airlines can implement promotional offers, such as discounted fares, package deals, or loyalty rewards, to incentivize travel during slower periods. By offering compelling incentives, such as discounted tickets or bonus miles, airlines can attract price-sensitive travellers and stimulate demand.

- Introduce seasonal campaigns: Launch targeted marketing campaigns highlighting seasonal destinations, events, or activities. By showcasing appealing travel experiences and exclusive offers, airlines can capture the interest of potential travellers and drive bookings during low-demand periods.

- Enhance flexibility: Provide flexible booking options, including waived change fees or flexible travel dates, to accommodate travellers' changing plans and preferences. By offering flexibility, airlines can reassure customers and encourage them to book flights even during uncertain times, thereby boosting demand.

- Partner with tourism boards or local businesses: Collaborate with tourism boards, hotels, or attractions to create package deals or joint promotions. By bundling airfare with accommodations, tours, or activities, airlines can offer comprehensive travel packages that appeal to a wider audience and drive bookings.

- Optimize pricing strategies: Analyze demand trends and adjust pricing dynamically to match supply and demand. Utilize revenue management techniques to offer competitive fares while maximizing revenue opportunities during low-demand periods.

During periods of high demand:

- Optimize pricing and revenue management: Implement dynamic pricing strategies to maximize revenue during peak periods. Utilize demand forecasting and revenue management tools to adjust fares in real-time based on demand fluctuations, competitor pricing, and booking trends. By optimizing pricing, airlines can capture the value of increased demand while maximizing profitability.

- Expand capacity and frequency: Increase flight frequencies or deploy larger aircraft to accommodate higher demand during peak seasons or busy travel periods. By expanding capacity, airlines can meet the needs of travelers and capture additional market share, thereby maximizing revenue potential.

- Enhance customer experience: Invest in superior customer service and amenities to differentiate the airline's offering and attract premium travelers. Provide personalized services, such as priority boarding, lounge access, or onboard amenities, to enhance the overall travel experience and encourage repeat business.

- Offer ancillary services: Promote ancillary revenue streams, such as seat upgrades, premium services, or in-flight amenities, to generate additional revenue during high-demand periods. By offering optional services and upgrades, airlines can enhance customer value and increase overall profitability.

- Collaborate with travel partners: Partner with hotels, rental car companies, and other travel providers to offer bundled packages or joint promotions. By collaborating with strategic partners, airlines can create comprehensive travel experiences and capture a larger share of the travel market.

Long-term strategies:

- Invest in fleet modernization: Upgrade the fleet with newer, fuel-efficient aircraft to improve operational efficiency, reduce operating costs, and enhance the passenger experience. Investing in modern aircraft technologies can position the airline for long-term sustainability and competitive advantage.

- Expand route network: Identify emerging markets or underserved routes for expansion to capture new customer segments and revenue opportunities. By expanding the route network strategically, airlines can increase market penetration and diversify revenue streams.

- Embrace digital transformation: Invest in digital technologies and e-commerce platforms to streamline operations, enhance customer engagement, and drive online bookings. By leveraging data analytics, artificial intelligence, and mobile applications, airlines can personalize the travel experience, optimize marketing efforts, and improve overall operational efficiency.

- Focus on sustainability: Implement environmentally friendly initiatives and sustainable practices to reduce carbon emissions and minimize the environmental impact of operations. By adopting sustainable aviation practices, airlines can enhance their corporate social responsibility profile, attract eco-conscious travellers, and contribute to a greener future.

- Cultivate customer loyalty: Develop robust loyalty programs and customer retention strategies to foster long-term relationships with frequent flyers and loyal customers. By offering rewards, incentives, and personalized experiences, airlines can cultivate customer loyalty, drive repeat business, and strengthen brand advocacy.

By implementing these long-term strategies, airlines can adapt to changing market dynamics, capitalize on growth opportunities, and position themselves for sustainable success in the competitive aviation industry.

Conclusion

In conclusion, this project embarked on a comprehensive journey to explore time series forecasting techniques and their application in the airline industry. By leveraging both Excel and Python, various models including Moving Average, Simple Exponential Smoothing, Holt's Model, Winter's Model, and SARIMA were implemented and evaluated. Through meticulous analysis of performance metrics such as Mean Absolute Deviation (MAD), Mean Absolute Percentage Error (MAPE), and Mean Squared Error (MSE), insights were gleaned regarding the effectiveness of each model in predicting passenger demand.

The comparison between Excel and Python implementations provided valuable insights into the consistency and accuracy of forecasting results across different platforms. Despite minor discrepancies attributed to rounding-off and implementation nuances, the overall agreement between the two approaches underscores the robustness of the forecasting methodologies employed.

Among the models evaluated, Winter's Model emerged as the most promising performer, exhibiting superior forecasting accuracy across various performance metrics. Its ability to capture seasonal patterns and adapt to changing demand dynamics positions it as a viable choice for airlines seeking reliable forecasting solutions.

Moreover, the interpretation of forecasted values revealed nuanced insights into demand fluctuations throughout the year. Strategies were proposed to capitalize on low-demand periods through targeted promotions and discounts, while high-demand periods present opportunities for revenue optimization and service enhancement. Long-term strategies, including route expansion and customer-centric initiatives, were also recommended to sustain growth and competitiveness in the airline industry.

In essence, this project underscores the importance of robust forecasting methodologies in driving informed decision-making and operational efficiency within the airline industry. By harnessing the power of data analytics and predictive modelling, airlines can proactively respond to market dynamics, optimize resource allocation, and ultimately enhance the overall passenger experience.

References

AltexSoft. (2024, February 29). *Airline Marketing and advertising strategies and use cases*. https://www.altexsoft.com/blog/airline-marketing-advertising/

RakanNimer. (2017, March 29). *Air passengers*. Kaggle. https://www.kaggle.com/datasets/rakannimer/air-passengers