MERLIN FORESIGHT

Time-series Park Demand Forecasting Model

Sergio Antonelli

Contents

Introduction

Assignment Background

The Merlin demand forecasting project is an ongoing linear project, currently in the testing phase. The premise of this project is to create a machine learning model to predict demand (attendance and therefore revenue) starting at Chessington World of Adventures, to then be rolled out and other parks globally (currently, the model has been rolled out at Chessington World of Adventures and Legoland Florida). This model is built using greykite, a forecasting library for python. The data used is past attendance for the last four years, as well as prebooks for the last 4 years split by channel (these include school prebooks, corporate prebooks, etc.). The regressors used by the model include school holidays, prebooks in advance, park events and weather. Greykite also finds patterns in the data to optimize the model, taking into considerations aspects such as seasonality and weekends. There are two separate models built for this demand forecasting, one using just past attendance, and another using prebooking data to forecast prebooks split by channel on the day. The results from both models are then imported into Power BI to create a dashboard for the finance team to use for revenue predictions and staff numbers. However, the forecast is mainly used by Boston Consulting Group who are working with Merlin to create a more optimized pricing plan for the park.

Requirements & Dashboard Design

The requirements for the model & dashboard are broad, as this is a somewhat experimental project. However, the requirements for the model are as follows:

- At least two weeks of attendance predictions.
- Predictions of tickets sold split by channel (annual passes, online purchases, promotions, etc.)
- A dashboard depicting said predictions and the model's historical accuracy.
- Daily predictions and dashboard refresh.

The dashboard predictions are as follows:

- Page 1: forecasted attendance for the next two weeks, as well as a YTD mean average error of the model.
    - On the same pane, a table showing all forecasted attendance tiers (these differ from park to park, but will usually be from 4 to 8 pricing tiers), as well as recommendations based on initially set attendance tiers.
- Page 2: forecasted prebooks split by channel.
- Page 3: historical forecasts for attendance for at least a year, with a mean average error and a percentage of correct tier predictions
- Page 4: same as pane 3, but for prebooking forecasts.

In order to connect to the data, we will simply be connecting to the correct container on an Azure storage account. Each region will have it's own container, each containing a folder for each park in that region.

Data Sources & Software

As in any data project, to collect all the relevant data, there was a lot of communication with both the project stakeholders and the business. The main data sources required involved the following:

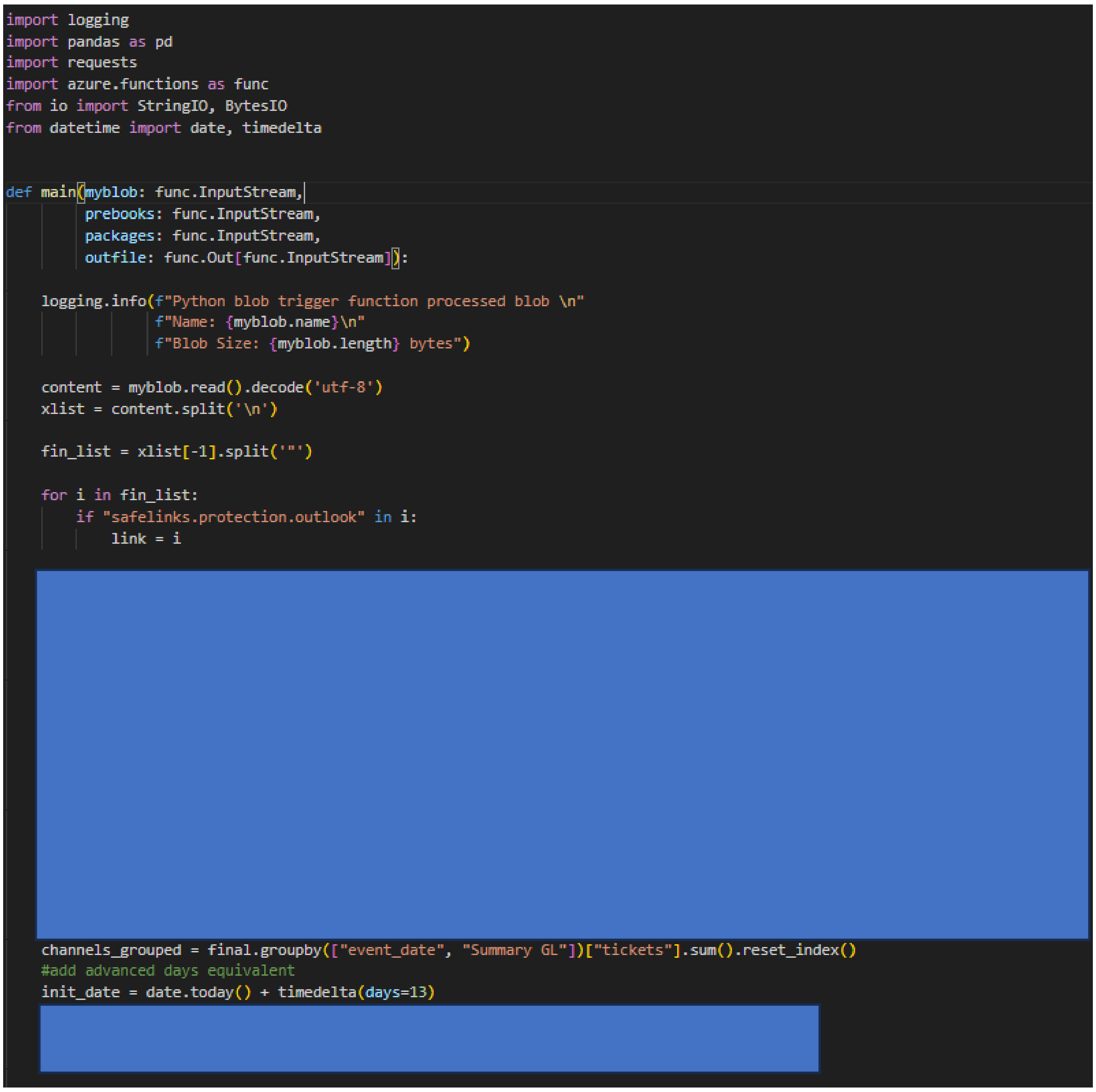
- Historical park attendance & prebooking data (from 2019 onward):
    - This data is taken from an Azure storage account, where data is ingested from accesso (ticketing system) servers.
    - The prebooking data is split by channel. The included channels, some of which are separated in sub-channels, are: Web, Annual Passes, Promotions, Hotels, Trade-in, Schools, Corporate and Walk up. The last two are other types of promotional ticket types: Consumer Frees & Sun Frees.
- Historical and current advanced prebooking data (therefore, the total amount of prebooks x days in advance from actual date):
    - This is a more complex data source. Currently, we do not have access to this data directly on our servers. We instead receive an email every day with a hyperlink containing the advanced prebooking data for the upcoming year. We will need to pipeline this to extract just the next 14 days' worth of data.
- Historical and forecasted weather data:
    - This is collected using an API from a weather forecasting website.
- Other data sources to be updated yearly:
    - School holidays: extensive research concluded the county of Kensington & Chelsea as the best school holidays calendar to use for this regressor. This is similar to most of London and surrounding, which make up the majority of the park's attendees.
    - Park events, closures, and hours: All of these regressors are collected from the park calendar.

Model Pipeline

Blob Trigger

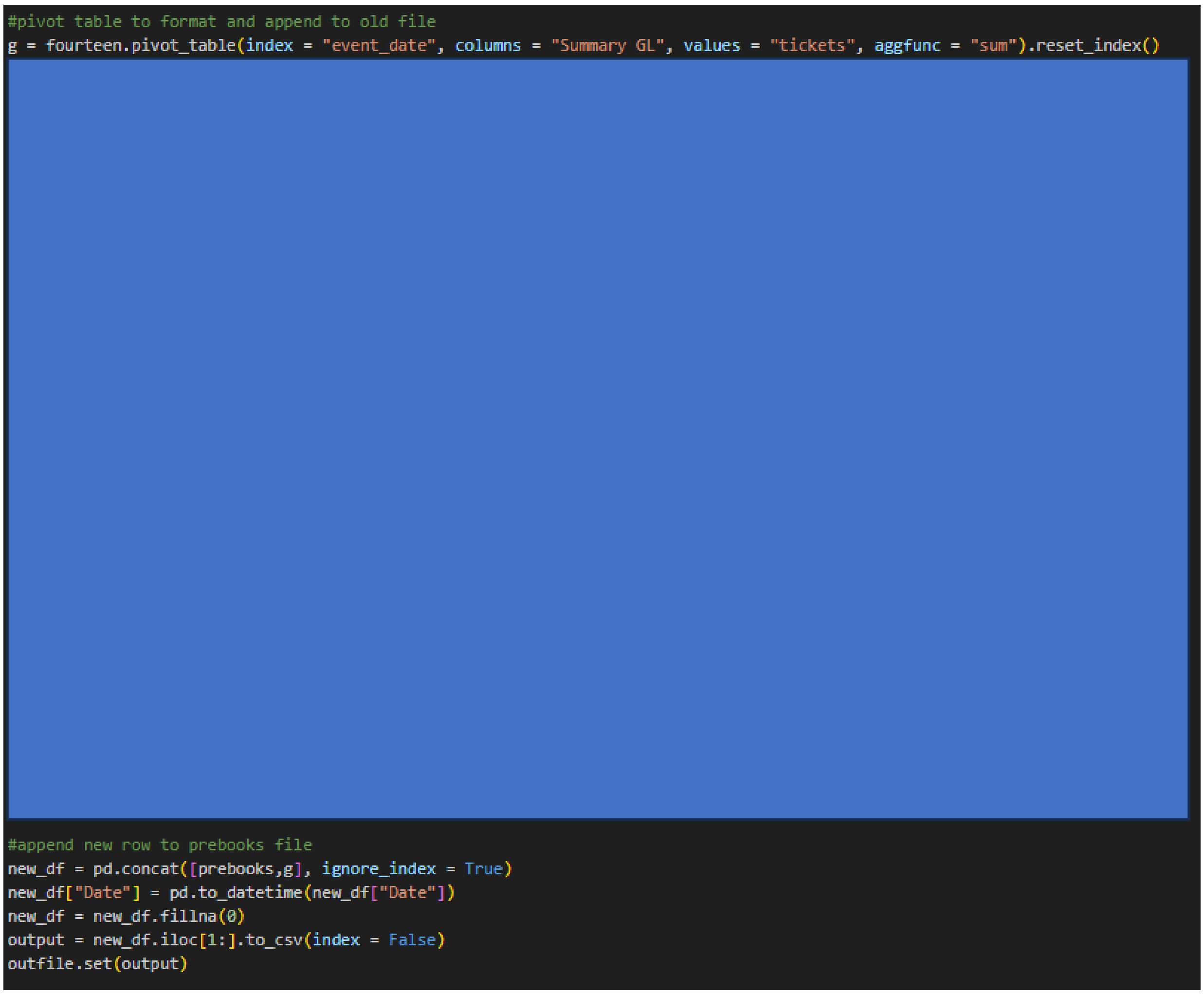
The model will run each night thanks to a chain of events which will lead to downstream reactions. The first event to trigger the pipeline is an "accesso insights" email containing the advanced prebooking data. An Azure Logic App will be triggered once these types of emails are received. The Logic App will count the files in the destination folder and save the email in a .txt file within the folder "EmailTXT" adding the number of current files to the name. This is done to prevent the logic app to overwrite the old files rather than adding a new one. Overwriting will cause the downstream function app to not recognize that a new file has been added, and would therefore not run. Once the new .txt file is added to the "EmailTXT" folder, an Azure Function will be triggered thanks to its connection to the folder. This logic app is displayed in figure 1.

The Function's job is to extract the hyperlink from the .txt file and send an HTTP request to extract the contents of the hyperlink as a pandas data frame in python. This file contains the current prebooking data for the next years' worth of dates. Our model only requires data for prebooks 14 days in advance. Therefore, from this data frame, we will only be extracting a single row of data, corresponding to prebooks for a date 14 days in the future. The Azure Function will then import another .csv file containing our historical prebooking data and will append this new row to it and store it back in the storage account. The first part of the function is dedicated to extracting the hyperlink ("link" variable) by splitting the .txt file contents and finding the correct text row identifier. The function will then use the requests package to get the contents of the .csv file within the hyperlink. After some formatting of both the imported data, as well as our own data, we can filter for the data relating to 14 days from the current date. Due to this code being Merlin IP, most of this code had to be censored. Below are snippets of this code:



```python
import logging
import pandas as pd
import requests
import azure.functions as func
from io import StringIO, BytesIO
from datetime import date, timedelta


def main(myblob: func.InputStream,
         prebooks: func.InputStream,
         packages: func.InputStream,
         outfile: func.Out[func.InputStream]):

    logging.info(f"Python blob trigger function processed blob \n"
                 f"Name: {myblob.name}\n"
                 f"Blob Size: {myblob.length} bytes")

    content = myblob.read().decode('utf-8')
    xlist = content.split('\n')

    fin_list = xlist[-1].split('"')

    for i in fin_list:
        if "safelinks.protection.outlook" in i:
            link = i

[redacted]
    channels_grouped = final.groupby(["event_date", "Summary GL"])["tickets"].sum().reset_index()
    #add advanced days equivalent
    init_date = date.today() + timedelta(days=13)
[redacted]
```

The second part of the function deals with some more formatting and pivots the table to get one row. This row is then appended to our historical prebooking file which is then saved back into our Azure container. Below are snippets of this code:



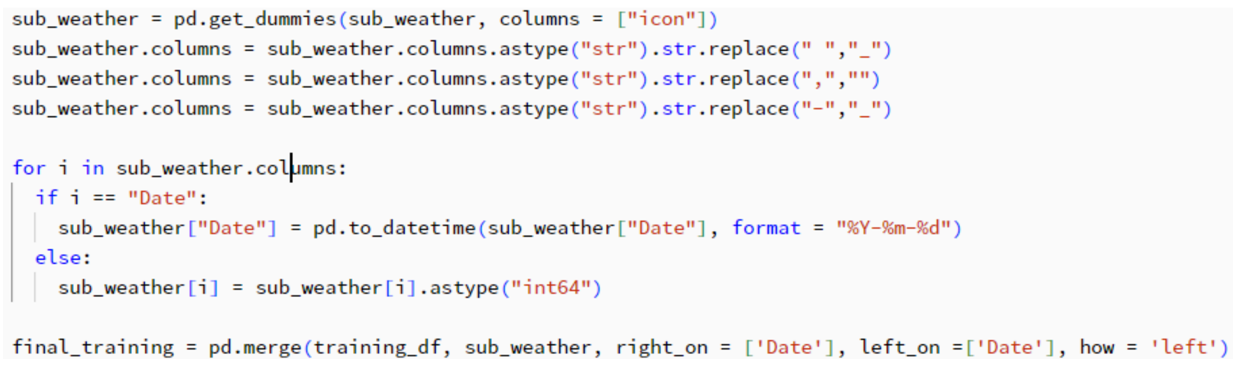
```
#pivot table to format and append to old file
g = fourteen.pivot_table(index = "event_date", columns = "Summary GL", values = "tickets", aggfunc = "sum").reset_index()
[redacted]
#append new row to prebooks file
new_df = pd.concat([prebooks,g], ignore_index = True)
new_df["Date"] = pd.to_datetime(new_df["Date"])
new_df = new_df.fillna(0)
output = new_df.iloc[1:].to_csv(index = False)
outfile.set(output)
```

Databricks Model Run

The python script in charge of running the actual forecasting model is set to run at 1AM every day. This is the only scheduled component of the model as the rest is run using various triggers set using Azure services. While the code is rather long and contains some sensitive data, the general pipeline is as follows:

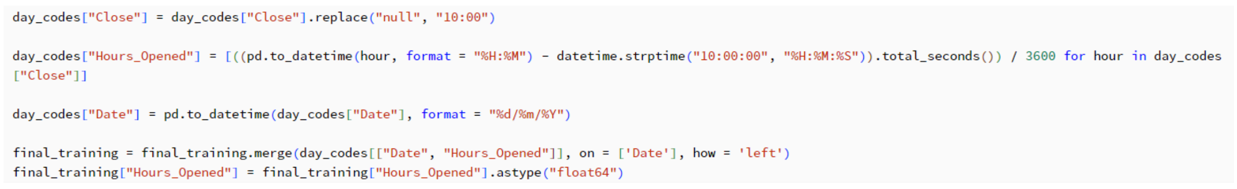
1. Import relevant packages (pandas, numpy, datetime, greykite, etc.)
2. Mount Azure Blob Storage (using storage account name & key)
3. Import historical attendance values from Blob Storage
4. Update historical attendance file with attendance data from new dates (this will update the data for all the parks that we plan to roll out the model to)
5. Save new attendance file and open using Pandas
6. Format table to have empty values for at least the following 14 days (required for model to run properly)
7. Add park closures as "Closed" column (binary values, 1 for closed, 0 for open)
8. Import historical weather values from Blob Storage
9. Run weather API to collect missing historical data as well as 14 day weather forecast
10. Update historical weather with missing historical dates

11. Create data dummies (multiple columns of binary regressors for one feature. Separate columns for rainy, clear, cloudy, etc. 1 for true, 0 for false) and merge to attendance dataframe



```
sub_weather = pd.get_dummies(sub_weather, columns = ["icon"])
sub_weather.columns = sub_weather.columns.astype("str").str.replace(" ","_")
sub_weather.columns = sub_weather.columns.astype("str").str.replace(",","")
sub_weather.columns = sub_weather.columns.astype("str").str.replace("-","_")

for i in sub_weather.columns:
  if i == "Date":
    sub_weather["Date"] = pd.to_datetime(sub_weather["Date"], format = "%Y-%m-%d")
  else:
    sub_weather[i] = sub_weather[i].astype("int64")

final_training = pd.merge(training_df, sub_weather, right_on = ['Date'], left_on =['Date'], how = 'left')
```

12. Add events regressor (manually added in using specific dates, will have to be updated once 2024 calendar is released)
13. Add school holidays regressor (.csv file imported with school holidays calendar for Kensington & Chelsea County), binary column: 1 for holiday, 0 for school day.
14. Add day codes: these make up two regressors, the park hours and the closed dates. The former is in the form of an integer representing the total number of hours the park is opened on any given day; the latter is a binary regressor, 1 for closed, 0 for open.



```
day_codes["Close"] = day_codes["Close"].replace("null", "10:00")

day_codes["Hours_Opened"] = [((pd.to_datetime(hour, format = "%H:%M") - datetime.strptime("10:00:00", "%H:%M:%S")).total_seconds()) / 3600 for hour in day_codes
["Close"]]

day_codes["Date"] = pd.to_datetime(day_codes["Date"], format = "%d/%m/%Y")

final_training = final_training.merge(day_codes[["Date", "Hours_Opened"]], on = ['Date'], how = 'left')
final_training["Hours_Opened"] = final_training["Hours_Opened"].astype("float64")
```

15. Add advanced prebooks regressor: this data is imported from Azure. Recently updated data using HTTP trigger.
16. We now have a data frame with 31 columns. 10 of these represent 10 separate channels of prebooks, and another 10 represent the prebooks for those channels 14 days in advance. The other 11 include the following columns: Date, Closed, Total Attendance, Events, Park Hours, School Holidays, Rainy, Clear, Cloudy, Partly Cloudy, Snow.
17. We are now ready to run the model. In order to make the model as quick as possible, we will separate the dataframe after each run with the relevant columns.
18. Starting with total attendance, we can remove all the channels columns, leaving us with 11 columns.
19. Each run after this is for predicting separate channels. Meaning it will use the channel total and the advanced prebooks for said channel, and won't need the total attendance, leaving us with 12 total columns.
20. There are a total of 11 runs, one for total attendance and 10 after that, one for each channel

```python
# Specifies dataset information
metadata = MetadataParam(
    time_col="Date",  # name of the time column
    value_col= i,  # name of the value column
    freq="D",  #"MS" for Montly at start date, "H" for hourly, "D" for daily, "W" for weekly, etc.
)

regressors=dict(
  regressor_cols= regressor_list
)

custom = dict(
  fit_algorithm_dict=dict(
      fit_algorithm="rf"
  )
)

model_components = ModelComponentsParam(
  regressors=regressors,
  custom=custom
)

forecaster = Forecaster()
result = forecaster.run_forecast_config(
    df=final_training_new,
    config=ForecastConfig(
        model_template=ModelTemplateEnum.SILVERKITE.name,
        forecast_horizon= 14,  # forecasts 14 steps ahead
        coverage=0.99,  # 99% prediction intervals
        metadata_param=metadata,
        model_components_param=model_components
    )
)
```

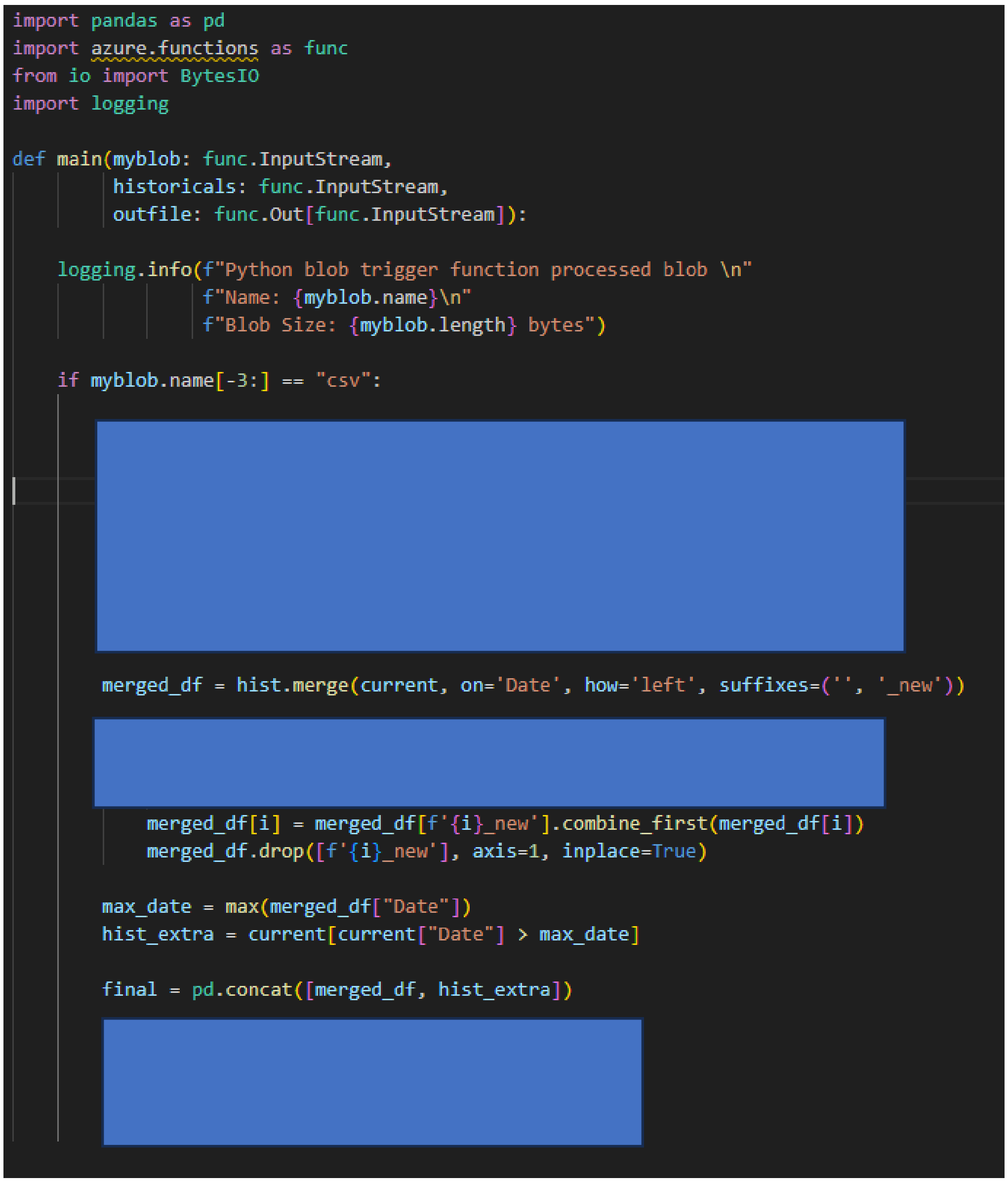
21. Each run is stored in a dictionary with a specific name as the key and the data frame as the value. We can then merge all the split channel runs into one to get a Total Prebooks forecast. This is another method to predict attendance, separate to the single model we ran at the start. We will also need each separate run which we can save within one big .csv file.

```python
for i in dict_dfs.keys():
  if merged_df.empty:
      merged_df['Date'] = dict_dfs[i]["ts"]
      merged_df['forecast'] = dict_dfs[i]["forecast"]
      merged_df['forecast_lower'] = dict_dfs[i]["forecast_lower"]
      merged_df['forecast_upper'] = dict_dfs[i]["forecast_upper"]
  else:
      merged_df = pd.merge(merged_df, dict_dfs[i], left_on='Date', right_on='ts', how='outer')
      merged_df['forecast'] = merged_df['forecast_x'].fillna(0) + merged_df['forecast_y'].fillna(0)
      merged_df['forecast_lower'] = merged_df['forecast_lower_x'].fillna(0) + merged_df['forecast_lower_y'].fillna(0)
      merged_df['forecast_upper'] = merged_df['forecast_upper_x'].fillna(0) + merged_df['forecast_upper_y'].fillna(0)
      merged_df.drop(['forecast_x', 'forecast_y', 'forecast_lower_x', 'forecast_lower_y', 'forecast_upper_x', 'forecast_upper_y'], axis=1, inplace=True)

# Sum the totals for each date
PrebookMethod = merged_df.groupby('Date').agg({
    'forecast': 'sum',
    'forecast_lower': 'sum',
    'forecast_upper': 'sum'
}).reset_index()
```

22. We will therefore save three files to Azure following these runs: TotalAttendance_Forecast (the first run), TotalPrebooks_Forecast (the sum of the last 10 runs) and SplitChannels_Forecast (the last 10 runs separated but kept in one file).

Azure Storage Trigger

Once the three forecast files are uploaded to the Blob storage, they will trigger another Azure Function. This function will take in these files and update historical forecast files. Thanks to the naming conventions of both the current and historical forecast files, one function will process all three types contemporarily. Since we want the newest forecasted values to overwrite the older historical values, we cannot simply use the .update() function but will instead have to merge all the new forecasted dates, and then use the combine_first function to keep only the newest values and remove the old forecasts. Since this will only keep forecasts that had matching dates, we will also have to append the rows with new dates (hist_extra). We can then save these historical .csv forecast files back to the Azure storage account. Snippets of this function are shown below:



```
import pandas as pd
import azure.functions as func
from io import BytesIO
import logging

def main(myblob: func.InputStream,
         historicals: func.InputStream,
         outfile: func.Out[func.InputStream]):

    logging.info(f"Python blob trigger function processed blob \n"
                 f"Name: {myblob.name}\n"
                 f"Blob Size: {myblob.length} bytes")

    if myblob.name[-3:] == "csv":

        [redacted]

        merged_df = hist.merge(current, on='Date', how='left', suffixes=('', '_new'))

        [redacted]
            merged_df[i] = merged_df[f'{i}_new'].combine_first(merged_df[i])
            merged_df.drop([f'{i}_new'], axis=1, inplace=True)

        max_date = max(merged_df["Date"])
        hist_extra = current[current["Date"] > max_date]

        final = pd.concat([merged_df, hist_extra])

        [redacted]
```

Dashboard Automation

Output Data

The data for the dashboard is a combination of exports pulled from the model outputs in Databricks, as well as historical attendance data that is saved after each run of the model (this includes prebook values and total attendance values) which will then be used to assess the accuracy of the model. All the tables referenced as model outputs are saved as .csv files on an Azure storage account. Table 1 has a comprehensive review of each piece of raw data that will later get imported into a dashboard.

DATA SOURCE	DESCRIPTION	SCHEMA
TOTATTENDANCE_FORECAST	Total forecasted attendance (model output)	Columns: “Date”, “forecast”, “forecast_lower”, “forecast_upper”
TOTPREBOOKS_FORECAST	Total forecasted prebook purchases (model output)	Same as above
SPLITCHANNELS_FORECAST	Forecasted prebook purchases split by channel (model output)	Same as above with repeating last three forecasts columns for each channel.
HISTORICAL_TOTATTENDANCE_FORECAST	Historical attendance forecasts (post-transformation)	Columns: “Date”, “Actuals”, “Forecast”, “Forecast Lower”, “Forecast Upper”
HISTORICAL_TOTPREBOOKS_FORECAST	Historical prebook forecasts (post_transformation)	Same as above
HISTORICAL_SPLITCHANNELS_FORECAST	Historical prebook forecasts split by channel (post_transformation)	Same as above with repeating last three forecasts columns for each channel.
HISTORICAL_ATTENDANCE	Contains historical values for prebooks split by channel and totals (with no forecasts)	Columns: “Date”, “Total”, “Web”, “AP”, “Promotions”, “Hotel”, “Trade”, “Schools”, “Corporate”, “Walk up”, “Consumer Frees”, “Sun Frees”
FISCAL_YEAR	A dictionary to convert dates in fiscal year, period and week for financial team.	Columns: “Date”, “FiscalPeriod”, “FiscalWeek”, “FiscalYear”
PARK_BUDGET	Range of dates from 2022 to 2023 containing set budget for park attendance. Used to compare our model vs current way of working at Chessington	Columns: “Date”, “Budget”

Connecting to data

Connecting to the .csv files was a simple process, using the built in “Get Data” function in Power BI (figure 2A). I connected directly to the Azure storage account holding all the data. The storage is split into containers, one for each country. Within each container are separate folders for each park, and within each park folder are four sub-folders:

- “Automation Files”: contains advanced prebooks, park budget, fiscal year, historical attendance, package codes & weather conditions.
- “Current Forecasts”: contains TotAttendance, TotPrebooks & SplitChannels forecasts.
- “Historicals”: contains one historical data file per forecasting type.
- “EmailTXT”: mentioned earlier, only used as a storage for .txt files created by the logic app in order for the Azure Function trigger to work properly.

Using Power Query, I imported the whole “chessington” folder from the “UK” container in our azure storage account. I did the same for legoland florida by importing “llf” from “US”. By doing this, I could easily work on all the files I need that are within this folder. These will have to be transformed to properly fit the dashboard requirements.

Data Transformation in Power Query

The first step in transforming our data was making sure that all our tables had proper headers and the correct data types. This model will forecast attendance with decimal numbers, so the all the forecasted numbers had to be converted from decimal numbers to whole numbers (since you can't have a decimal number as attendance). Next, some of the date columns were not automatically detected due to their long format. I therefore had to first change the data type to datetime, and then date. Most other data types were correctly predicted. Most column headers were then renamed to something more sensible, such as "ts" to "Date" or "forecast_lower" to "Forecast Lower". park_budget and fiscal_year were already formatted correctly after these adjustments. The processes used to transform the rest of the data sources are listed below:

TotAttendance & TotPrebook Forecasts

For these data sources (figure 3A), I merged the park_budget table onto "Date" to get budget values for each of the upcoming forecasted dates using a left-outer join. I then removed the "actual" column (which is automatically generated as a blank column in our model by the python library we use). The result is a table with 5 columns, as seen in figure 3B.

Historical Attendance

This table was already properly formatted as it was cleaned and modelled in Databricks prior to the model run.

Historical TotAttendance & Historical TotPrebooks

These data sources share an identical format but use two different methods to obtain values (figure 4A). I therefore had to add a custom column to each table called "Method", which represents whether the table used the "attendance" method or the "prebook" method. I then appended the latter to the former to create a big table containing all values. The reasoning for doing this rather than creating a relationship is to later create visuals with both methods which can be selected using slicers. I then merged the actual total attendance values from the transformed and cleaned blob storage data table using a left-outer join (final result in figure 4B).

SplitChannels & Historical SplitChannels

Before working on the channels forecast table, I needed a dictionary for channels. I therefore manually created a table with two columns: channel ID and channel name (figure 5).

For the channels forecast, I started with a wide data format, where I had a row with unique dates, and then three columns per channel (forecast, forecast lower bound and forecast upper bound). I first unpivoted all the columns other than date. I then split the column formed from column names in order to get a separate column with values: forecast, forecast_lower and forecast_upper. I then repivoted this new column to get new values separated by forecast boundaries in three columns, and separated by channel using a channels column. Finally, I merged the dictionary table for channels I had created earlier, kept the channel IDs and removed the names for a cleaner dataset (figure 6).

Relationships

Before applying all these changes, some tables are disabled from loading, as they won't be needed in the visualization and will allow for quicker loading times. The tables I will keep in this model, which have now been renamed to something more sensible, are as follow:

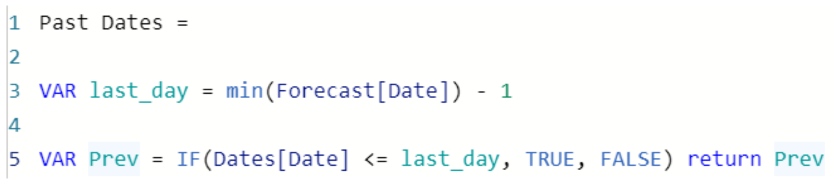
- TotPrebooks: "Prebooks Forecast"
- TotAttendance: "Attendance Forecast"
- Historical SplitChannels: "Historical Channel Forecasts"
- Historical TotAttendance & TotPrebooks: "Historical Totals Forecast"
- SplitChannels: "Split Channels Forecast"
- Codes: dictionary to retrieve channel name from channel ID in other tables
- Dates: table of dates created with DAX (figure 7)
- Weather: weather description per day
- Fiscal Calendar: fiscal calendar for reference

After applying all the changes to the dataset, relationships are created between tables to get the desired output in the visuals. We used a sort of "Kimball style" relationship between the data tables. To do this, a "Dates" table is created using DAX expressions. All the tables are then connected to the dates table with one to one or many to one relationships. The channels dictionary table was also connected using one to many relationships to all tables containing channel IDs. The final output of these relationships can be visualized in figure 8.

Measures and DAX Columns

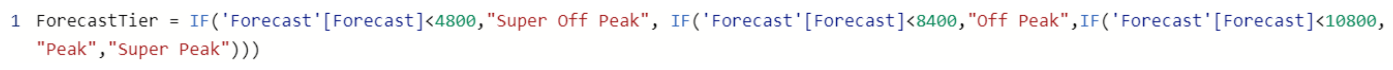
There are various other calculated columns created in some of the tables for better calculations, as well as some measures to display on cards throughout the dashboard. These are listed below:

- Dates Table:
    - Past Dates (column): This is a Boolean column showing whether each date is part of previous forecasts or the upcoming forecast. This is used to filter out upcoming dates in the historical forecast visuals and other measures.



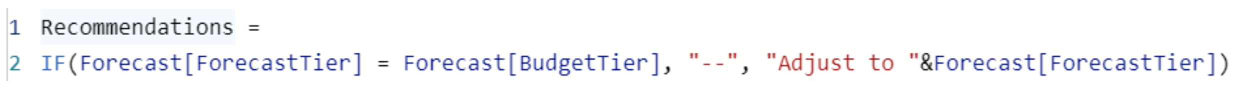
```
Past Dates =

VAR last_day = min(Forecast[Date]) - 1

VAR Prev = IF(Dates[Date] <= last_day, TRUE, FALSE) return Prev
```

- Forecast & Prebook Forecast Table (figure 9A):
    - Forecast Tier (column): This is the attendance tier calculated using the forecasted attendance numbers.



```
ForecastTier = IF('Forecast'[Forecast]<4800,"Super Off Peak", IF('Forecast'[Forecast]<8400,"Off Peak",IF('Forecast'[Forecast]<10800,
"Peak","Super Peak")))
```

    - Budget Tier (column): Same as forecast tier but for the pre-set park budget



```
Recommendations =
IF(Forecast[ForecastTier] = Forecast[BudgetTier], "--", "Adjust to "&Forecast[ForecastTier])
```

    - Super off-peak, off-peak, peak, super peak (measures): Represent each attendance tier as their maximum values, used in visual as lines (super peak = 12000, peak = 10400, off peak = 8400, super off peak = 4800).

- Recommendations (column): A string showing recommendations based on set tier and forecasted tier

```
Recommendations =
IF(Forecast[ForecastTier] = Forecast[BudgetTier], "--", "Adjust to "&Forecast[ForecastTier])
```

- Past Channel Forecast:
  - Difference (column): Absolute difference between forecasted attendance and actual attendance

```
Difference = ABS('Past Channel Forecasts'[Actuals] - 'Past Channel Forecasts'[forecast])
```

- Past Forecasts (figure 9B):
  - Actual Tier & Forecast Tier (columns): Same as previous tier columns, based on attendance
  - Difference (column): Same as previous difference column
  - Accuracy (column): Boolean column. True if actual and forecast tier match, otherwise false.

```
Accuracy = IF('Past Forecasts'[Actual Tier] = 'Past Forecasts'[Forecast Tier], TRUE, FALSE)
```

  - Tier Accuracy % (measure): measures percentage of accurate dates using “Accuracy” column.

```
Tier Accuracy % = DIVIDE(CALCULATE(COUNTROWS('Past Forecasts'), 'Past Forecasts'[Accuracy] = TRUE), COUNTROWS('Past Forecasts'))
```

The built dataset for the dashboard was then published to a workspace on our Power BI platform prior to any visualization, and the empty report that gets generated was removed. This is common practice at Merlin to allow for others to connect to the same dataset and create their own dashboards if needed. I therefore created a new .pbix file and connected to the dataset we created earlier. The visualization is described in detail in the next section.

Data Visualization

According to the requirements of this dashboard, we will have four pages within the dashboard showing the upcoming totals forecast, the upcoming channels forecast, the historical forecast for totals and the historical forecast for channels. Finally, I added a fifth page showing just the actual totals with various filters as per a request from the financial team. The five pages, along with descriptions for each, are listed in the next section.

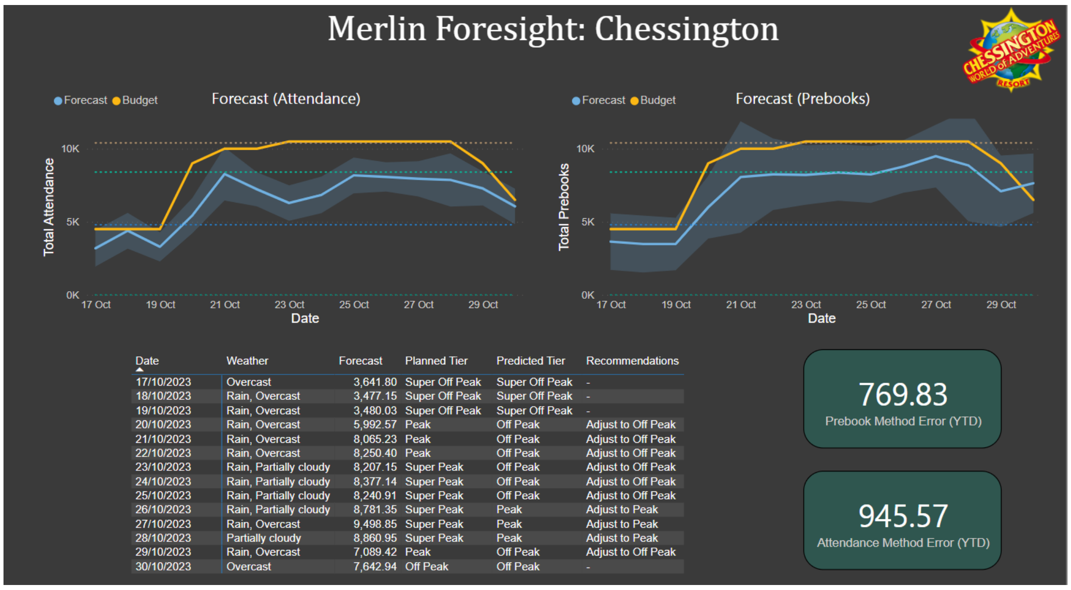
Dashboard Design

All dashboard screenshots below are using a random dataset for data privacy purposes, and do not represent any real-life data.

Page 1: Upcoming attendance forecast.

On this first page, we will be creating visuals for the upcoming two weeks of forecasted attendance.

1. Line Graph Visual: This pane will show both the total attendance forecast method and the prebook method. These will be shown in two separate line graphs, both showing error bars and lines representing each attendance tier, as well as another line showing the set budget for each date. The X-axis is therefore date, while the Y-axis are budget, forecast, and the four tiers. To set the error bars on the forecast line, we can go on Analytics -> Error Bars -> input lower and upper boundaries for forecast.



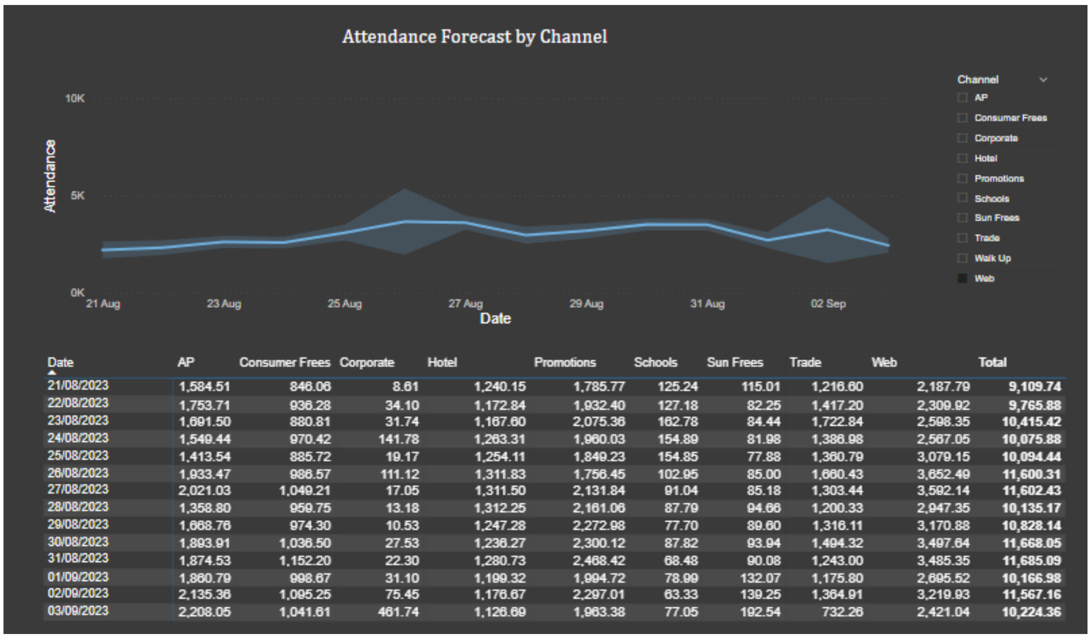
Date	Weather	Forecast	Planned Tier	Predicted Tier	Recommendations
17/10/2023	Overcast	3,641.80	Super Off Peak	Super Off Peak	-
18/10/2023	Rain, Overcast	3,477.15	Super Off Peak	Super Off Peak	-
19/10/2023	Rain, Overcast	3,480.03	Super Off Peak	Super Off Peak	-
20/10/2023	Rain, Overcast	5,992.57	Peak	Off Peak	Adjust to Off Peak
21/10/2023	Rain, Overcast	8,065.23	Peak	Off Peak	Adjust to Off Peak
22/10/2023	Rain, Overcast	8,250.40	Peak	Off Peak	Adjust to Off Peak
23/10/2023	Rain, Partially cloudy	8,207.15	Super Peak	Off Peak	Adjust to Off Peak
24/10/2023	Rain, Partially cloudy	8,377.14	Super Peak	Off Peak	Adjust to Off Peak
25/10/2023	Rain, Partially cloudy	8,240.91	Super Peak	Off Peak	Adjust to Off Peak
26/10/2023	Rain, Partially cloudy	8,781.35	Super Peak	Peak	Adjust to Peak
27/10/2023	Rain, Overcast	9,498.85	Super Peak	Peak	Adjust to Peak
28/10/2023	Partially cloudy	8,860.95	Super Peak	Peak	Adjust to Peak
29/10/2023	Rain, Overcast	7,089.42	Peak	Off Peak	Adjust to Off Peak
30/10/2023	Overcast	7,642.94	Off Peak	Off Peak	-

769.83
Prebook Method Error (YTD)

945.57
Attendance Method Error (YTD)

2. Table with recommendations: This table will show the planned and predicted tier for each day, along with the recommendations for that day and a column for weather, to give guidance to the pricing team on what price to set at each given day.
3. Cards: This page will have three cards. Two of these will show the average error of our model for each method from the start of the year. We can achieve this by using a relative date filter and select the last year. To then filter out future dates, we can use the dates column "Past Dates" and place it in the filters pane, selecting "True".

Page 2: Upcoming Channels



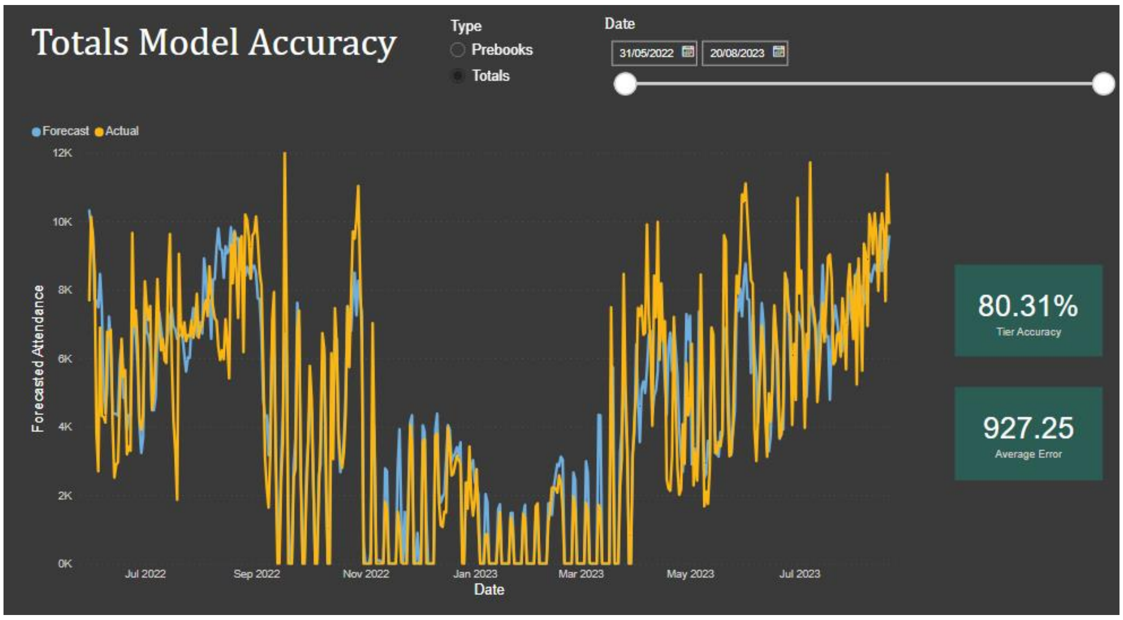
Date	AP	Consumer Frees	Corporate	Hotel	Promotions	Schools	Sun Frees	Trade	Web	Total
21/08/2023	1,584.51	846.06	8.61	1,240.15	1,785.77	125.24	115.01	1,216.60	2,187.79	9,109.74
22/08/2023	1,753.71	936.28	34.10	1,172.84	1,932.40	127.18	82.25	1,417.20	2,309.92	9,765.88
23/08/2023	1,691.50	880.81	31.74	1,167.60	2,075.36	162.78	84.44	1,722.84	2,598.35	10,415.42
24/08/2023	1,549.44	970.42	141.78	1,263.31	1,960.03	154.89	81.98	1,386.98	2,567.05	10,075.88
25/08/2023	1,413.54	885.72	19.17	1,254.11	1,849.23	154.85	77.88	1,360.79	3,079.15	10,094.44
26/08/2023	1,933.47	986.57	111.12	1,311.83	1,756.45	102.95	85.00	1,660.43	3,652.49	11,600.31
27/08/2023	2,021.03	1,049.21	17.05	1,311.50	2,131.84	91.04	85.18	1,303.44	3,592.14	11,602.43
28/08/2023	1,358.80	959.75	13.18	1,312.25	2,161.06	87.79	94.66	1,200.33	2,947.35	10,135.17
29/08/2023	1,668.76	974.30	10.53	1,247.28	2,272.98	77.70	89.60	1,316.11	3,170.88	10,828.14
30/08/2023	1,893.91	1,036.50	27.53	1,236.27	2,300.12	87.82	93.94	1,494.32	3,497.64	11,668.05
31/08/2023	1,874.53	1,152.20	22.30	1,280.73	2,468.42	68.48	90.08	1,243.00	3,485.35	11,685.09
01/09/2023	1,860.79	998.67	31.10	1,199.32	1,994.72	78.99	132.07	1,175.80	2,695.52	10,166.98
02/09/2023	2,135.36	1,095.25	75.45	1,176.67	2,297.01	63.33	139.25	1,364.91	3,219.93	11,567.16
03/09/2023	2,208.05	1,041.61	461.74	1,126.69	1,963.38	77.05	192.54	732.26	2,421.04	10,224.36

On this page, we will be showing the forecasted values split by channel for the next two weeks.

1. Line Graph Visual: Another line graph visual here with less complexity to the previous ones. This will only have the forecasted numbers on the Y-axis with an error band, with dates on the X-axis.
2. Slicer: A slicer for channel type to check each separate channel's forecasted numbers for the next two weeks.

3. Table: A table showing total attendance values split by channel per date for easier access to specific numbers. This visual has been disconnected from the filter on the same page as we want it shown at all times.

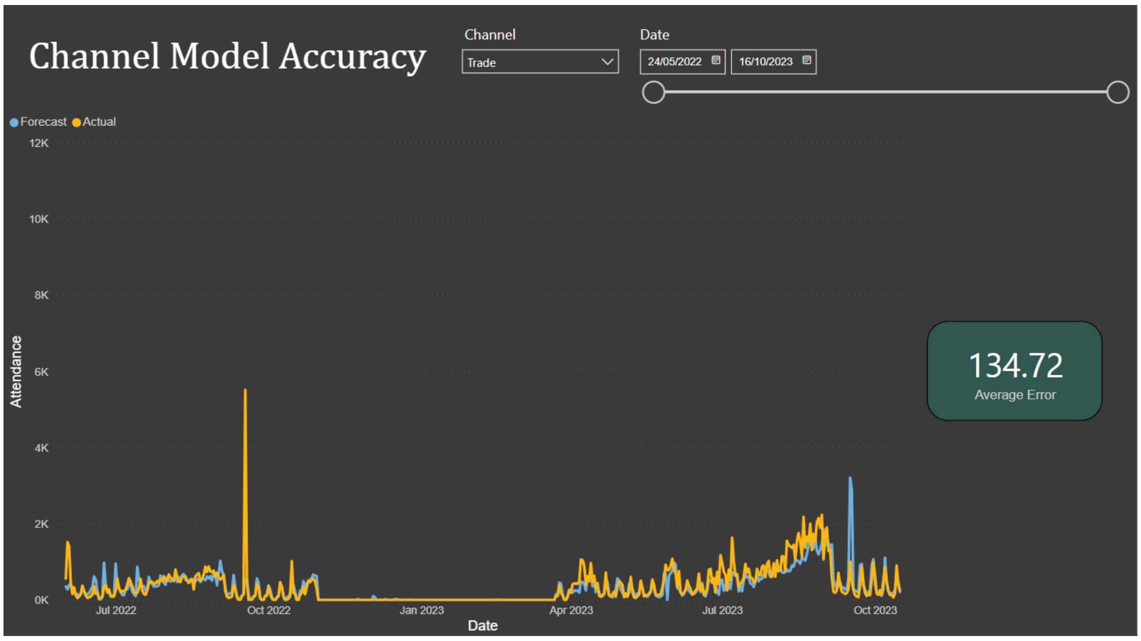
Page 3: Past Forecasts



Here, we display all values forecasted in the past. These range from may 2022 to the day prior of the upcoming forecast.

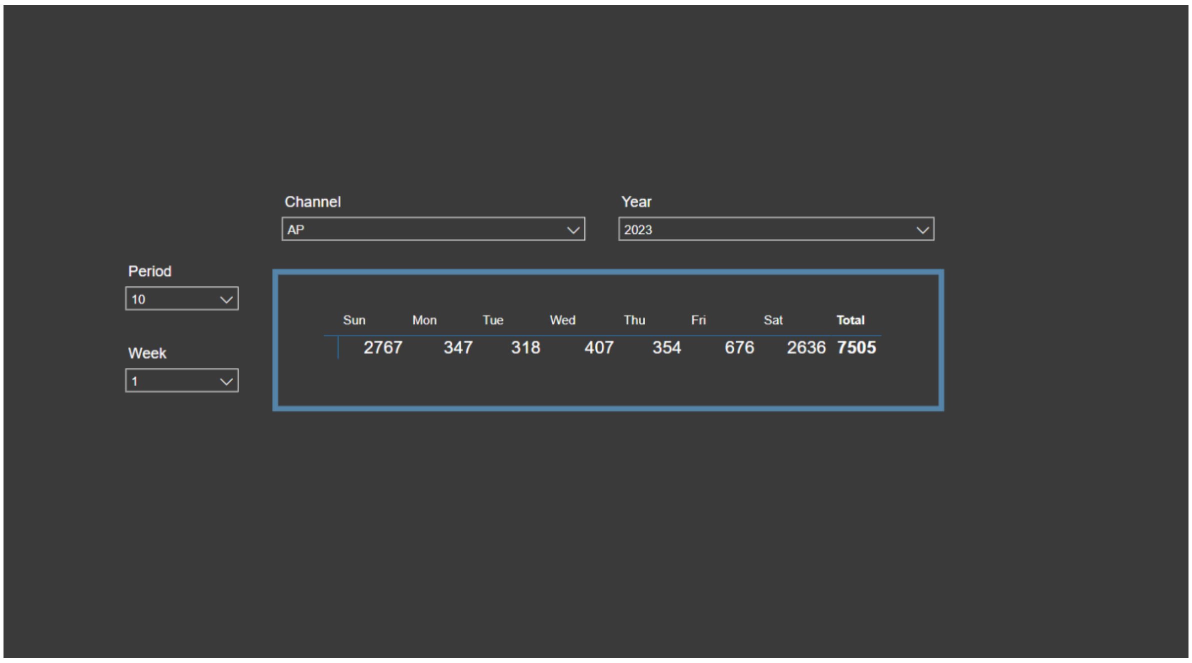
1. Line graph visual: this visual will show the forecasted values and the actual attendance values throughout time.
2. Cards: two cards, one for tier accuracy (percentage of dates where the forecast correctly predicted the attendance tier), and one for mean absolute error (measured by using the "Difference" column calculated earlier, and getting the average of this column)
3. Slicers: two slicers on this page, one for method of prediction (totals or prebooks) and one for date, using a date slider.

Page 4: Past Channel Forecasts



This page is somewhat similar to the previous, where the only difference is that there is no tier accuracy, as it cannot be measured when split by channel. Furthermore, instead of a filter for prediction method, there will be a filter for Sales Channel, enabling the user to select which channel they want to view historical forecasts for. The Average error card remains, but measures the average error for each sales channel and then averages these values out.

Page 5: Historical Attendance



Sun	Mon	Tue	Wed	Thu	Fri	Sat	Total
2767	347	318	407	354	676	2636	**7505**

This page shows the historical attendance in a format more legible and usable for the finance team at Chessington. This is because attendance values will be split by financial periods.

1. Table: the table will show actual attendance values from Sunday to Monday
2. Slicers: There are various slicers here, including sales channel, year, financial period and financial week.

Implementation & Testing

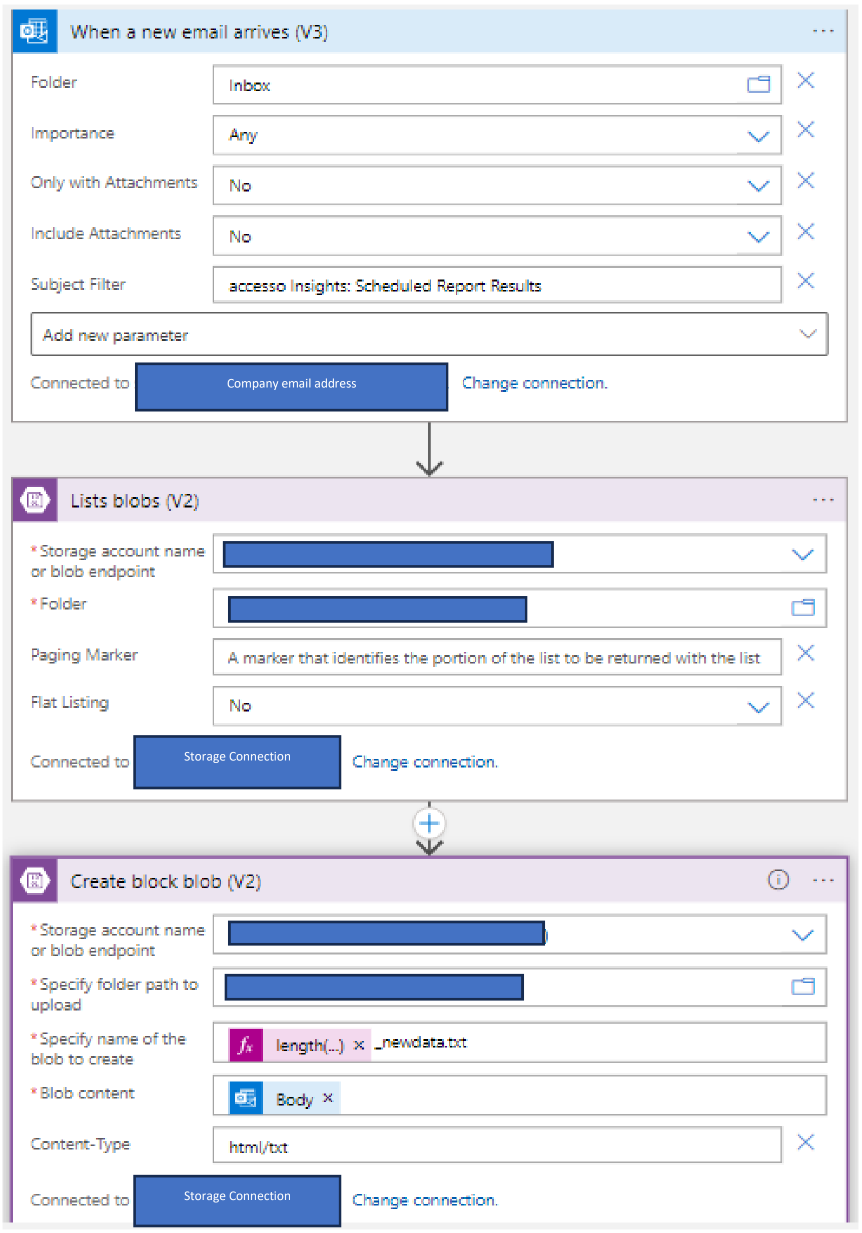
After deploying this dashboard, we have been receiving feedback and questions from both the financial team at Chessington and the pricing team at BCG. Since these departments are using the model for separate use-cases, they often request added visuals or features which sometimes must be pushed back. For example, the financial team has requested to predict revenue by using the channels forecast, as we know the pricing for each channel ticket. However, this has been pushed back, as our model should not be used specifically for revenue prediction purposes but should instead be an all-encompassing tool which separate departments can use. Throughout the testing, we've also been able to notice which of the two methods is more accurate, and can therefore make a decision later, with user feedback included, on which method to use going forward.

Conclusion

The main results we see from our model are an 85% accuracy in pricing tier predictions, with a mean average error of 760 guests with an average attendance of 6435 using the Prebooks method (meaning a 11.8% error rate in park attendance). We plan to reduce this error down to below 10% as the model learns from new data every day. The dashboard is currently the 35th most used Power BI dashboard out of 417 within the organization with 11 unique users. The outcome of the Merlin Foresight model is a widespread interest in the power of machine learning for business insights throughout Merlin. This product will soon be rolled out to more attractions, as the interest for it has been growing exponentially as it gets more and more accurate. The new Chief Digital & Data Officer has in fact asked for a worldwide rollout in the next couple of financial quarters, which will require more support from other data scientists and data engineers. This dashboard can be used as a template for other parks as well but will still require a few minor adjustments. While rolling out new and bigger parks, the unique users of the product will increase exponentially.

Rolling out to new attractions will require more work than expected due to the different needs and use cases of the model for each park and cannot simply be copied and pasted. To date, the model has increased the revenue by an average of 56% on days where pricing action was enacted from recommendations given in the dashboard. This model has also worked as a catalyst in the business to bring in more in house development of machine learning tools and other software which we have been outsourcing for a long time, costing the business a fortune. This has also led to new data scientists being hired to carry on the work for these models and more. Next steps also include the productionalizing the code that runs the model, to ease the roll out of future products. Furthermore, thorough documentation will have to be put in place to ease the handover to future owners of the product.

Appendix



When a new email arrives (V3)

Folder	Inbox
Importance	Any
Only with Attachments	No
Include Attachments	No
Subject Filter	accesso Insights: Scheduled Report Results

Add new parameter

Connected to Company email address Change connection.

Lists blobs (V2)

*Storage account name or blob endpoint	[redacted]
*Folder	[redacted]
Paging Marker	A marker that identifies the portion of the list to be returned with the list
Flat Listing	No

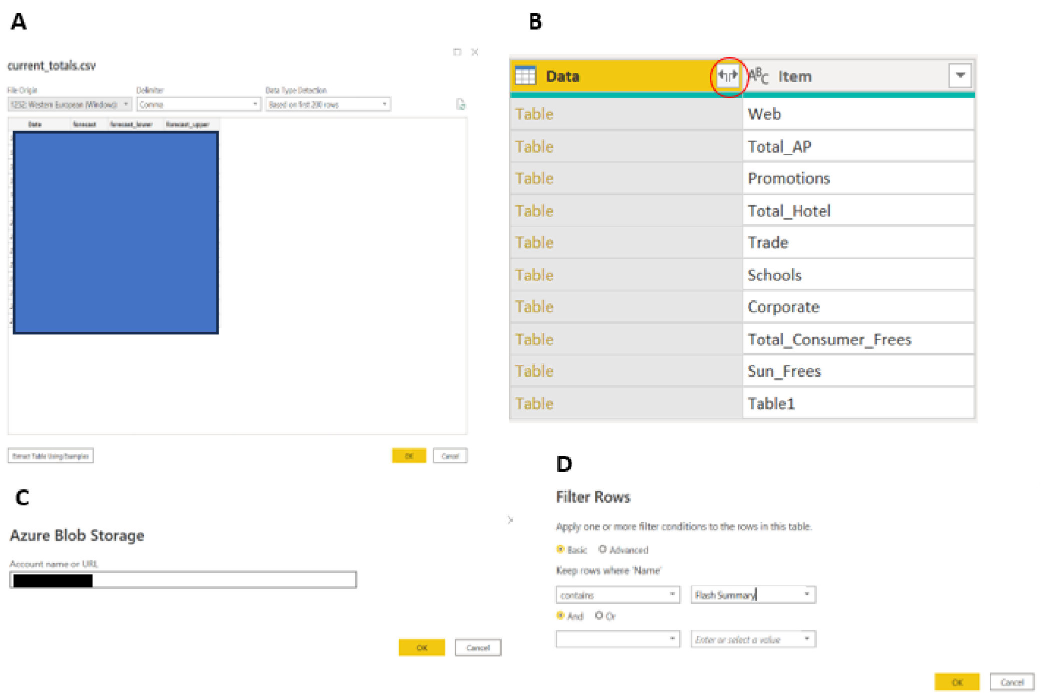
Connected to Storage Connection Change connection.

Create block blob (V2)

*Storage account name or blob endpoint	[redacted]
*Specify folder path to upload	[redacted]
*Specify name of the blob to create	length(...) × _newdata.txt
*Blob content	Body ×
Content-Type	html/txt

Connected to Storage Connection Change connection.

Figure 1: Azure Logic App. Step 1: New email arrives with subject “accesso Insights: Scheduled Report Results”. Step 2: List number of files within output folder. Step 3: convert email to .txt and change name using length of list from previous step (to avoid overwrites). Connection strings and keys have been covered for data privacy.



A

current_totals.csv

File Origin: 1252: Western European (Windows)
Delimiter: Comma
Data Type Detection: Based on first 200 rows

Date	forecast	forecast_lower	forecast_upper
[redacted]	[redacted]	[redacted]	[redacted]

B

Data	Item
Table	Web
Table	Total_AP
Table	Promotions
Table	Total_Hotel
Table	Trade
Table	Schools
Table	Corporate
Table	Total_Consumer_Frees
Table	Sun_Frees
Table	Table1

C

Azure Blob Storage

Account name or URL: [redacted]

D

Filter Rows

Apply one or more filter conditions to the rows in this table.

[✓ Basic] Advanced

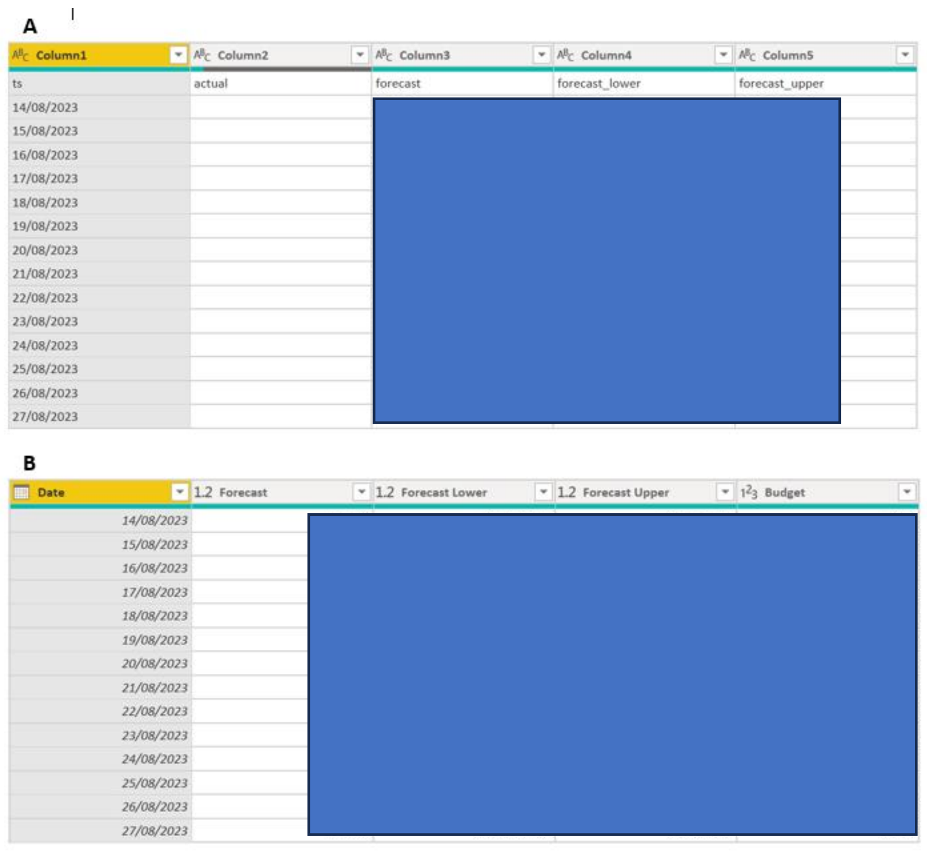
Keep rows where 'Name'

contains | Flash Summary

[✓ And] Or

Enter or select a value

Figure 2: Connecting to data through different sources. A: connecting to .csv files and loading them into Power Query. B: Combining multiple excel sheets into one table. C: Connecting to Azure Blob Storage using account name. D: Filtering out unwanted files from blob storage.



A

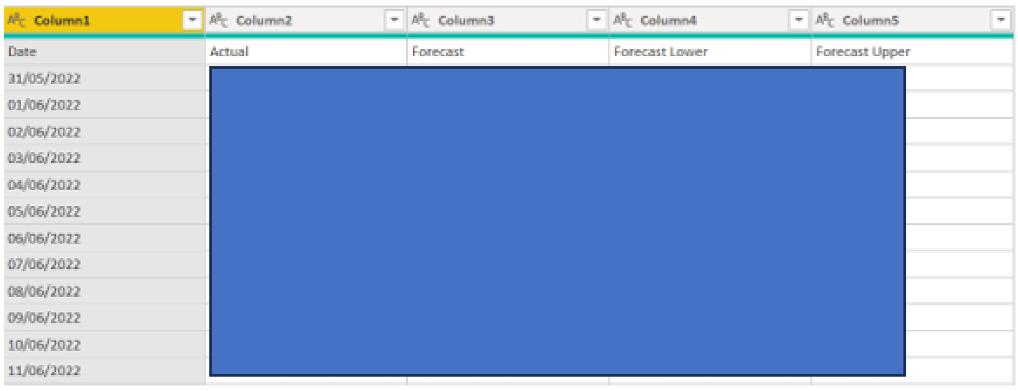
Column1	Column2	Column3	Column4	Column5
ts	actual	forecast	forecast_lower	forecast_upper
14/08/2023		[redacted]	[redacted]	
15/08/2023		[redacted]	[redacted]	
16/08/2023		[redacted]	[redacted]	
17/08/2023		[redacted]	[redacted]	
18/08/2023		[redacted]	[redacted]	
19/08/2023		[redacted]	[redacted]	
20/08/2023		[redacted]	[redacted]	
21/08/2023		[redacted]	[redacted]	
22/08/2023		[redacted]	[redacted]	
23/08/2023		[redacted]	[redacted]	
24/08/2023		[redacted]	[redacted]	
25/08/2023		[redacted]	[redacted]	
26/08/2023		[redacted]	[redacted]	
27/08/2023		[redacted]	[redacted]	

B

Date	Forecast	Forecast Lower	Forecast Upper	Budget
14/08/2023	[redacted]	[redacted]	[redacted]	[redacted]
15/08/2023	[redacted]	[redacted]	[redacted]	[redacted]
16/08/2023	[redacted]	[redacted]	[redacted]	[redacted]
17/08/2023	[redacted]	[redacted]	[redacted]	[redacted]
18/08/2023	[redacted]	[redacted]	[redacted]	[redacted]
19/08/2023	[redacted]	[redacted]	[redacted]	[redacted]
20/08/2023	[redacted]	[redacted]	[redacted]	[redacted]
21/08/2023	[redacted]	[redacted]	[redacted]	[redacted]
22/08/2023	[redacted]	[redacted]	[redacted]	[redacted]
23/08/2023	[redacted]	[redacted]	[redacted]	[redacted]
24/08/2023	[redacted]	[redacted]	[redacted]	[redacted]
25/08/2023	[redacted]	[redacted]	[redacted]	[redacted]
26/08/2023	[redacted]	[redacted]	[redacted]	[redacted]
27/08/2023	[redacted]	[redacted]	[redacted]	[redacted]

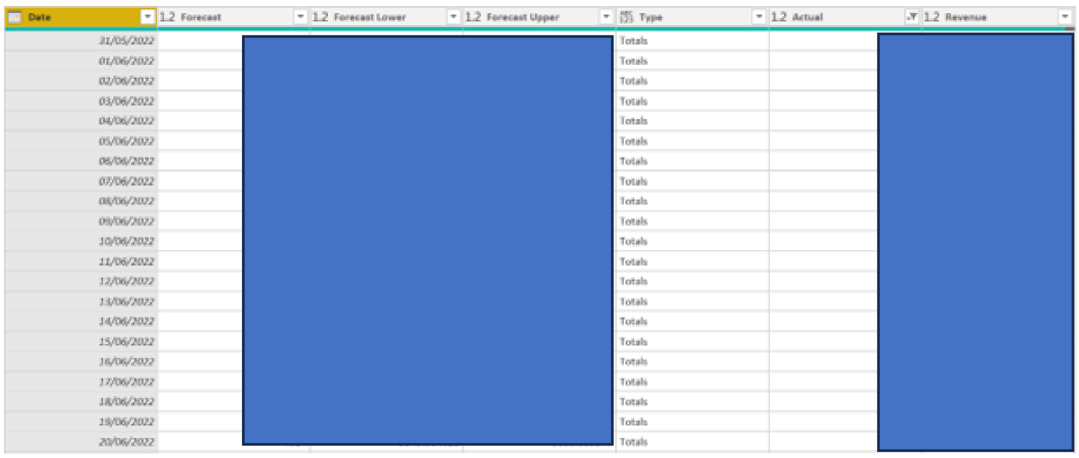
Figure 3: TotAttendance_forecast & TotPrebooks_forecast.csv pre (A) and post (B) transformation. Budget column merged on left.

A



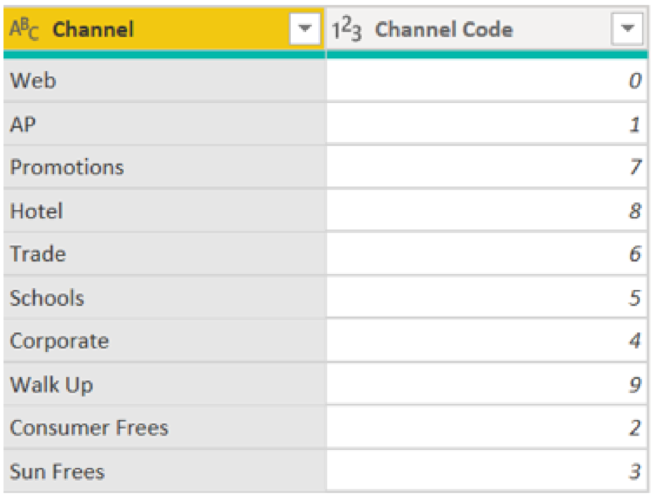
Column1	Column2	Column3	Column4	Column5
Date	Actual	Forecast	Forecast Lower	Forecast Upper
31/05/2022	[redacted]	[redacted]	[redacted]	[redacted]
01/06/2022	[redacted]	[redacted]	[redacted]	[redacted]
02/06/2022	[redacted]	[redacted]	[redacted]	[redacted]
03/06/2022	[redacted]	[redacted]	[redacted]	[redacted]
04/06/2022	[redacted]	[redacted]	[redacted]	[redacted]
05/06/2022	[redacted]	[redacted]	[redacted]	[redacted]
06/06/2022	[redacted]	[redacted]	[redacted]	[redacted]
07/06/2022	[redacted]	[redacted]	[redacted]	[redacted]
08/06/2022	[redacted]	[redacted]	[redacted]	[redacted]
09/06/2022	[redacted]	[redacted]	[redacted]	[redacted]
10/06/2022	[redacted]	[redacted]	[redacted]	[redacted]
11/06/2022	[redacted]	[redacted]	[redacted]	[redacted]

B



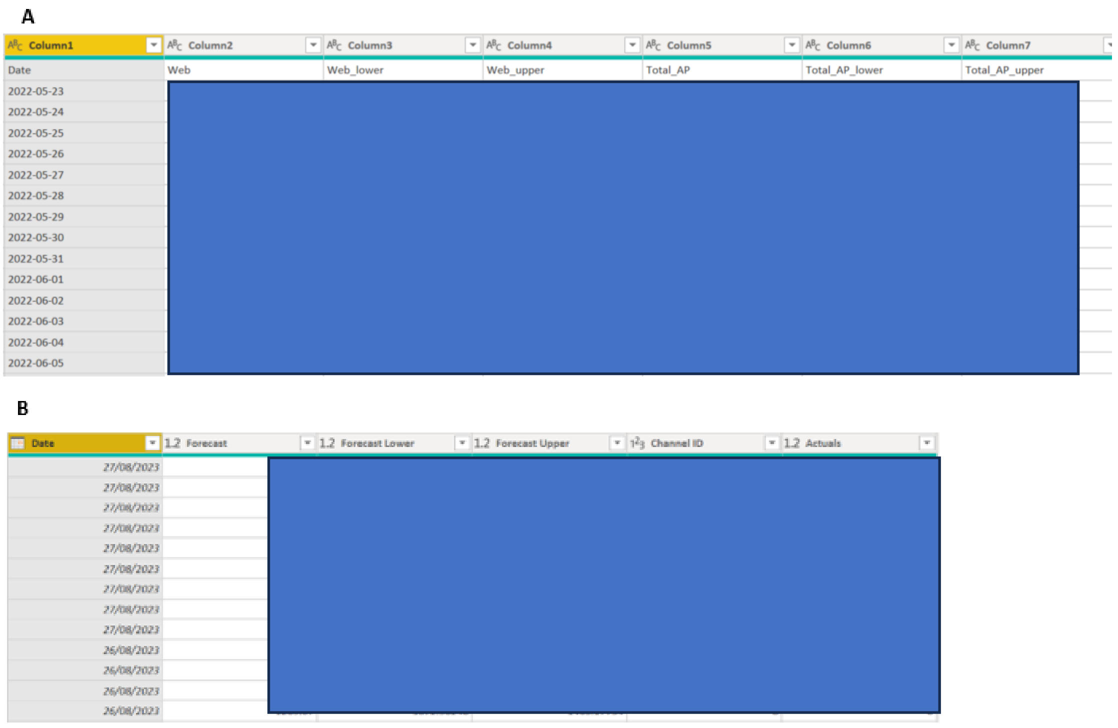
Date	Forecast	Forecast Lower	Forecast Upper	Type	Actual	Revenue
31/05/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
01/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
02/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
03/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
04/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
05/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
06/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
07/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
08/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
09/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
10/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
11/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
12/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
13/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
14/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
15/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
16/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
17/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
18/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
19/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]
20/06/2022	[redacted]	[redacted]	[redacted]	Totals	[redacted]	[redacted]

Figure 4: A) Initial table format for historical forecasts (prebooking and attendance). B) Final result after transforming both tables, adding a custom column "Type" and appending one to another.



Channel	Channel Code
Web	0
AP	1
Promotions	7
Hotel	8
Trade	6
Schools	5
Corporate	4
Walk Up	9
Consumer Frees	2
Sun Frees	3

Figure 5: Creating a dictionary of channel types. Referenced initial table (A) and removed duplicates from new table, as well as code column. Added a numbered channel code column (B).



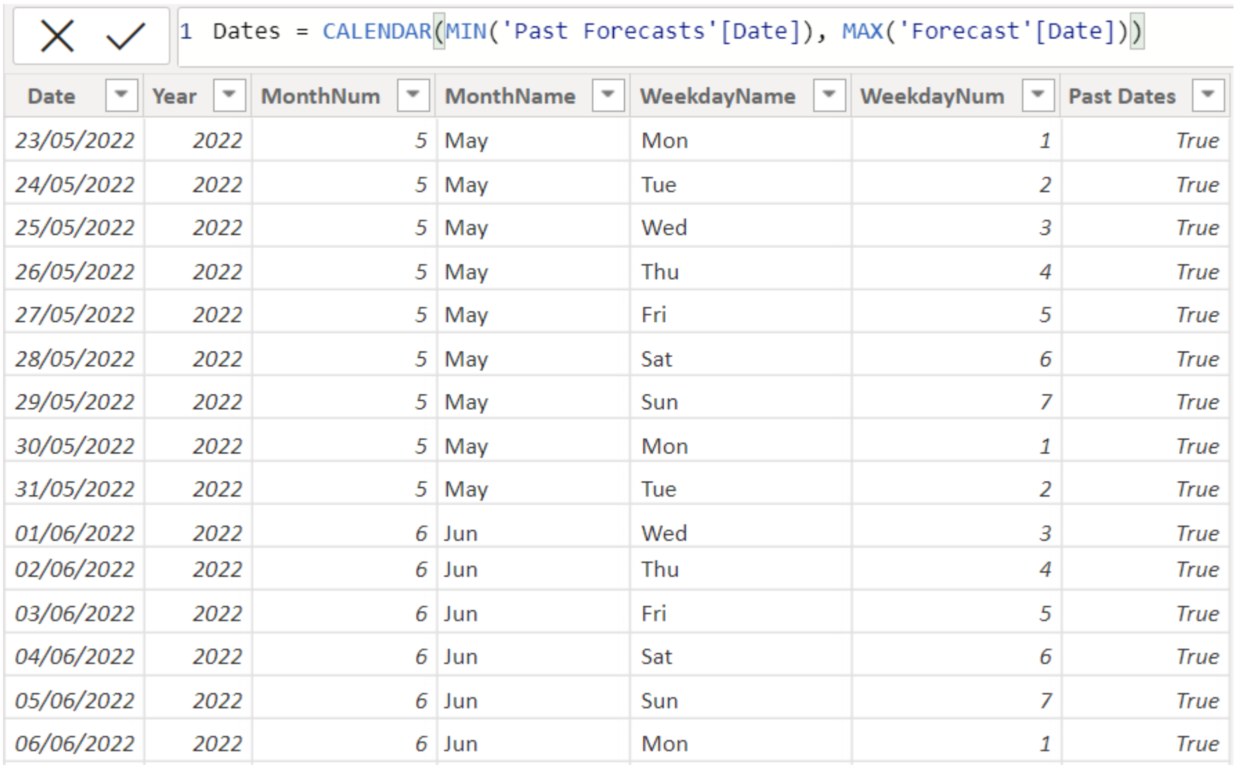
A

Column1	Column2	Column3	Column4	Column5	Column6	Column7
Date	Web	Web_lower	Web_upper	Total_AP	Total_AP_lower	Total_AP_upper
2022-05-23	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2022-05-24	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2022-05-25	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2022-05-26	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2022-05-27	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2022-05-28	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2022-05-29	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2022-05-30	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2022-05-31	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2022-06-01	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2022-06-02	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2022-06-03	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2022-06-04	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
2022-06-05	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

B

Date	Forecast	Forecast Lower	Forecast Upper	Channel ID	Actuals
27/08/2023	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
27/08/2023	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
27/08/2023	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
27/08/2023	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
27/08/2023	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
27/08/2023	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
27/08/2023	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
27/08/2023	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
27/08/2023	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
26/08/2023	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
26/08/2023	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
26/08/2023	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
26/08/2023	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]

Figure 6: Initial table format for channels.xlsx (A). Final result after transformation and merging channel ID onto type (B).



```
Dates = CALENDAR(MIN('Past Forecasts'[Date]), MAX('Forecast'[Date]))
```

Date	Year	MonthNum	MonthName	WeekdayName	WeekdayNum	Past Dates
23/05/2022	2022	5	May	Mon	1	True
24/05/2022	2022	5	May	Tue	2	True
25/05/2022	2022	5	May	Wed	3	True
26/05/2022	2022	5	May	Thu	4	True
27/05/2022	2022	5	May	Fri	5	True
28/05/2022	2022	5	May	Sat	6	True
29/05/2022	2022	5	May	Sun	7	True
30/05/2022	2022	5	May	Mon	1	True
31/05/2022	2022	5	May	Tue	2	True
01/06/2022	2022	6	Jun	Wed	3	True
02/06/2022	2022	6	Jun	Thu	4	True
03/06/2022	2022	6	Jun	Fri	5	True
04/06/2022	2022	6	Jun	Sat	6	True
05/06/2022	2022	6	Jun	Sun	7	True
06/06/2022	2022	6	Jun	Mon	1	True

Figure 7: Dates table with DAX formula. Created calendar table using minimum and maximum dates of forecast tables as start and end dates. Past forecast dates start in may 2022, while upcoming forecasts end in august 2023.

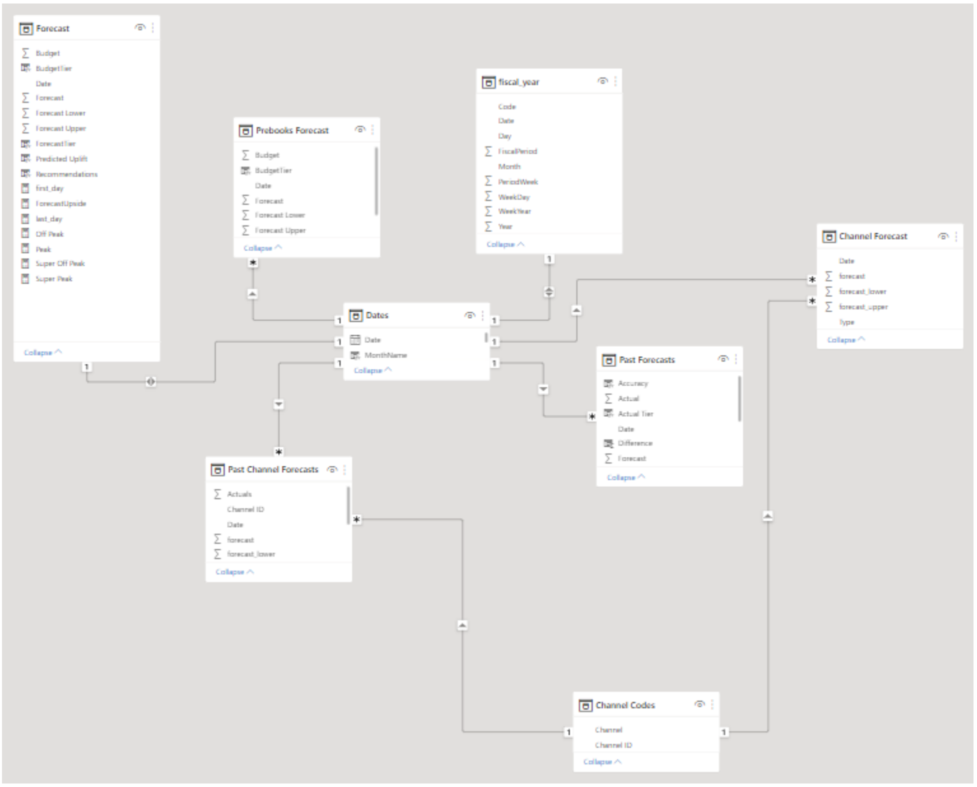
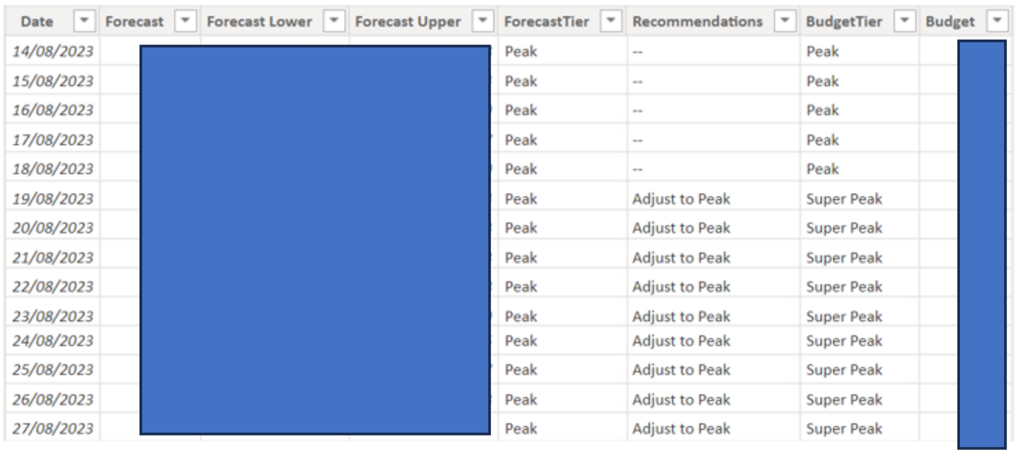


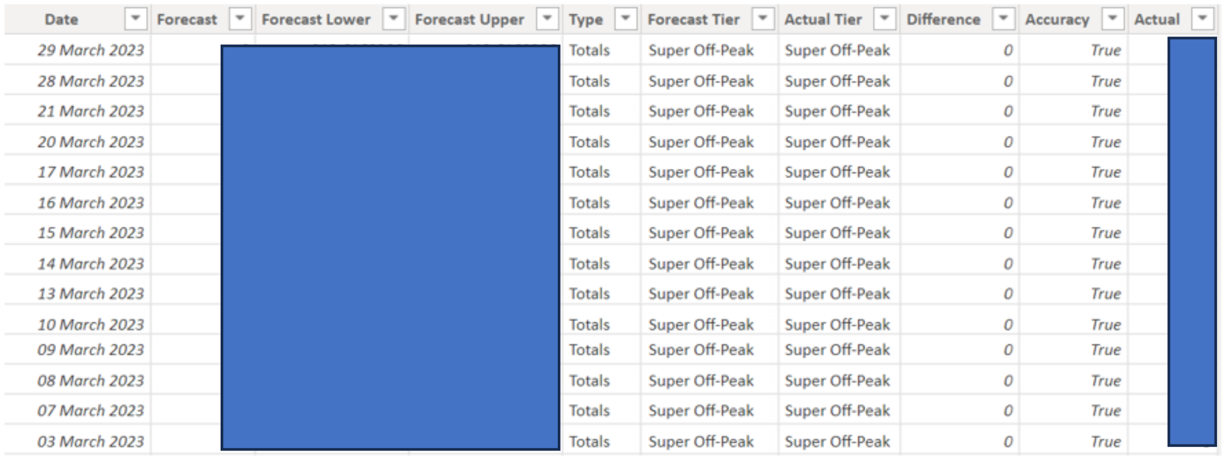
Figure 8: Relationships model. Based on Kimball-style organization, all connected to central dates column. Channel codes used as a dictionary table and therefore does not contain dates.

A



Date	Forecast	Forecast Lower	Forecast Upper	ForecastTier	Recommendations	BudgetTier	Budget
14/08/2023	[redacted]	[redacted]	[redacted]	Peak	--	Peak	[redacted]
15/08/2023	[redacted]	[redacted]	[redacted]	Peak	--	Peak	[redacted]
16/08/2023	[redacted]	[redacted]	[redacted]	Peak	--	Peak	[redacted]
17/08/2023	[redacted]	[redacted]	[redacted]	Peak	--	Peak	[redacted]
18/08/2023	[redacted]	[redacted]	[redacted]	Peak	--	Peak	[redacted]
19/08/2023	[redacted]	[redacted]	[redacted]	Peak	Adjust to Peak	Super Peak	[redacted]
20/08/2023	[redacted]	[redacted]	[redacted]	Peak	Adjust to Peak	Super Peak	[redacted]
21/08/2023	[redacted]	[redacted]	[redacted]	Peak	Adjust to Peak	Super Peak	[redacted]
22/08/2023	[redacted]	[redacted]	[redacted]	Peak	Adjust to Peak	Super Peak	[redacted]
23/08/2023	[redacted]	[redacted]	[redacted]	Peak	Adjust to Peak	Super Peak	[redacted]
24/08/2023	[redacted]	[redacted]	[redacted]	Peak	Adjust to Peak	Super Peak	[redacted]
25/08/2023	[redacted]	[redacted]	[redacted]	Peak	Adjust to Peak	Super Peak	[redacted]
26/08/2023	[redacted]	[redacted]	[redacted]	Peak	Adjust to Peak	Super Peak	[redacted]
27/08/2023	[redacted]	[redacted]	[redacted]	Peak	Adjust to Peak	Super Peak	[redacted]

B



Date	Forecast	Forecast Lower	Forecast Upper	Type	Forecast Tier	Actual Tier	Difference	Accuracy	Actual
29 March 2023	[redacted]	[redacted]	[redacted]	Totals	Super Off-Peak	Super Off-Peak	0	True	[redacted]
28 March 2023	[redacted]	[redacted]	[redacted]	Totals	Super Off-Peak	Super Off-Peak	0	True	[redacted]
21 March 2023	[redacted]	[redacted]	[redacted]	Totals	Super Off-Peak	Super Off-Peak	0	True	[redacted]
20 March 2023	[redacted]	[redacted]	[redacted]	Totals	Super Off-Peak	Super Off-Peak	0	True	[redacted]
17 March 2023	[redacted]	[redacted]	[redacted]	Totals	Super Off-Peak	Super Off-Peak	0	True	[redacted]
16 March 2023	[redacted]	[redacted]	[redacted]	Totals	Super Off-Peak	Super Off-Peak	0	True	[redacted]
15 March 2023	[redacted]	[redacted]	[redacted]	Totals	Super Off-Peak	Super Off-Peak	0	True	[redacted]
14 March 2023	[redacted]	[redacted]	[redacted]	Totals	Super Off-Peak	Super Off-Peak	0	True	[redacted]
13 March 2023	[redacted]	[redacted]	[redacted]	Totals	Super Off-Peak	Super Off-Peak	0	True	[redacted]
10 March 2023	[redacted]	[redacted]	[redacted]	Totals	Super Off-Peak	Super Off-Peak	0	True	[redacted]
09 March 2023	[redacted]	[redacted]	[redacted]	Totals	Super Off-Peak	Super Off-Peak	0	True	[redacted]
08 March 2023	[redacted]	[redacted]	[redacted]	Totals	Super Off-Peak	Super Off-Peak	0	True	[redacted]
07 March 2023	[redacted]	[redacted]	[redacted]	Totals	Super Off-Peak	Super Off-Peak	0	True	[redacted]
03 March 2023	[redacted]	[redacted]	[redacted]	Totals	Super Off-Peak	Super Off-Peak	0	True	[redacted]

Figure 9: Upcoming (A) and historical (B) forecasts. Identical tables split by channels also exist. Added tier, recommendations, difference, and accuracy columns.