**Faculty of Natural Sciences**

**School of Computer Science and Mathematics**

**Collaborative Application Development**

**COURSEWORK**

**Group 3 Members**

| **No.** | **Student Number** | **Student Name** |
| --- | --- | --- |
| 1 |  | Lucy |
| 2 |  | Kelvin |
| 3 |  | Julie |
| 4 |  | Nivedita |
| 5 |  | Umer |
| 6 |  | Awais |

**INTRODUCTION**

In the first week, we met with our client, Gerald. Gerald runs an event organizing company that sells conference tickets for various events. Their main challenge is accurately predicting the final number of ticket sales. Gerald gave us 7 datasets from previous events to use for our prediction models. He mentioned that there's no set target for ticket sales and that some datasets are from 2020, during the COVID-19 pandemic.

After a Q&A session, our group of 6 members began working on the task of creating an event ticket sales forecasting application.

Together, we decided to:

1. Preprocess the data.
2. Select and test prediction/forecasting libraries.
3. Use RMSE or MAPE to choose the best library for the prediction application.
4. Build the application.

In order to evaluate the precision of our predictive models and forecasting task, we computed multiple statistical metrics:

* The average size of errors between our anticipated values and the actual results is provided by the Mean Absolute Error (MAE) measure. It is helpful in determining the average distance between our forecasts and the true values.
* Similar to MAE, Root Mean Squared Error (RMSE) takes into account the square of the deviations between the expected and actual values. This provides a measure of the spread of errors among our forecasts by assigning greater weight to larger errors.
* The average absolute % difference between the expected and actual values is measured by the Mean Absolute % Error, or MAPE. It aids in our comprehension of how accurate our forecasts are in comparison to the actual values, especially when dealing with forecasts when the percentage error scale is significant.

These metrics aid in measuring our models' performance. Reduced MAE, RMSE, and MAPE values signify improved precision and a closer match between our forecasts and the observed results.

Likewise, TensorFlow, an influential open-source library aimed at the facilitation of machine learning processes, provides remarkable benefits encompassing robust support tailored for deep learning paradigms, scalability conducive to expansive datasets, and an all-encompassing ecosystem which includes utilities for the construction, training, and deployment of models. Its accommodation to diverse platforms and programming languages augments its flexibility and integrative potential.

On the contrary, it exhibits significant complexity and is associated with a steep learning gradient, necessitating substantial computational resources and a high degree of expertise to optimise its performance.

In relation to this project, the utilisation of TensorFlow was dumped on account of the inadequacy in the magnitude of the provided dataset to fully exploit TensorFlow’s capabilities, thereby rendering simpler and less resource-demanding tools more suitable for specified tasks.

**Data Pre-processing - Nivedita**

Data preprocessing is a fundamental step in data analysis and machine learning that involves transforming raw data into a format that is clean, consistent, and suitable for analysis. This step is critical because the quality of the data directly impacts the effectiveness of the analysis and the accuracy of predictive models. In this report, we focus on the pre-processing steps applied to attendance data collected over several years across different datasets.

**Summary of the raw data:**

| **Dataset** | **Columns** | **Rows** | **Important Information** |
| --- | --- | --- | --- |
| D19  (Target Audience- IT Managers) | **5** | 1185 | 1. Details of the year of 2019  2. Details of ‘Attended’ column:  Yes- 839  No- 247  NaN- 99  3. Details of ‘Attendee Status’:  Attending- 1082  Cancelled- 90  Booker not attending- 13 |
| D21  (Target Audience- IT Managers) | **5** | 671 | Details of the year of 2020 and 2021  2. Details of ‘Attended’ column:  Yes- 340  NaN- 331  3. Details of ‘Attendee Status’:  Attending- 669  Cancelled- 2 |
| GP21  (Target Audience- Property Managers | **5** | 798 | Details of the year of 2020 and 2021  2. Details of ‘Attended’ column:  NaN- 798  3. Details of ‘Attendee Status’:  Attending- 767  Cancelled- 31 |
| MSE21  (Target Audience-  Education Property Managers | **5** | 1601 | Details of the year of 2021  2. Details of ‘Attended’ column:  NaN- 1601  3. Details of ‘Attendee Status’:  Attending- 1582  Cancelled- 19 |
| NP21  (Target Audience- Property Managers | **5** | 401 | Details of the year of 2020 and 2021  2. Details of ‘Attended’ column:  Yes- 280  NaN- 93  No- 28  3. Details of ‘Attendee Status’:  Attending- 401 |
| SRM22  (Target Audience- Education Managers | **5** | 998 | Details of the year of 2022  2. Details of ‘Attended’ column:  NaN- 722  Yes- 196  No- 80  3. Details of ‘Attendee Status’:  Attending- 998 |
| SRM23  (Target Audience- Education Managers | **5** | 753 | Details of the year of 2023  2. Details of ‘Attended’ column:  NaN- 405  Yes- 300  No- 48  3. Details of ‘Attendee Status’:  Attending- 741  Cancelled- 12 |

**Overview of Data Pre-Processing Steps**

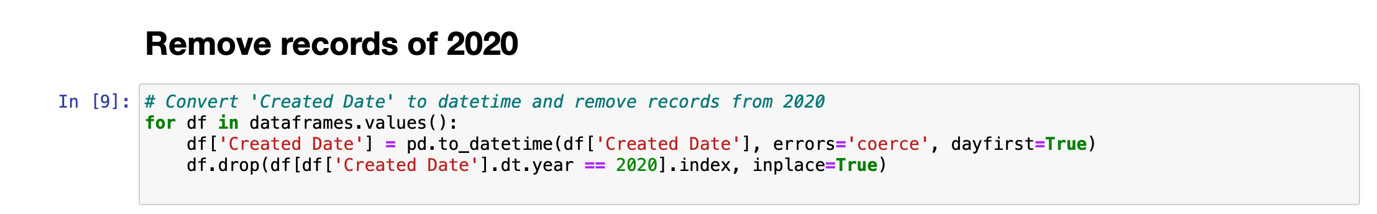
Data pre-processing is essential for optimising the accuracy and efficiency of data analysis and machine learning models. In this project, we implemented several key pre-processing steps

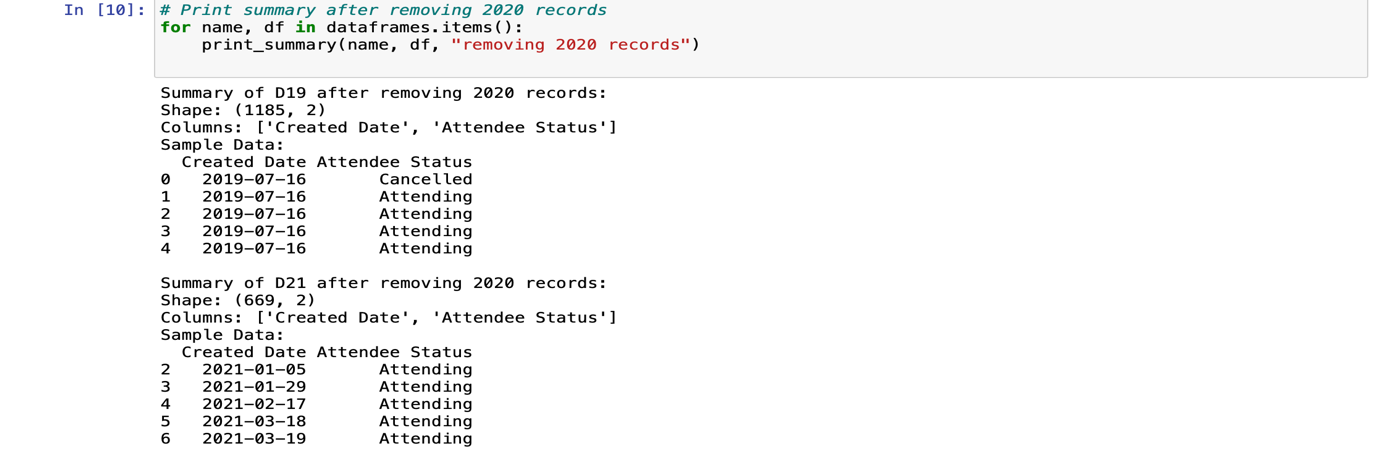
**Removing Unnecessary Columns:**

* **Purpose:** Simplifies the dataset by eliminating irrelevant features that do not contribute to the analysis or predictive modelling. This reduces the complexity and computational load during processing.
* **Impact:** Streamlining the data helps focus the model on relevant features, which can improve performance and reduce the risk of overfitting.

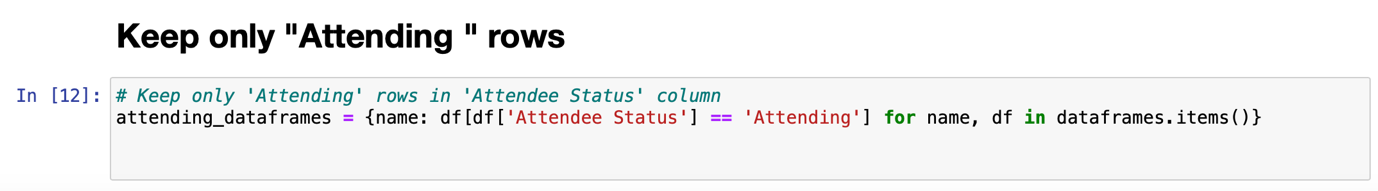
After the second week meeting with client, we decided to remove-

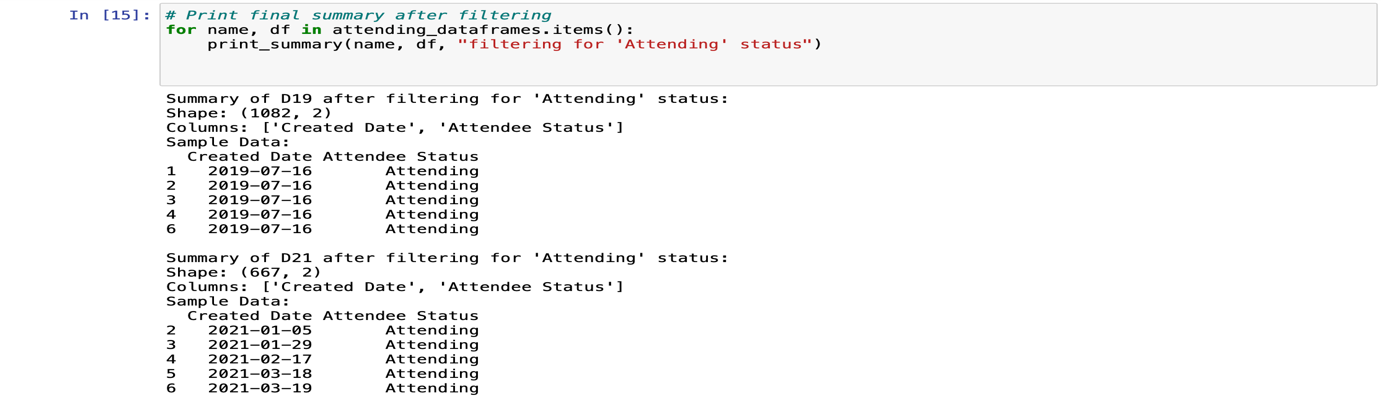
* *2020 records:* This was the period when pandemic was going on, which might have affected the predictive model indirectly.





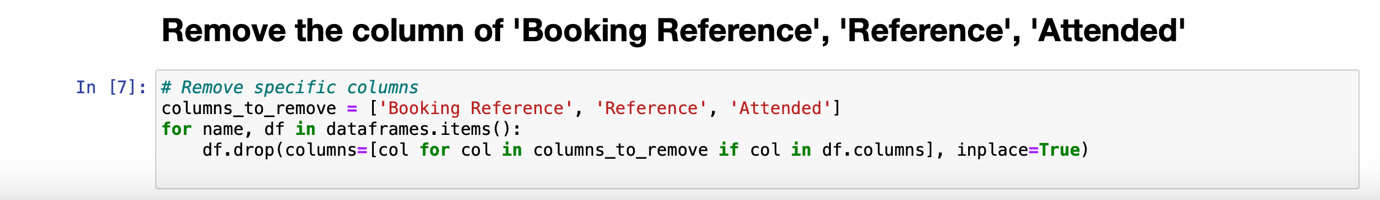
* *‘Attended’ column:* In this column almost 4049 entries out of 6407 fall into the category of ‘NaN’, which stands for ‘Not a Number’ and is typically used in datasets to denote missing or undefined values.

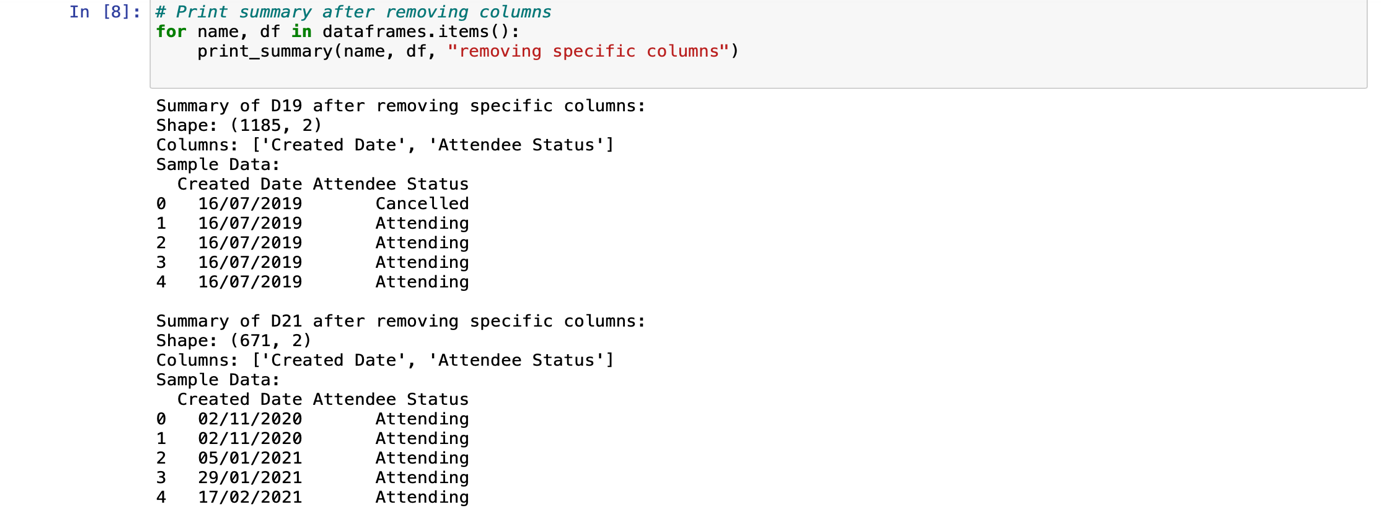




The presence of a total of 403 ‘No’ entries could be of interest depending on the context of the event. For example, if attendance is crucial, these ‘No’ entries could warrant further investigation or follow-up. Based on the given details, approximately 63% data related to ‘Attended’ column are missing, 6% of the total entries fall into the section of ‘No’. So, we decided to delete the column instead of doing the imputation. There are several ways to deal with missing values, like- Mean/Median/Mode Imputation, K-Nearest Neighbours (KNN), Interpolation. This method is generally not recommended when the missing data proportion is this high, as it artificially inflates the sample size with potentially misleading values. When almost 63% of data are missing from a dataset, handling this significant amount of missing data becomes especially challenging, and the methods used can have a substantial impact on the reliability and validity of the analysis.

* *‘Attendee Status’ column:* In this column we only focused on the ‘Attending’ status. After discussing with the client, we realised that the main goal of this project is how many people are going to attend the conference instead of cancelling the booking. To make a robust prediction model we decided to delete the entries other than ‘Attending’.
* *‘Booking Reference’ and ‘Reference’ column:* In this project, these two columns have no significance with the prediction. Due to this reason, we cleaned the entries related to these columns.





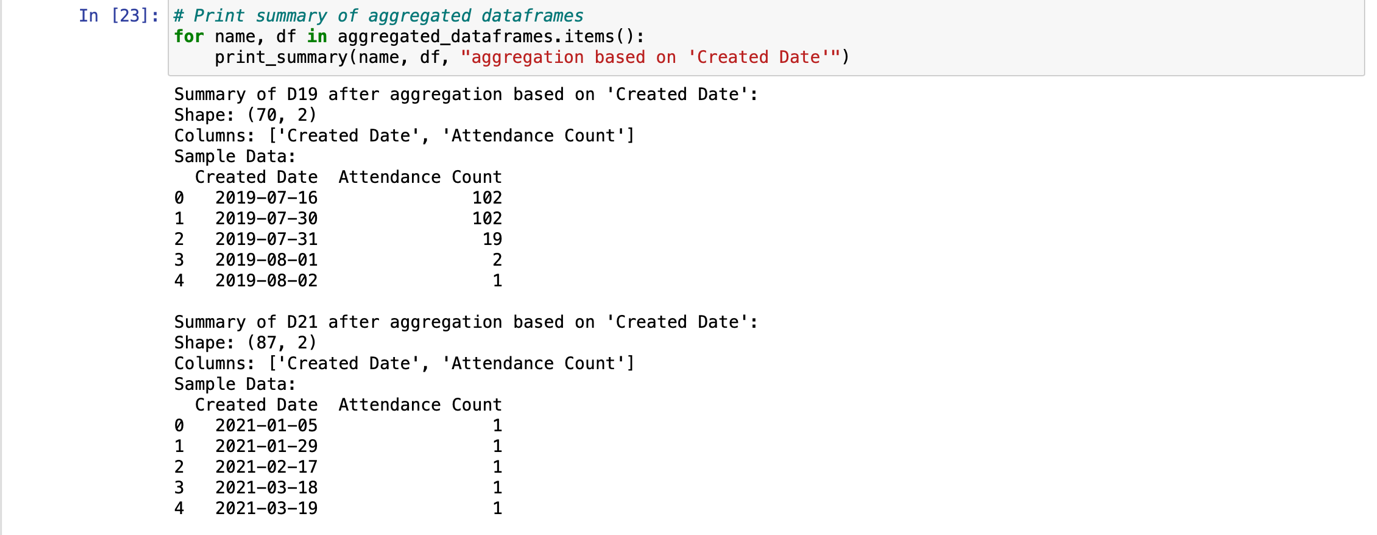
**2. Data Aggregation:**

We aggregated the data by dates and counting occurrences, which is a common method in data analysis to simplify and summarise data, making it easier to observe trends and patterns over time. This transformation is particularly useful for time series forecasting or any analysis where the frequency of events within specific time frames is relevant. It helps in understanding the distribution and trends of data points across different dates.

* **Impact:** Aggregation can either enhance or diminish the accuracy of insights drawn from the data. While it can clarify trends, over-aggregation can lead to loss of critical details. The key is to find the right level of aggregation that balances detail with clarity.

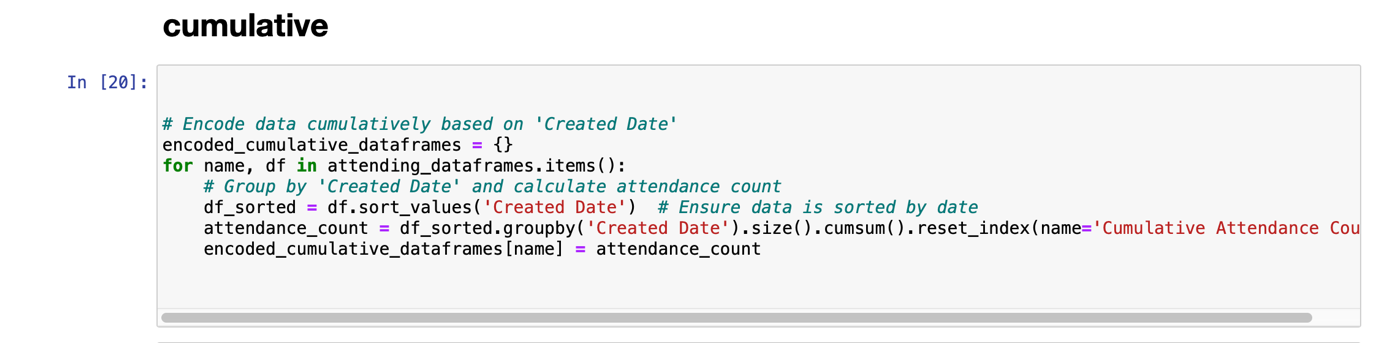


The function aims to aggregate attendance data by counting the number of occurrences for each 'Created Date' in multiple data frames and store the aggregated results in new dataframes.



# *Cumulative:*

The function aims to aggregate attendance data by calculating the cumulative count of occurrences for each 'Created Date' in multiple data frames and store the cumulative results in new dataframes.



By applying these pre-processing steps, we ensure that the data is clean, structured, and ready for further analysis or machine learning modelling. These steps help in extracting meaningful insights and building robust predictive models.

After cleaning the data, the next step was to find the best predictive models for our data. Since our data includes only the date and value of ticket sales, it's considered time series forecasting. We decided to use three libraries to undertake the task; ARIMA, XGBoost, and Prophet then selected the best among the three.

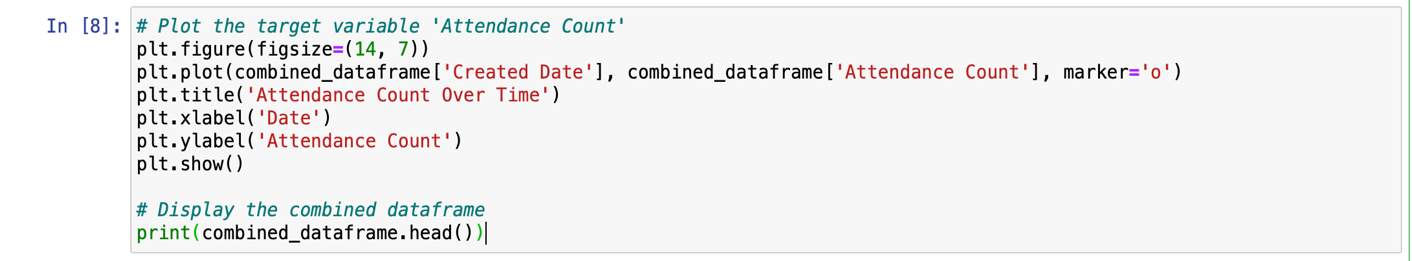
**EXPLORING ARIMA FOR FORECASTING**

In the realm of time series forecasting, the AutoRegressive Integrated Moving Average (ARIMA) model stands out due to its ability to capture the intricate patterns in sequential data. ARIMA's flexibility allows it to handle non-stationarity through differencing, while its autoregressive (AR) and moving average (MA) components model the dependencies between observations. Given the notable fluctuations and potential seasonal effects in the dataset, ARIMA, and its seasonal variant SARIMA, are explored to provide accurate forecasts for the task in-hand.

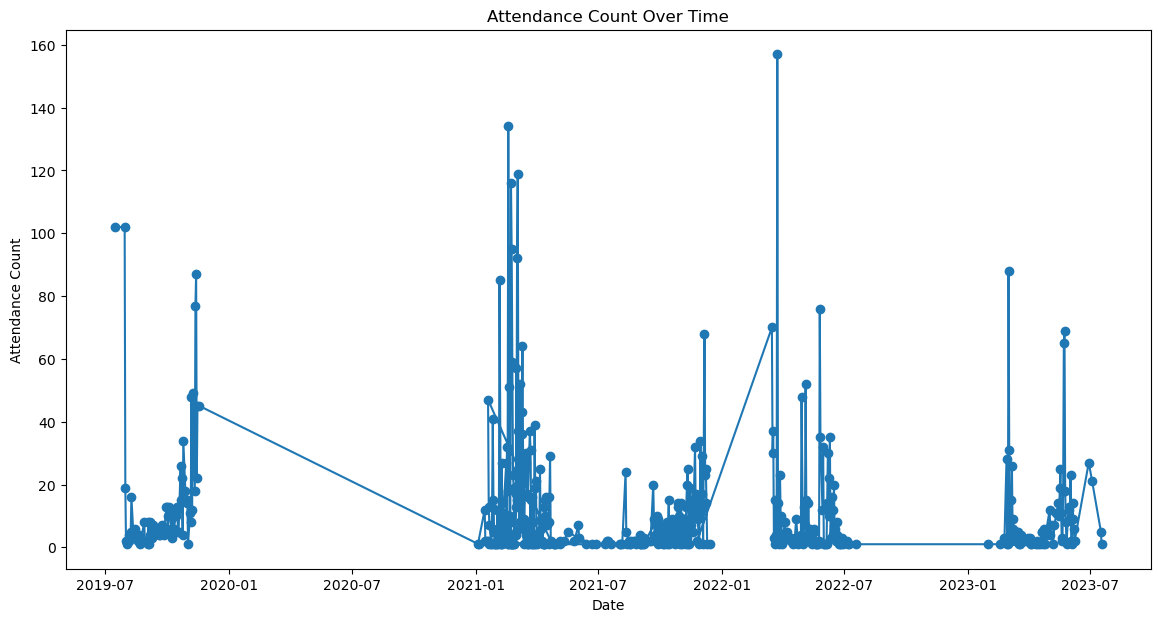
**Rationale behind the use of ARIMA related to this project-** Using ARIMA to predict conference attendance is highly relevant to this project goal, given its strengths in time series forecasting. Because ARIMA's ability to capture patterns in historical attendance data can provide accurate forecasts, essential for planning and resource allocation.

*Visualise the data:*

The goal of this project is to predict conference attendance, and visualising the data is a crucial step in achieving this goal.



Visualising the attendance count over time is crucial for understanding trends and patterns, as it allows us to easily identify seasonal effects, anomalies, and overall trends in the data. This understanding is essential for selecting appropriate forecasting models. The plot helps determine whether attendance is increasing, decreasing, or remaining constant, informing future strategies and decisions. Additionally, visualising the data enables a preliminary assessment of stationarity, which is necessary for ARIMA modelling, and helps identify recurring patterns or seasonal effects that might guide the inclusion of seasonal components in the model or suggest the need for alternative models.

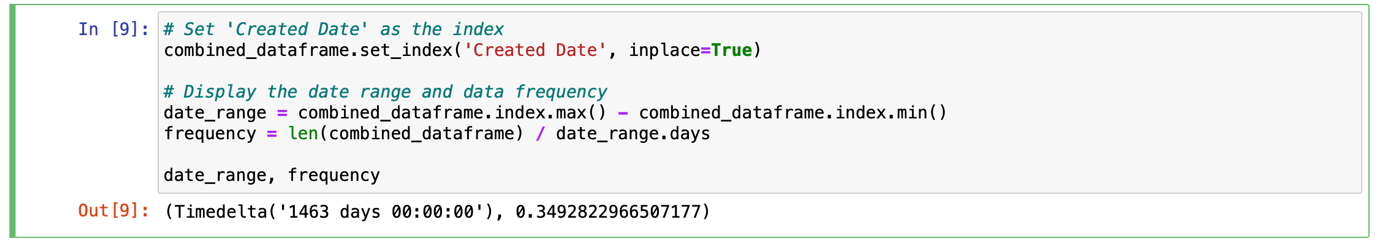


*Explanation:* The graph shows significant fluctuations in attendance over the years, with some periods of high attendance followed by periods of very low attendance. There are noticeable gaps particularly around the end of 2020 and beginning of 2021. This is the indicative of external factors affecting attendance (e.g., the impact of COVID-19).

Several spikes exceed 100 in attendance, particularly in 2021 and 2022. These could represent special events or anomalies where attendance was unusually high. The drop to nearly zero attendance around the end of 2020 and early 2021 is significant and may warrant further investigation.

*'Created Date' as index:*

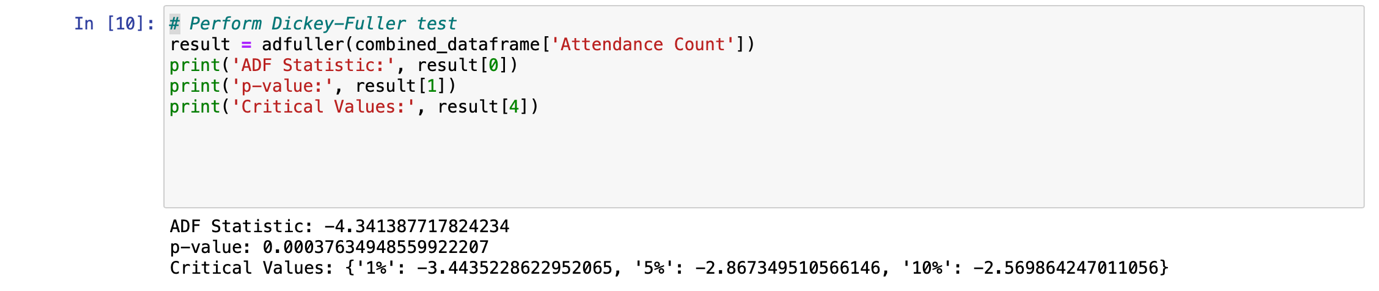
Setting the 'Created Date' as the index is a standard preprocessing step for time series data, enabling time-based operations and analysis. Calculating the date range ensures that the data covers a sufficient period for meaningful analysis and forecasting. Knowing the date range and data frequency assists in selecting the right parameters for the ARIMA model and in determining if any preprocessing, such as resampling or interpolation, is needed.



It indicates that the dataset spans 1463 days (about 4 years) and has an average frequency of approximately 0.35 data points per day, meaning there is roughly one data point every three days.

*Dickey-Fuller test, check the p-value:*

ARIMA models require the time series data to be stationary. Stationarity means that the statistical properties of the series (mean, variance, autocorrelation, etc.) are constant over time. Non-stationary data can lead to unreliable and inaccurate model predictions. Non-stationarity can manifest as trends, seasonality, or other time-dependent structures in the data. To test for this, the Dickey-Fuller test is used.

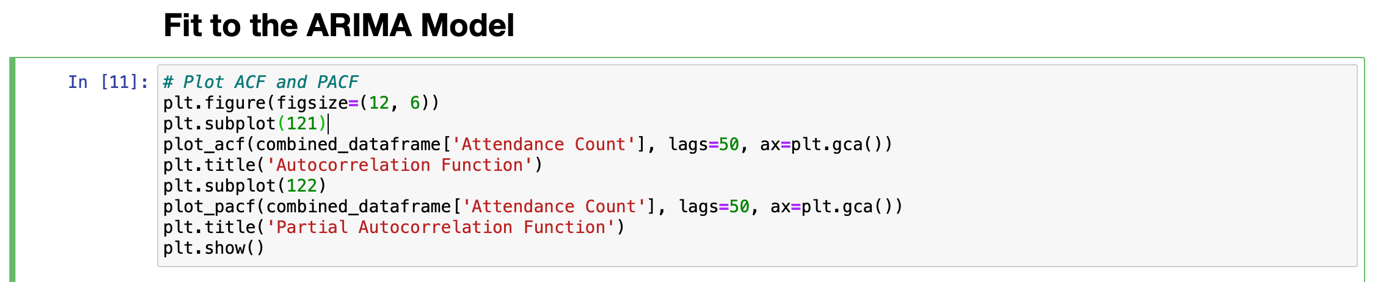


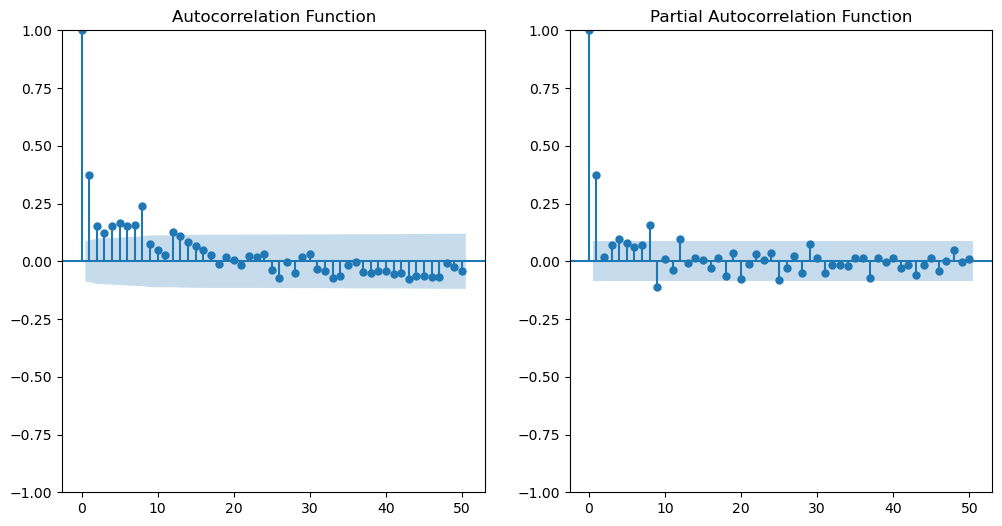
Since the ADF Statistic is less than all the critical values at the 1%, 5%, and 10% significance levels, and the p-value is less than 0.05, we reject the null hypothesis. This means:

The time series is stationary. No differencing is needed for the ARIMA model. Given that the time series is already stationary, we can proceed with fitting an ARIMA model without applying differencing.

*ACF and PACF:*

The ACF and PACF plots provide essential insights into data patterns by revealing underlying trends and seasonality, which are crucial for configuring the ARIMA model correctly. They can also be used after fitting the ARIMA model to check the residuals' ACF and PACF, helping diagnose if the model has captured the data's structure adequately. By offering a visual representation of autocorrelations, these plots aid in making informed decisions about the model's parameters, ultimately leading to more accurate and reliable forecasts.





ACF (Autocorrelation Function):

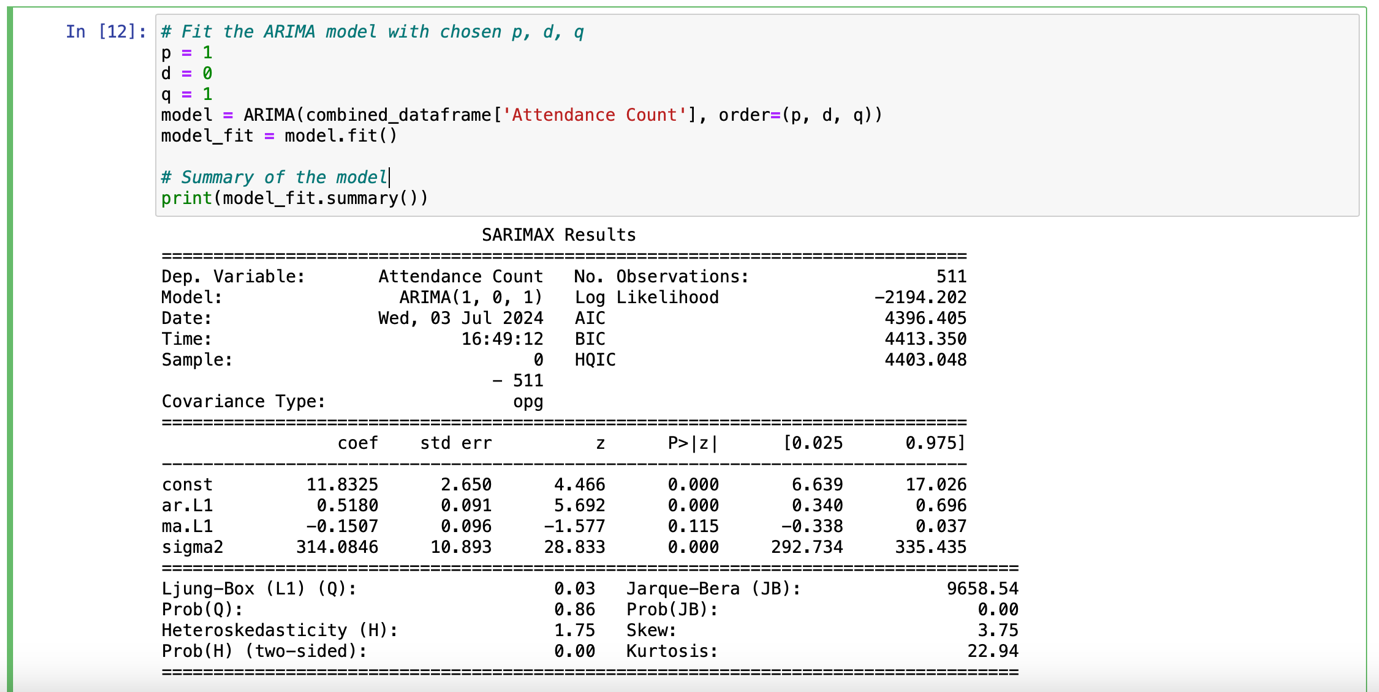
The ACF plot shows the correlation of the time series with its own lagged values. If there is a sharp drop after lag 1, it indicates that the series is stationary and has a short memory of past values. From the ACF plot, it looks like the ACF value drops significantly after lag 1 and gradually decreases, which suggests a potential MA (Moving Average) component.

PACF (Partial Autocorrelation Function):

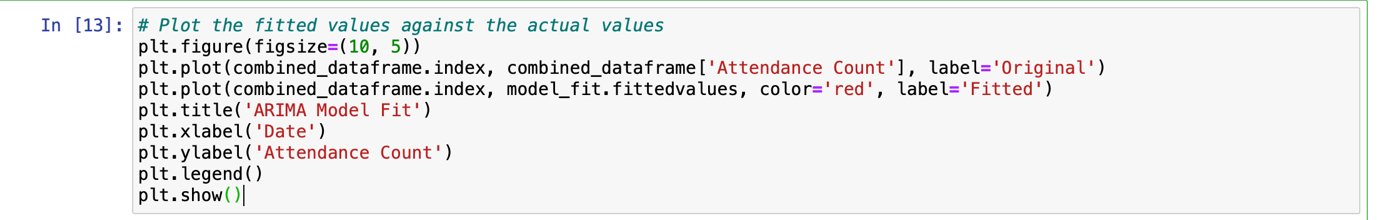
The PACF plot shows the partial correlation of the time series with its own lagged values, after removing the effects of intermediate lags. A sharp drop after lag 1 in the PACF plot suggests that the AR (AutoRegressive) process is of order 1. From the PACF plot, the partial autocorrelation drops after lag 1, suggesting an AR component.

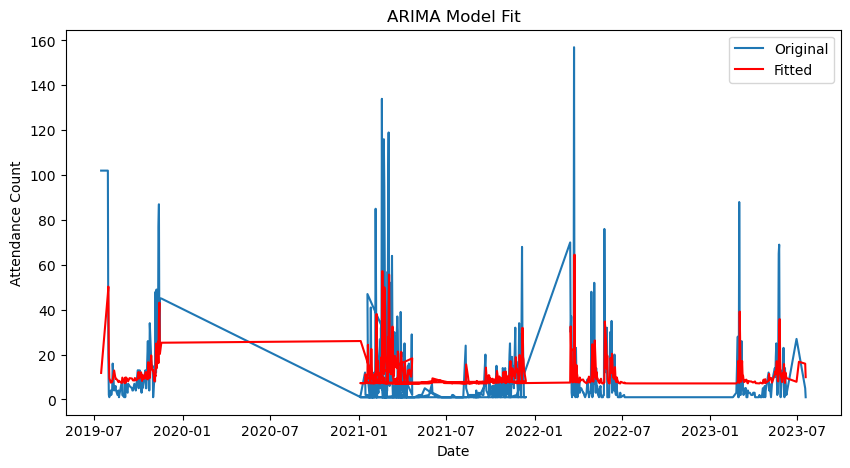
*Output of the ACF and PACF:*

p=1 and q=1 for the ARIMA model. Since we have already determined the series is stationary, d=0.



The output indicates that the ARIMA(1, 0, 1) model fits the 'Attendance Count' data, with the autoregressive term being significant, while the moving average term is not. The diagnostics suggest no significant autocorrelation in the residuals but highlight issues with normality and heteroskedasticity, which might affect the model's predictive performance. Further investigation and possibly model adjustments are needed to address these issues for improved forecasting accuracy.



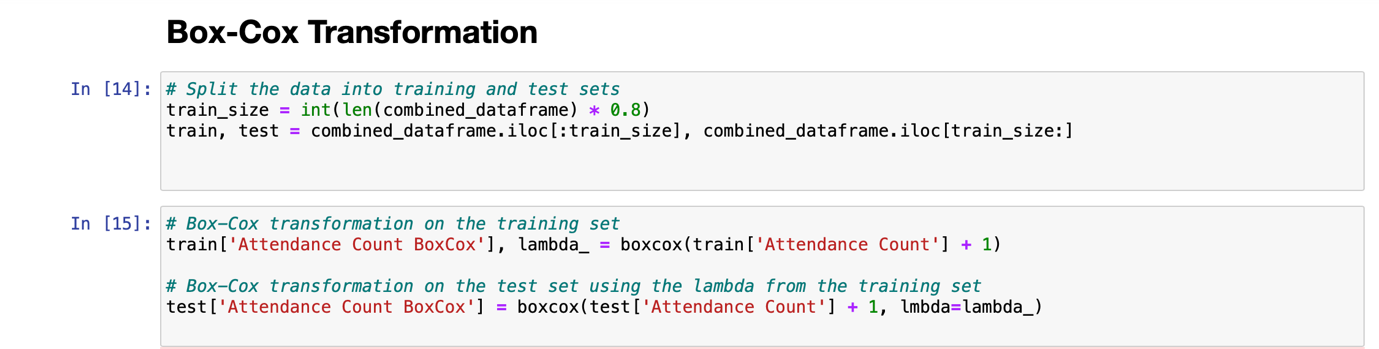


X-Axis (Date) The x-axis represents the timeline from July 2019 to July 2023. It shows the progression of time over the span of four years. Y-Axis (Attendance Count) The y-axis represents the attendance count, which ranges from 0 to 160. It measures the number of attendees recorded on each date. Data Lines Blue Line (Original): This line represents the original attendance count data. It shows the actual recorded attendance over time. Red Line (Fitted): This line represents the values predicted by the ARIMA(1,0,1) model. It shows the model's fit to the actual data.

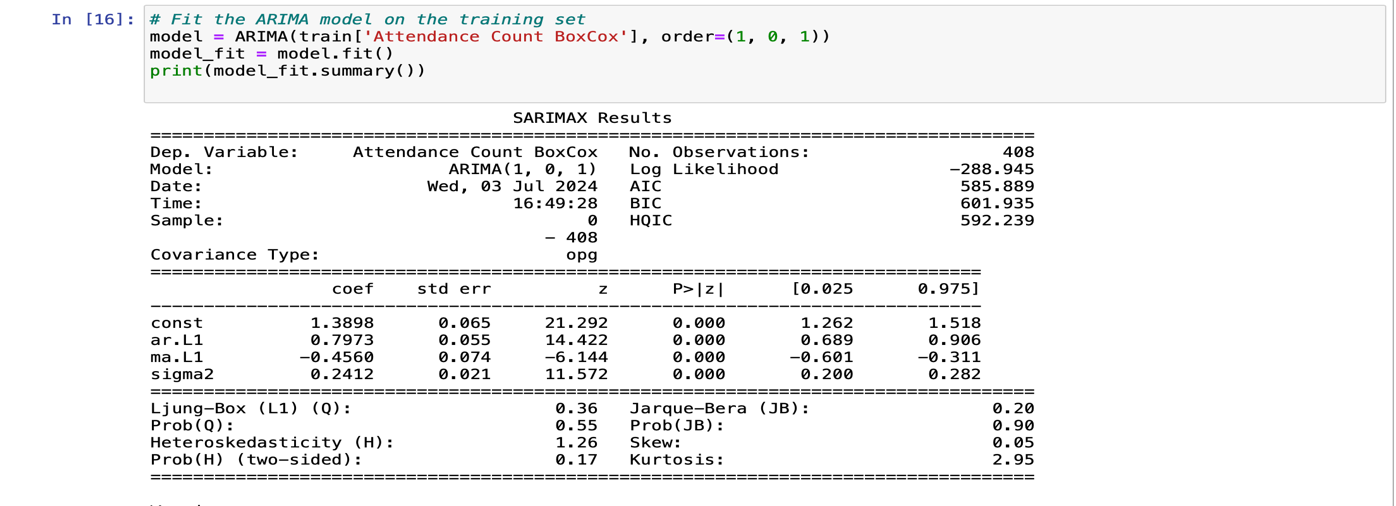
The ARIMA(1,0,1) model provides a smoothed representation of the attendance count data. The model captures the general trend and direction of the data but struggles with sharp peaks and sudden changes, which is typical for simple ARIMA models. This smoothing effect suggests that while the model is useful for understanding the general pattern, it might not be the best choice for data with high variability and frequent extreme values.

*Box-Cox transformation:*

The plot indicates that the ARIMA model's fitted values (red line) do not closely match the actual attendance counts (blue line) in several places, particularly where there are spikes or drops in the data, suggesting that the model may not be capturing the data's variability and volatility adequately. Applying a Box-Cox transformation can be a good choice in this situation for several reasons: it stabilises variance, which is useful when dealing with heteroskedasticity (non-constant variance) as indicated in the diagnostic tests; it normalises the data, addressing issues of skewness and kurtosis highlighted in the Jarque-Bera test results; and it improves model fit, making the ARIMA model better at capturing patterns in the data by stabilising variance and normalising the distribution.

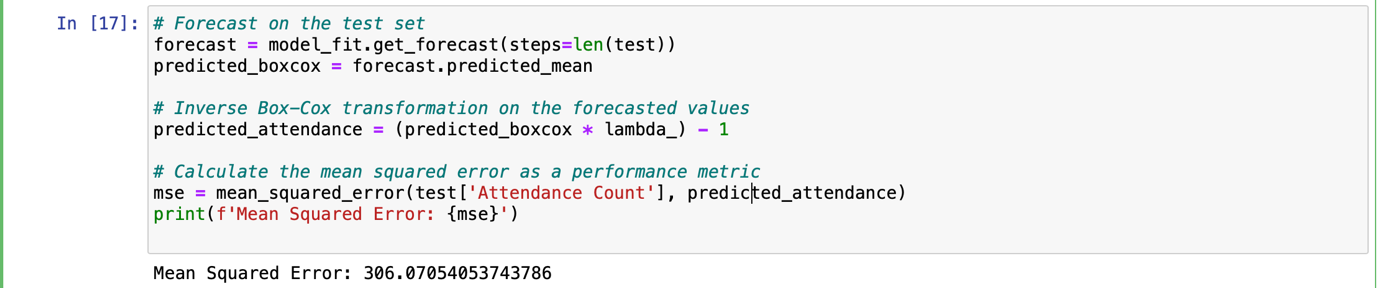


The Box-Cox transformation has improved the model's performance by stabilising variance and normalising the data. The coefficients for both the AR and MA terms are statistically significant, and the model diagnostics indicate no significant autocorrelation, normality of residuals, and constant variance. These improvements suggest that the Box-Cox transformation was beneficial, leading to a more accurate and reliable ARIMA model fit for the transformed 'Attendance Count' data.



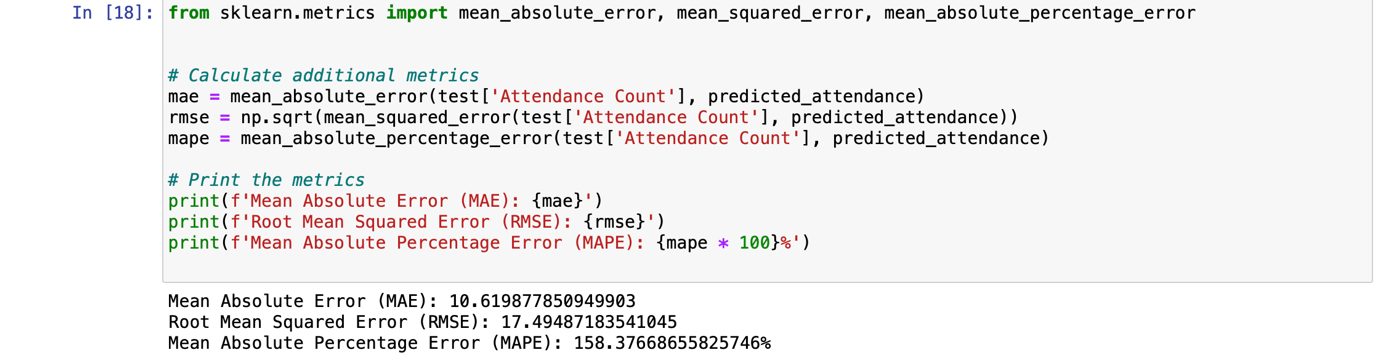
The MSE is a commonly used metric to evaluate the accuracy of a forecasting model. Lower values of MSE indicate better model performance, as they signify smaller errors between the actual and predicted values.

The output, Mean Squared Error: 306.07054053743786, indicates the average squared difference between the actual and predicted attendance counts is approximately 306.07.



The particularly high MAPE suggests that while the model may perform adequately in absolute terms (as shown by MAE and RMSE), it struggles with relative accuracy, especially for smaller attendance counts where percentage errors can be large. This highlights the need for further model improvement, possibly through:

* **Model Refinement**: Adjusting the ARIMA parameters or exploring different models.
* **Additional Features**: Incorporating additional predictors (e.g., time of year, marketing efforts, speaker popularity) to improve model accuracy.
* **Data Transformation**: Further data preprocessing or transformations to stabilise variance and improve model fit.



### Conclusion:

The application of ARIMA modelling to predict conference attendance has demonstrated its utility in capturing the underlying patterns within the historical data. While the initial ARIMA model provided a good fit for the training data, it struggled with high variability and sudden changes in the test data, indicating potential overfitting. The introduction of the Box-Cox transformation improved model performance by stabilising variance and normalising the data, leading to more reliable forecasts. However, the exceptionally high Mean Absolute Percentage Error (MAPE) suggests that further model refinement is necessary.

**Exploring XGBoost for Conference Ticket Registration Prediction**

The second library to trial was XGBoost. XGBoost was chosen as a library due its strong sales prediction ability along with having the ability to handle missing values seamlessly (Simplilearn, 2023), which is imperative in this project due to our decision to remove the data affected by the COVID-19 pandemic. Highlighted previously, this was the main reason that the ARIMA library was unable to predict accurately.

**Benefits of using XGBoost**

As a powerful machine learning algorithm, XGBoost has many advantages suitable for our project:

1. Robust Handling of Missing Data: Managing missing values effectively allows the model to remain robust and accurate when dealing with incomplete datasets.
2. High Prediction Accuracy: XGBoost is known for its exceptional performance and high accuracy in predictive tasks, especially in sales and demand forecasting (Chen & Guestrin, 2016).
3. Efficiency and Scalability. A programme which is capable of handling large datasets while performing optimally is imperative for this project as Gerard may use the application for extensive ticket registration data over multiple conferences at a time as his clients see the benefits of this technology.

**Trial Results and Discussion**

**Approach 1: All CSV Files used for training**

In this approach, all datasets were used to train the model. This returned a Root Mean Squared Error (RSME) of 20.604. XGBoost uses parameters to fine-tune a model. After researching the many available parameters, it was decided that 'max\_depth': 6, 'min\_child\_weight': 1, 'subsample': 0.8, 'colsample\_bytree': 0.8, and 'learning\_rate': 0.1 would be used at this stage. These were selected due to their compatibility with our data and the nature of the project. The levels were chosen to maximise the model’s prediction ability, providing a strong starting point while allowing future room for improvement. For example, a low number was chosen for the minimum child weight to ensure the predictive model could make a prediction, even if it was to be a bad one. However, it was at this point that we realised cumulative ticket sales were needed, as the end result was 24.61 predicted ticket sales, shown in figure 1. This information was quickly passed and discussed with the team to ensure the same error was not repeated.

A screenshot of a computer

Description automatically generated

Figure 1. Predicted Sales and RSME of Approach 1

**Approach 2 – Cumulative Ticket Sales and Test vs Train Data**

Upon reflection on how to improve this and test the model accurately, it was decided to keep one CSV file out of the training set to have separate training and testing datasets. This would allow us to test the efficiency of the model on real data. The group also decided to add Mean Absolute Percentage Error (MAPE) as a performance measure. All parameters were kept the same; however, the RSME increased to 43.04, and the MAPE was 84.27%. These results were significantly higher than before, the only difference being the cumulative recording of tickets.

A graph of a number of data

Description automatically generated with medium confidence

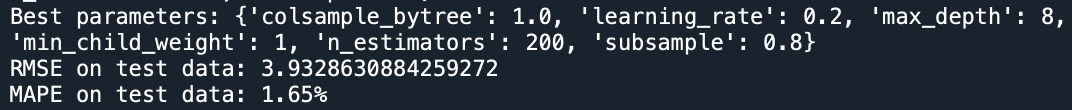
Figure 2 – Actual vs Predicted Cumulative Registrations.

Confident in the decision to separate the test and training datasets, we researched ways of improving the RSME and MAPE; a parameter grid was found as a way of finding the ideal combination in parameter levels. This was included in our code as seen below:

A computer screen shot of a code

Description automatically generated

This returned the following results:



With this strong result, as a team we decided to go ahead with developing the usability of this programme.

**Software Usability and Transition to Prophet**

Together and separately, we trialled many different ways of turning this programme into a usable tool for Gerard. Unfortunately, despite the low error rates, the program still fell short of realistic results. For example, we tried using a start and end date of the registration window, considering the current point in that period along with the ticket registrations. In one instance, a period of four months was used, with 100 tickets sold at the halfway mark. The program then predicted that only 3.64 more tickets would be sold by the time the conference came around, which was clearly unrealistic. See below.

A screen shot of a computer

Description automatically generated

As a team, we invested many hours into making this software work. However, all efforts resulted in similar, unrealistic outcomes. Due to time restrictions, we decided to shift our focus to the Prophet library, which is designed specifically for time series forecasting, including sales predictions. We trialled additional software concept ideas we had discussed for this program with Prophet, which gave more favourable results. To avoid explaining these different approaches multiple times for the two different libraries, these will be explained in the next part of this report with the favourable results.

When discussing our transition to Prophet, a key benefit of this library is its ease of use and straightforward implementation, requiring minimal parameter tuning (Taylor & Letham, 2018) which we discussed may have been a hindrance in our XGBoost discovery, as with so many available parameters it is hard to truly know which ones would work best for this project. Finding a library which has less complexities to allow us to create a usable product for Gerard was our priority, while strengthening this with adding more features develops over time.

**Using Prophet to do ticket Sales forecasting**

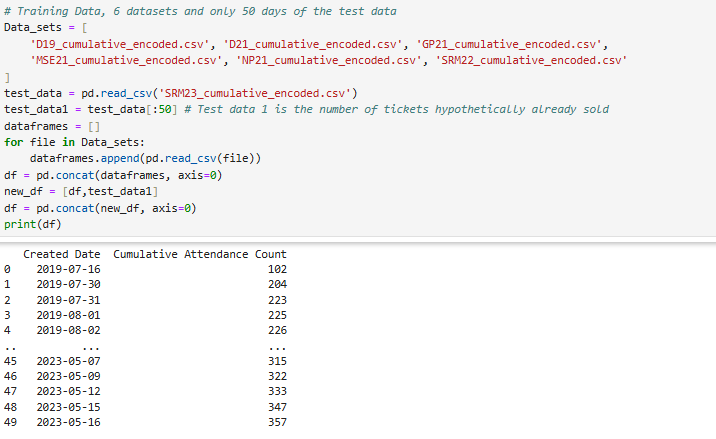
Prophet is a method for predicting time series data using an additive model. It fits non-linear trends with seasonality on a daily, weekly, and annual basis, together with the effects of holidays. Strong seasonal effects in time series and multiple seasons of historical data are ideal for its effectiveness. Prophet usually handles outliers well and is resilient to missing data and trend changes (Prophet, 2023). Prophet can be executed both in R and Python.

Prophet is particularly well-suited for this type of data because it was designed specifically for sales prediction tasks. It simplifies the forecasting process by requiring just two columns: a date column and a numeric column representing the quantity to be forecasted.

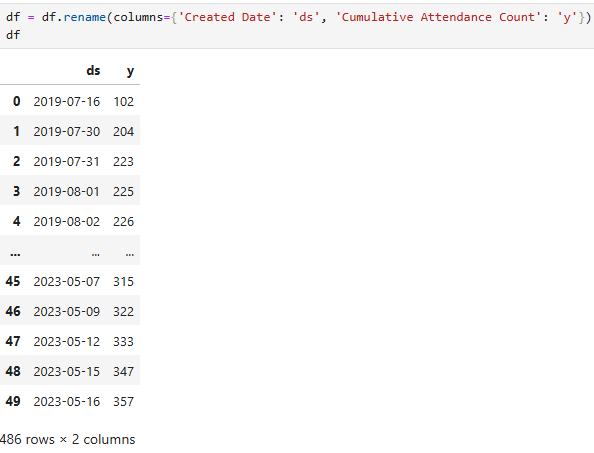
To try and get the best results from Prophet, three approaches were utilised.

1. Using Cumulative Data of tickets sales over a period
2. Using per day ticket sales over a period (daily aggregated ticket sales)
3. Prediction using Similar/closely related types of events

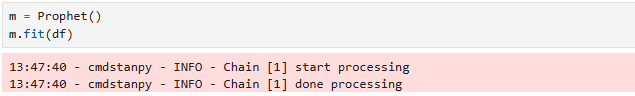
**Approach 1: Using Cumulative Data of tickets sales over a period**

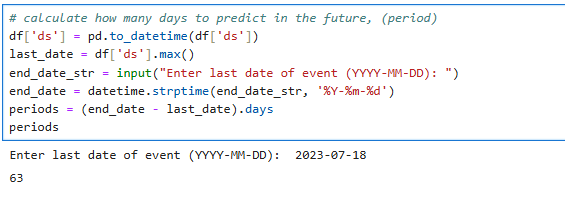
In this approach, dataset comprised seven separate datasets where six were merged to form a larger dataset, and the seventh dataset, only 50 rows (representing days when tickets were hypothetically sold) was selected. The objective was to predict the total number of tickets that might be sold by the planned D-day. The rest of the seventh dataset served primarily as a test dataset to validate the predictions made from the merged dataset. 

To work with prophet, we renamed these two columns (Created Date and Cumulative Attendance Count) to ‘ds’ and ‘y’ respectively.

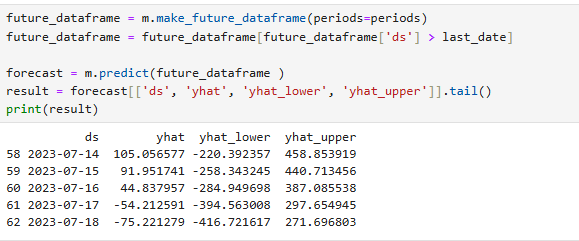


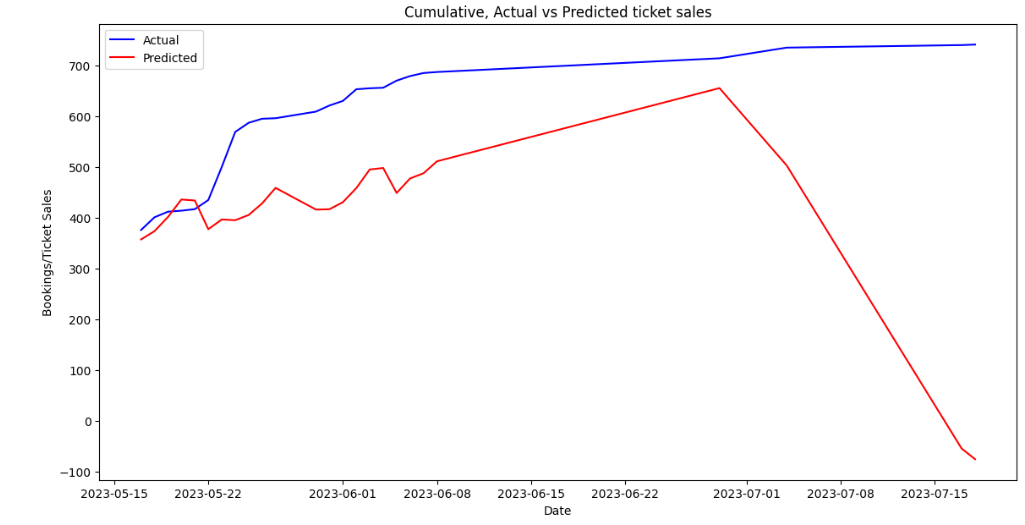
Then proceeded to train the model.



From here, we needed to determine the number of days into the future for which we wanted to make predictions. The client does input the last date leading up to the D-Day, and the program then calculates the days remaining until that point. 

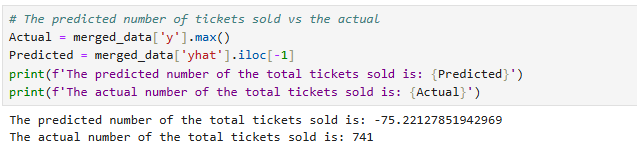
With the number of days calculated, we make a future data frame of the prediction of tickets till that said date.



To see how well the model performed, we compare the predicted ticket sales with the actual tickets sold (remaining rows of the test data, from row 50 to the end) by plotting them side by side. The results are shown below. 

***Graph 1: Cumulative Graph for Actual vs Predicted ticket sales***

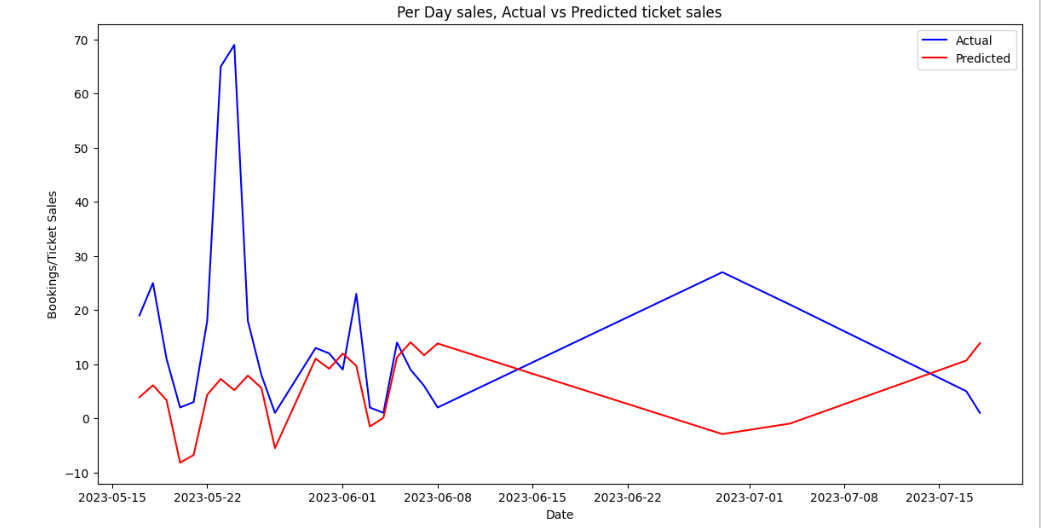
We notice that the graph drops sharply as it gets close to the event day. This happens because, with cumulative data, ticket sales reach a peak and then stop, which the model sees as a drop when a new event begins. Ticket sales are high at the end of the current event, but when a new event starts, sales begin at a low and then gradually increase over time.



The final number predicted by the model is **-75.** This approach was a complete failure.

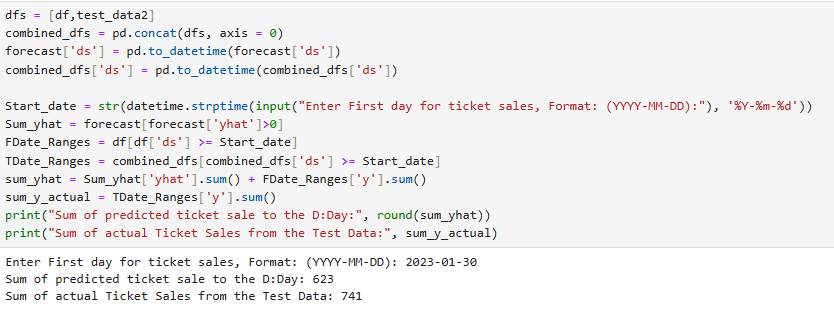
**Approach 2: Using per day tickets sales over the period (daily aggregated ticket sales)**

Using the same procedure and using datasets of daily aggregated ticket sales. We retrained the model again and these were the findings



***Graph 2: Per-Day sales Graph for Actual vs Predicted ticket sales***

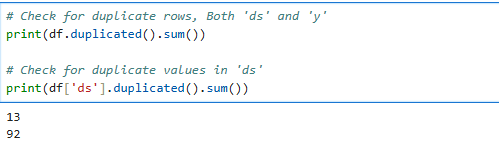
The predicted total number of tickets sold was **623** vs the actual sales of **741** tickets. This approach seemed to be better than using the cumulative dataset.



To improve the model, we tried to adjust it to focus mainly on the days we have sales data and ignore the days when no ticket sales happened. To do this, we set all days without sales to 'NaN'.

First, we identify all the days without any data.

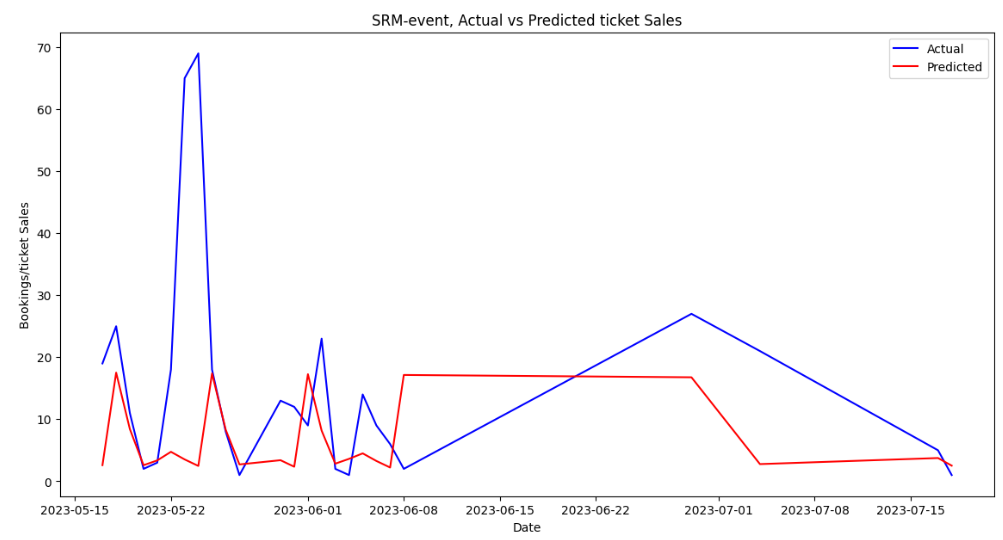
In total, there are 1007 days. However, setting these dates as ‘NaN’ was not feasible because some days had overlapping ticket sales for two or more different events. As a result, some dates had duplicates. These were the results from our duplicate ‘whole row and date’ check.



This issue stemmed from duplicate dates within the 2021 datasets. Prophet struggles with duplicated data, so the solution is to either remove these duplicates or aggregate them by taking their mean and then retrain the model. However, this situation also highlighted the diversity in ticket sales. From this observation, an idea emerged: why not train the data separately for similar types of events? Based on information from the client, we identified three groups of events: IT Managers, Property Managers as well as Education Managers events. With that, we embarked on approach three.

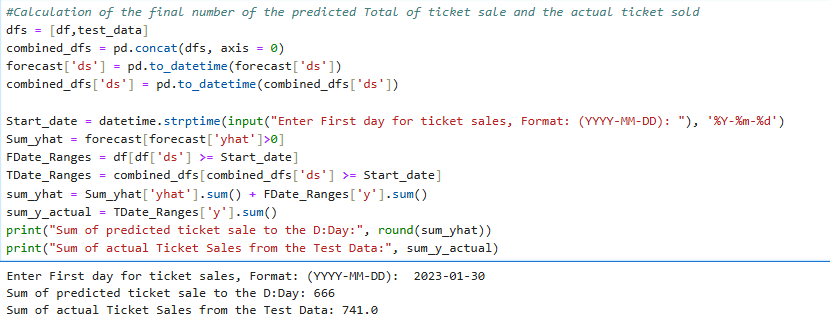
**Approach 3: Similar types of events**

With three types of events identified, we choose to use the Education Managers given it had more data and few dates with zero ticket sales unlike the other two events. Using the same approach as before, we retrained the model and below were the results;



*Graph 3: Similar types of events, G Graph for Actual vs Predicted ticket sales*

The predicted number of ticket sales was **666** against the actual number of **741** tickets. This approach bore better predictions than the previous two approaches.



In order to evaluate the precision of our predictive models and forecasting task, we computed multiple statistical metrics:

1. The average size of errors between our anticipated values and the actual results is provided by the Mean Absolute Error (MAE) measure. It is helpful in determining the average distance between our forecasts and the true values.
2. Similar to MAE, Root Mean Squared Error (RMSE) takes into account the square of the deviations between the expected and actual values. This provides a measure of the spread of errors among our forecasts by assigning greater weight to larger errors.
3. The average absolute % difference between the expected and actual values is measured by the Mean Absolute % Error, or MAPE. It aids in our comprehension of how accurate our forecasts are in comparison to the actual values, especially when dealing with forecasts when the percentage error scale is significant.

These metrics aid in measuring our models' performance. Reduced MAE, RMSE, and MAPE values signify improved precision and a closer match between our forecasts and the observed results. Looking at our approaches, these were the results

**Approach 1:**

1. Mean Absolute Error (MAE): **188.84503334381242**
2. Root Mean Squared Error (RMSE): **271.76176427456505**
3. Mean Absolute Percentage Error (MAPE): **110.51%**

Approach 1 showed a large negative discrepancy in predicted total tickets sold compared to the actual total. The error metrics also show significant inaccuracies, especially with high RMSE and MAPE values, suggesting the model's predictions are far from the actual values.

**Approach 2:**

1. Mean Absolute Error (MAE): **13.48200238621738**
2. Root Mean Squared Error (RMSE): **20.567592913634403**
3. Mean Absolute Percentage Error (MAPE): **16.81%**

Approach 2 shows a closer alignment between predicted and actual ticket sales, with lower MAE, RMSE, and MAPE values compared to Approach 1. This indicates improved accuracy in predicting sales leading up to the event.

**Approach 3:**

1. Mean Absolute Error (MAE): **12.01937594367789**
2. Root Mean Squared Error (RMSE): **20.114267989606788**
3. Mean Absolute Percentage Error (MAPE): **75.09%**

Approach 3 also shows further improvement in accuracy compared to both Approach 1 and Approach 2. The MAE and RMSE values are lower, indicating closer predictions to the actual ticket sales. However, MAPE is actually higher than approach 2.

**Conclusion**

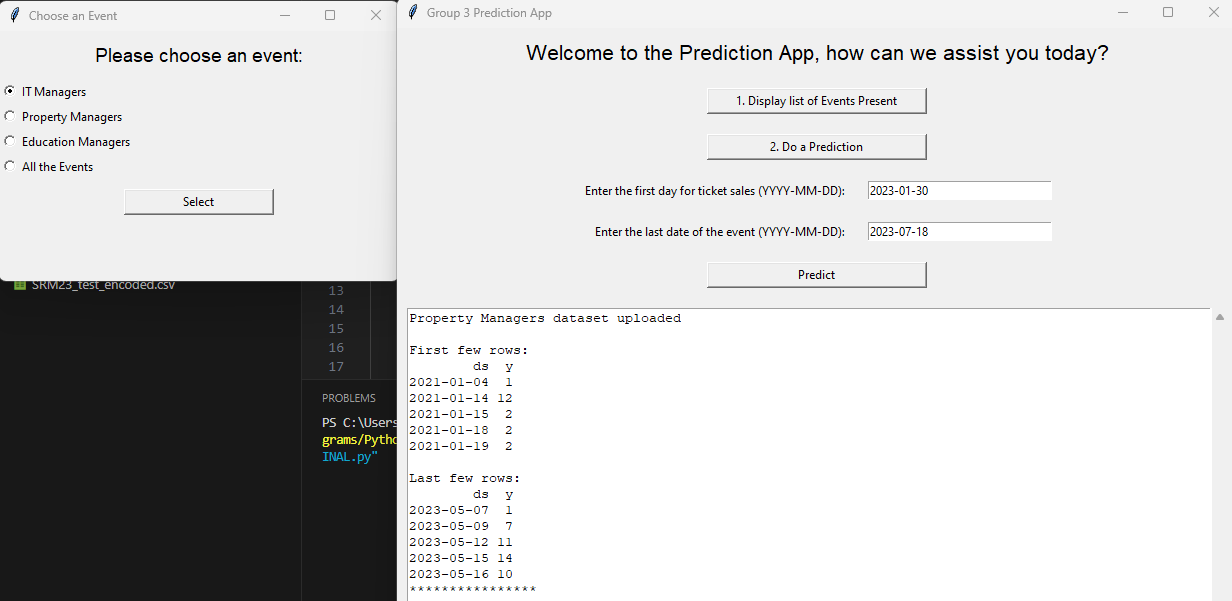
Unlike RMSE, MAPE normalises the mistakes as percentages, it makes comparing different models or datasets easier. This makes straightforward comparisons between datasets with different scales.

Using the Prophet Library to do prediction, approach two (using the whole dataset) has a better MAPE followed by approach three where the same type of event data was used for training of the model and making prediction.

**CREATION OF AN APPLICATION**

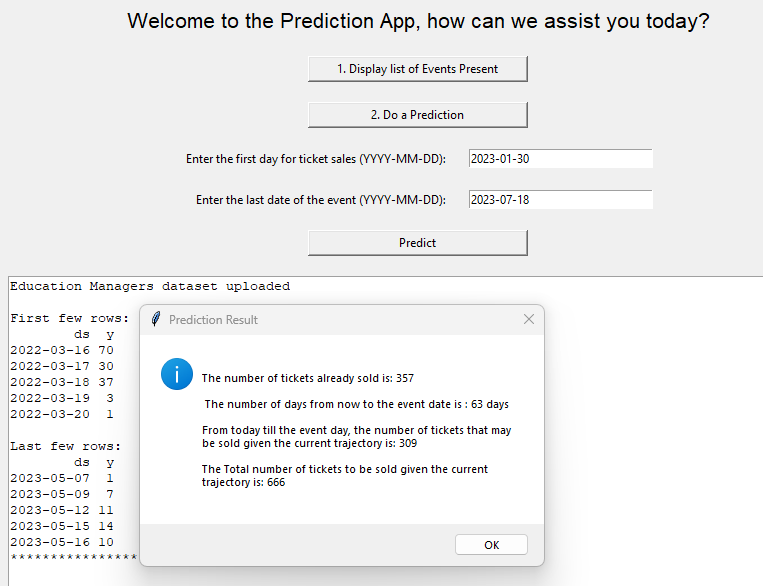
Since Prophet had better MAPE scores, we chose to create an interactive application using it. We used the Tkinter GUI application since it provides a user-friendly interface for our prediction using the Prophet library. It offers functionalities to display events and perform predictions, making it a valuable tool for our task.

**THE PREDICTIVE APPLICATION.**

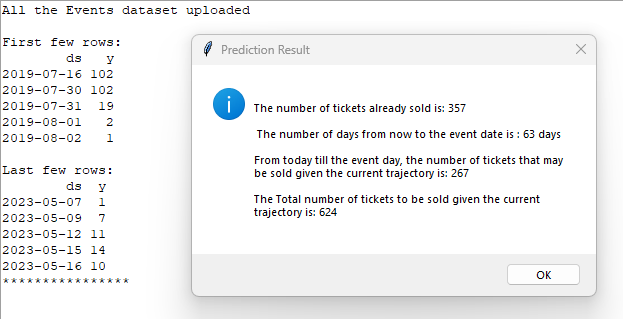
Gerald, our client, asked for a very simple interactive application, so we decided to use known details. In our case, the known details are the start date, end date, and target audience. Gerald also mentioned that sometimes there is no specific target number of tickets to be sold. With this in mind, we developed an application that asks for these three pieces of information.

Using the same methods from Prophet's approaches 2 and 3, where we use aggregated ticket sale data, you can choose which datasets to train the model and get a predicted value for ticket sales. We used the latest dataset, SRM23, targeting the Education Manager audience, and tested it with an actual total ticket sale of 741. Here are the results:

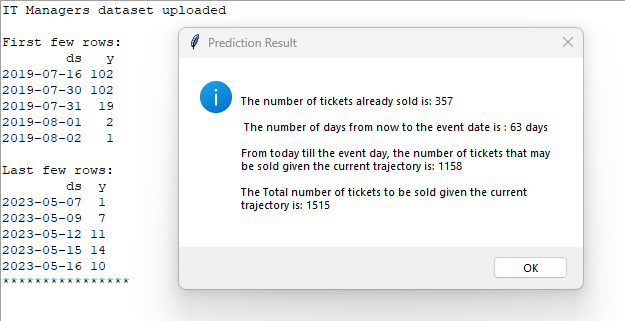
Using Education Managers datasets only to train the model;



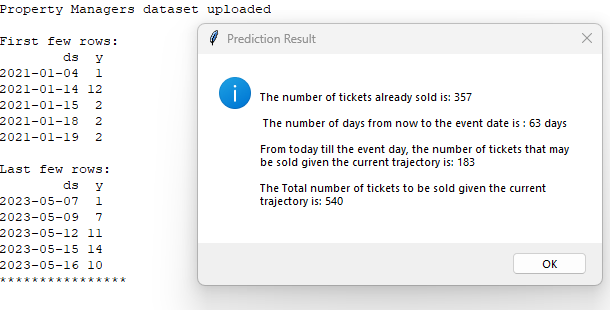
Using All the dataset,



Using IT managers datasets as the train data



And finally using property managers;



From the above model training, we found that using the same type of event data gives the closest value to the actual tickets sold, followed closely by using all datasets indiscriminately. This might be because the target audience's behaviour could be the same, making it the best for predictions.

Predictions can be repeated with different datasets based on the number of days tickets have been sold. If the client has a target number of sales, this predictive model can guide them on whether to increase advertising for ticket sales. It can show if they are falling behind in sales or if they are on the right track.

**The Execution of the Application**

To execute the application, follow these steps:

1. Ensure that all dependencies are installed. You can install them using pip:

***pip install tkinter pandas matplotlib prophet***

1. Save the complete code into a Python file, e.g., prediction\_app.py.
2. Ensure that all the dataset is named .csv and in the same directory as the Python file. This dataset should have at least two columns: date and sales aggregated per day.
3. Run the Python script:
4. The application window will open. Follow these steps to use the application:
5. Click the "Display list of Events Present" button to load and display events data to be used for the training model.
6. Click the "Do a Prediction" button to enable the input fields to be filled.
7. Enter the start date and stop date for the prediction in the format ‘*YYYY-MM-DD’.*
8. Click the "Predict" button to perform the prediction and view the results.

**Reference:**

1. Chen, T. & Guestrin, C., 2016. XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pp. 785-794.
2. Facebook, 2023. Prophet: Forecasting at scale. [online] Available at: <https://facebook.github.io/prophet/> [Accessed 6 July 2023].
3. Simplilearn, 2023. What is XGBoost? An Introduction to XGBoost Algorithm in Machine Learning. [online] Available at: <https://www.simplilearn.com/what-is-xgboost-algorithm-in-machine-learning-article> [Accessed July 2024].
4. Taylor, S. J. & Letham, B., 2018. Forecasting at scale. *The American Statistician*, 72(1), pp. 37-45.
5. XGBoosting, 2024. Tune XGBoost "max\_delta\_step" Parameter. [online] Available at: <https://xgboosting.com/tune-xgboost-max_delta_step-parameter/> [Accessed July 2024].
6. Abadi, M. et al., 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. CoRR, abs/1603.04467. Available at: <https://arxiv.org/abs/1603.04467> [Accessed July 2024].
7. Ramsundar, B. & Zadeh, R.B., 2018. TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning. O'Reilly Media.