01/08/2024

# Long-Term Forecasting & Explainability

## EPA Assessment Method 1

*Word Count: 3833*

Sergio Antonelli
MERLIN ENTERTAINMENTS

# Contents

## Introduction

Merlin is currently undergoing a digital transformation. One of the key points behind the transformation is focusing on data and making processes more efficient. One aspect of this was a two-week short-term forecasting project for guest visitation at all our attractions. The main purpose of this project was to make pricing adjustments, which are usually made no longer than two weeks out. These pricing adjustments are usually made from a fixed price, decided at the start of the year using something called “budget”, which is what the business is aiming for, and what it uses to make operational decisions. However, there are two main issues with budget: the first is the inaccuracy in the budget calculations. The second is the lengthy process that revenue analysts go through to calculate these. The calculation consists in a sort of intuition, based on past year performance in specific seasons, as well as inbound tourism and economic factors, and many other data sources. All of these are excel files, where very basic calculations are used to compute an approximate budget for attendance and revenue. The business therefore wants to come up with an alternative, more data-driven way to measure these budgets, which will both improve accuracy and improve efficiency, speeding up the process by removing hours of weekly work for analysts.

## Scope & Project Outcomes

This project is a POC (proof of concept) for a larger scope. This therefore means that we need to keep the requirements as basic as possible to be able to complete what we need to prove the usefulness of such a tool. The deadline for this POC is the start of august for a basic build of both an explanatory and forecasting model, as well as a template for a power bi tool displaying the results for easy access by revenue managers. This extends to the end of august for a more robust product for the POC locations. We have landed on LEGOLAND Florida as a starting point for the POC, with the London Cluster as a stretch goal. The main project outcomes are:

1. An explanatory model, to give a clear idea of the key indicators to performance for both revenue and attendance, so that the business knows what factors can and can’t be influenced to improve business performance.
2. A weekly forecasting model twelve months out, which the business will use for operational improvements and preparing for busier/quieter periods throughout the year, as well as revenue planning and predictions for shareholders.
3. A power bi dashboard depicting both models week by week throughout the year, so that revenue analysts can have quick and easy to use access to this information.

## Project Plan

The plan for the project consists in the following:

1. Week 1: build a development environment for the product and set up a storage for the model results
2. Week 2: pick 5 key data sources for the model and initial exploration of this data
3. Week 3: pulling key data sources in and transform in usable format
4. Week 4: Research & build explanatory model using key data sources

5. Week 5: Use findings of explanatory model to build initial forecasting model for next 12 months
6. Week 6: Build a power bi dashboard with the key findings

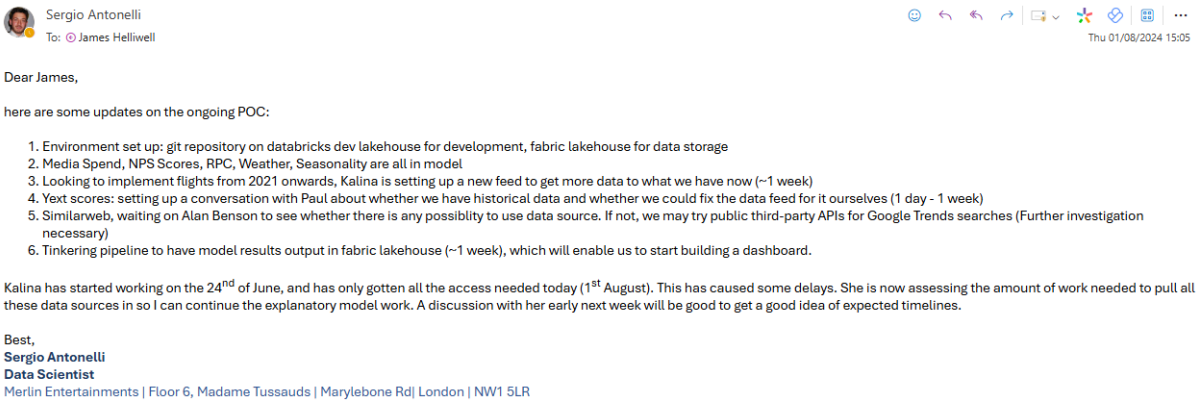
## Legislation, Organisational Policies & Procedures

To maintain data protection standards throughout any data processing activity, the Merlin tech team has a data protection team that oversees any projects that involve the use of personal data. If a project uses any personal data, a ROPA must be completed (Record of Processing Activity), where all the personal data processing is recorded. Following the RoPA, if there is high risk personal data processing (for example, the processing of large volumes of employee payroll data), according to Article 35(1) in the UK GDPR, a DPIA (Data Protection Impact Assessment) must be completed by the contract owner. Thankfully, none of the data sources that have been selected contain any personal data. However, we have been discussing the possibility of adding Mastercard payments data in the model if the POC gets approval, which will require anonymization and a DPIA. All the data I will be using is classified as internal use only. For the POC, none can be considered highly confidential and is usually available for most employees dealing with data at Merlin.

In line with the new company policy, I will be using a workspace set up in Databricks within a git repository that I have set up with the help of the data platform team. Since this is a POC, I will mostly be using fabric Onelake as a source and storage for data. However, once the data platform is fully built, we will focus on moving our processes fully to Databricks using Unity Catalog.

## Team Structure

This POC was requested by our Chief Strategic Officer, Linda Zhou. I was put on this project by the Chief Data Officer, Kinnari Linda. The lead for this project will be James Helliwell, who works in the finance department at Merlin. I will report to him, and he will feed information back to Linda. Figure 1 shows an example of a summary I wrote up for him to feed back to Linda.



Sergio Antonelli
To: James Helliwell
Thu 01/08/2024 15:05

Dear James,

here are some updates on the ongoing POC:

1. Environment set up: git repository on databricks dev lakehouse for development, fabric lakehouse for data storage
2. Media Spend, NPS Scores, RPC, Weather, Seasonality are all in model
3. Looking to implement flights from 2021 onwards, Kalina is setting up a new feed to get more data to what we have now (~1 week)
4. Yext scores: setting up a conversation with Paul about whether we have historical data and whether we could fix the data feed for it ourselves (1 day - 1 week)
5. Similarweb, waiting on Alan Benson to see whether there is any possiblity to use data source. If not, we may try public third-party APIs for Google Trends searches (Further investigation necessary)
6. Tinkering pipeline to have model results output in fabric lakehouse (~1 week), which will enable us to start building a dashboard.

Kalina has started working on the 24nd of June, and has only gotten all the access needed today (1st August). This has caused some delays. She is now assessing the amount of work needed to pull all these data sources in so I can continue the explanatory model work. A discussion with her early next week will be good to get a good idea of expected timelines.

Best,
**Sergio Antonelli**
**Data Scientist**
Merlin Entertainments | Floor 6, Madame Tussauds | Marylebone Rd| London | NW1 5LR

*Figure 1: Email summary of current project progress and potential next steps*

The data engineer on this project, Kalina Todorova, was only onboarded later on, and will help me make the pipelines more robust and in line with Merlin strategy after the POC is completed and approved. Prior to her start, I helped get her access to databricks and to my git repo through the help of the data platform team. An example of this sort of exchange is in figure 2.

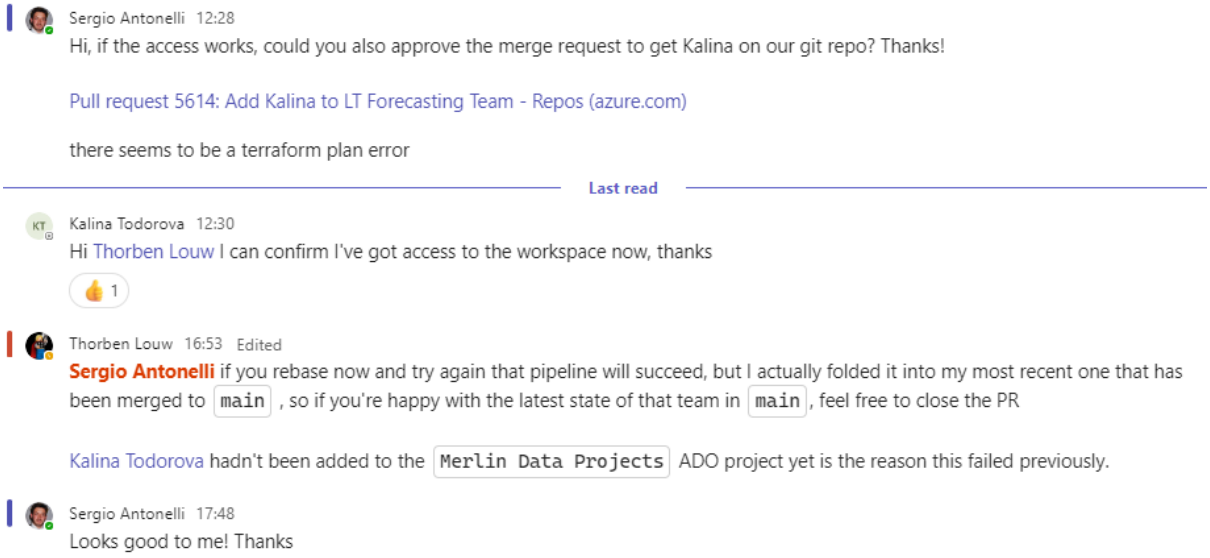



*Figure 2: teams communication with data platform team to get data engineer access to required tools*

## Development Environment & Repo Structure

For the development of this product, I will be using Azure Databricks, which easily connects to other azure products, which will make it easy to pull in most of our data sources. To keep the code robust, and avoid breaking it to the point of no repair, I set up a git repository. Figure 3 shows the structure of my git repo.

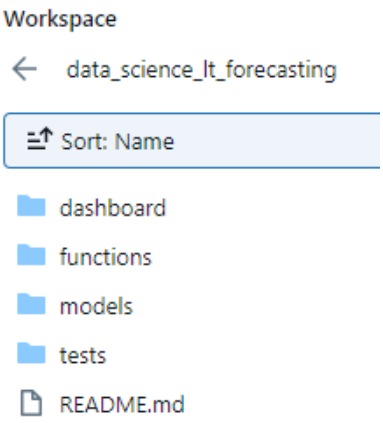



*Figure 3: git repo folder structure*

The dashboard folder will contain my power bi dashboard and connect to a fabric workspace. The models folder will contain all the code for my models. The config file within the models folder is a python file with a list of classes, one global config class and then one config class for each attraction. Each configuration for the attractions will contain information such as the name of the attraction, the start and end date for the training data, a list of variables for the explanatory model, and various other variables that feed into the pipeline. All my functions are within three separate files within the functions folder. One script contains all helper functions to build the models, one is for pre-transformation work, and one is for grouping all my functions together in one pipeline. The product_pipeline notebook calls the pipeline functions for all attractions by feeding in their respective config classes.

## Data Transformation

I have two dependent variables, for two separate models: Visitors and Revenue. This data is stored in a fabric onelake table which I pull into databricks to carry out my transformation in pyspark (figure 4):

```
def join_dimensions(spark, global_config):

    channels = spark.read.format('delta').load(global_config.channelsPath)
    attractions = spark.read.format('delta').load(global_config.attractionsPath)
    calendar = spark.read.format('delta').load(global_config.calendarPath)
    trading_flash=spark.read.format('delta').load(global_config.tradingPath)

    trading_flash = trading_flash.withColumn("Year", substring("Attribute", 1,4).cast("int"))
    split_col = split(trading_flash['Attribute'], ' ')
    trading_flash = trading_flash.withColumn("Scenario", split_col.getItem(1))
    trading_flash = trading_flash.withColumn("Type", split_col.getItem(2))

    joined_df = trading_flash.join(calendar, (trading_flash["Year"] == calendar["Year"]) & (trading_flash['Week'] == calendar['TM1_Week']) & (trading_flash['Day'] == calendar['Day']), 'inner') \
        .join(attractions, trading_flash['Attraction_TM1_Code'] == attractions['Attraction_TM1_Code'], 'left') \
        .join(channels, trading_flash['Account'] == channels['TM1_Account_Code'], 'left')

    df = joined_df.select(
        calendar['Date'], calendar['Year'], calendar['Day'],
        trading_flash['Scenario'], trading_flash["Type"], trading_flash["LCY"],
        attractions['Presentation_Attraction_Name'],
        attractions["Cluster"],
        channels['TM1_Account'], channels['Summary_Account_Description'], channels["Level"], channels["Sales_Channel"], channels["Sub_Account_Description"]
    )

    return df, calendar.toPandas()
```

*Figure 4: finance trading data pull from onelake & initial transformation with pyspark*

I pulled in various dimension tables, including sales channels, attractions and calendar, as well as my fact table (trading_flash). I then joined these together and carried out relevant tranformations to end up with a daily dataframe for all attractions and both Revenue and Visitors data (column “Type”).

The key data sources selected for the POC are as follows:

- Flights data

  This dataset is third-party administrative data, which contains information on flight arrivals at various destinations throughout the world. In our case, we need any arrivals to Orlando, Florida. Figure 5 is the code I used for the transformation of flights data, pulled from a fabric lakehouse into databricks:

```
def flights_pull(spark, global_config):

    flights_2023 = spark.read.format('delta').load(global_config.flightsPath2023)
    flights_2019 = spark.read.format('delta').load(global_config.flightsPath2019)

    flights_2023 = flights_2023.select("destinationCountry", "destinationCity", "originCountry", "PeriodStart", "Amount")
    flights_2019 = flights_2019.select("destinationCountry", "destinationCity", "originCountry", "PeriodStart", "Amount")

    flights = flights_2019.union(flights_2023)

    flights_grouped = flights.groupBy("PeriodStart", "destinationCountry", "destinationCity") \
        .sum("Amount") \
        .withColumnsRenamed({"sum(Amount)": "Flights", "PeriodStart" : "Date", "destinationCountry" : "Country", "destinationCity" : "City"}) \
        .select("Date", "Country", "City", "Flights")

    flights_df = flights_grouped.toPandas()

    return flights_df
```

*Figure 5: flights data pull from onelake & initial transformation with pyspark*

  This data source is very extensive and has millions of rows of data. Using pyspark enables me to do some quick transformation before converting to a pandas dataframe for further transformation within my model pipeline. The source is split into two tables, one for data from 2019 and one for data from 2023. I therefore pulled both tables using abfs paths

which I saved in my config file. I then concatenated them, before grouping the table by the dimensions needed in my pipeline. From the resulting table, I will only need to filter by the attraction name to get the desired dataframe.

- Marketing data

  This is first-party administrative data (routinely gathered), formatted as a table with total marketing spend for various attractions and types of marketing spend. This data source is also in fabric onelake and I will therefore pull it similarly to the flights data above (figure 6).

```
def media_spend_pull(spark, global_config):

    media_spend = spark.read.format('delta').load(global_config.mediaPath)
    attractions = spark.read.format('delta').load(global_config.attractionsPath)

    media = media_spend.join(attractions, media_spend['attraction'] == attractions['Presentation_Attraction_Name'], 'left') \
        .filter(media_spend["HFM_Account"] == "IS4210") \
        .select("HFM_Account", "HFM_SubAccount", "Month", "Year", "Presentation_Attraction_Name", "Cluster", "amount") \
        .groupBy("Cluster", "Presentation_Attraction_Name", "Year", "Month") \
        .sum("amount") \
        .withColumnRenamed("sum(amount)", "Media Spend") \
        .toPandas()

    return media.fillna(0)
```

*Figure 6: media spend data pull from onelake & initial transformation with pyspark*

Here, I filtered my data by the desired account (IS4210), which represents media spend. I then grouped by my desired columns and, once again, converted the dataframe to a pandas dataframe. This data source contains some NaN values, which I replaced with 0 to represent periods where the business didn't invest in marketing.

- RPC (Revenue Per Customer)

  This data is also organizational and can be pulled from a calculation of our two dependant variables, Revenue and Visitors. All I have to do is a simple division (Revenue / Visitors) to get a good estimate of ticket prices. However, I also have to split these by sales channel, which will require a pivot (see figure 7).

```
def rpc_pull(df):

    updated_df = df.filter(df["Sales_Channel"] == 1) \
    .filter((df["Summary_Account_Description"] == "Trade") | (df["Summary_Account_Description"] == "Consumer")) \
    .filter(df["Scenario"] == "Actual") \
    .groupBy("Date", "Sub_Account_Description", "Cluster", "Presentation_Attraction_Name") \
    .pivot("Type") \
    .sum("LCY")

    grouped_df = updated_df.withColumn("RPC", updated_df["Revenue"] / updated_df["Visitors"]) \
    .groupBy("Date", "Cluster", "Presentation_Attraction_Name") \
    .pivot("Sub_Account_Description") \
    .sum("RPC") \
    .toPandas()

    return grouped_df.fillna(0).add_prefix("rpc_")
```

*Figure 7: revenue per customer calculation from financial trading dataframe*

To do this, I take the output dataframe from the join_dimesions function (figure 2) and do a few filters to get the desired data. I also pivot my "Type" column to first get two columns, one for visitors and one for revenue. Now I can create a new column and measure RPC. To then split this by channel, I pivot the column "Sub_Account_Description" with RPC as values and add a prefix "rpc_", while also filling in any missing values with 0. I now have multiple columns, one for each channel, representing an estimated ticket price for each sales channel every day.

- Brand Health

  Brand health is measured using surveys at the end of a guest's experience at our parks, making it a source of research data (collected for research purposes). This data contains NPS (Net Promoter Score), Value for Money & Satisfaction. To pull these into the model, I used another fabric Onelake table (see figure 8). This follows a similar transformation to the other fabric Onelake tables.

```
def nps_pull(spark, global_config):

    nps_scores = spark.read.format('delta').load(global_config.npsPath)
    attractions = spark.read.format('delta').load(global_config.attractionsPath)

    nps = nps_scores.join(attractions, nps_scores['pbi_attraction_code'] == attractions['Attraction_TM1_Code'], 'inner') \
        .select("survey_date", "Cluster", "Presentation_Attraction_Name", "survey_kpi", "score", "surveys") \
        .withColumn("pct_score", col("score") / col("surveys")) \
        .groupBy("survey_date", "Cluster", "Presentation_Attraction_Name").pivot("survey_kpi").agg({"pct_score": "first"}) \
        .withColumnsRenamed({"survey_date": "Date",
                             "Net Promoter Score (NPS)" : "NPS",
                             "Satisfaction Top Box (Very Satisfied)" : "Satisfaction",
                             "Value for Money (Excellent)" : "Value For Money"})

    return nps.toPandas()
```

*Figure 8: net promoter score data pull from onelake & initial transformation with pyspark*

- Weather

  Weather data is a good open data source (freely available) for our explanatory model, but won't be implemented in the time-series forecast, as we cannot predict weather further out than two weeks. To get this data, I used an API from "visual crossing", which stores data in fabric Onelake in a table that I created. I can then pull this data similarly to these other data sources.

After getting my key data sources set up in my pipeline, I moved to some initial analysis of my data, including some correlation analysis and some feature engineering, depending on the data structures I have available:

- For brand health, I am only interested in Net Promoter Score as a general idea of how much a guest was satisfied with their experience, and therefore will discard Satisfaction & Value for Money scores.
- For our weather data, I will be looking at rain, cloud cover and temperature. To reduce these values down to one, I can first scale them and then perform a

```
def reduce_dimensions(df, cols, n_components, dim_name):

    agg_cols = df[cols]

    scaler = StandardScaler()
    agg_cols_scaled = scaler.fit_transform(agg_cols)

    pca = PCA(n_components=n_components)
    agg_cols_pca = pca.fit_transform(agg_cols_scaled)

    df[dim_name] = agg_cols_pca[:, 0]

    return df
```

*Figure 9: dimensionality reduction and scaling function using principal component analysis & standard scaler from scikit learn*

Principal Component Analysis (PCA) to reduce them down to one dimension. I created a function to carry out this analysis for any further data sources which may require it (figure 9).

- For RPC, I carried out the same analysis as we have a lot of different channels, and these would overshadow any other data source if inputted all together. Before the dimensionality reduction, I created a correlation matrix (figure 10) for my channel RPC values and got rid of any highly correlated (>0.8) values, which would be made redundant. I also got rid of any columns with mostly zero values, as they could introduce some noise in the model.

| | rpc_Accommodation | rpc_Annual passes | rpc_Central Corporate | rpc_Central Schools | rpc_City Passes | rpc_Consumer frees | rpc_Contact centre | rpc_Gifting | rpc_In-Market | rpc_Local Corporate | rpc_Local Groups | rpc_Local Schools | rpc_Local Travel and tour operator | rpc_Merlin online | rpc_Online Travel Agents | rpc_Other consumer | rpc_Package Tours | rpc_Retail | rpc_Third party online |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| rpc_Accommodation | 1.000000 | 0.188121 | 0.196160 | 0.688054 | -0.001018 | -0.000905 | -0.000199 | -0.001095 | 0.424405 | 0.100529 | 0.311400 | 0.247795 | -0.000376 | 0.314278 | 0.292970 | -0.000672 | 0.129148 | 0.434378 | -0.002345 |
| rpc_Annual passes | 0.188121 | 1.000000 | 0.453099 | 0.140289 | 0.118342 | -0.002533 | 0.000287 | -0.004185 | 0.461802 | 0.014622 | 0.588803 | 0.612738 | -0.003594 | 0.684467 | 0.624584 | 0.133010 | 0.287309 | 0.132695 | 0.547208 |
| rpc_Central Corporate | 0.196160 | 0.453099 | 1.000000 | 0.167824 | 0.099619 | -0.002494 | 0.000089 | -0.004170 | 0.419540 | 0.012883 | 0.519053 | 0.536045 | -0.002954 | 0.579154 | 0.560742 | 0.093234 | 0.267279 | 0.158896 | 0.459656 |
| rpc_Central Schools | 0.688054 | 0.140289 | 0.167824 | 1.000000 | -0.000529 | -0.000539 | -0.000091 | 0.002734 | 0.360423 | 0.000185 | 0.275798 | 0.251251 | -0.000775 | 0.271095 | 0.256041 | -0.000370 | 0.145211 | 0.518764 | -0.001246 |
| rpc_City Passes | -0.001018 | 0.118342 | 0.099619 | -0.000529 | 1.000000 | -0.001100 | -0.000173 | -0.001242 | 0.177802 | -0.000212 | 0.142347 | 0.154807 | -0.000728 | 0.165918 | 0.142872 | 0.016446 | 0.129746 | 0.000449 | 0.223522 |
| rpc_Consumer frees | -0.000905 | -0.002533 | -0.002494 | -0.000539 | -0.001100 | 1.000000 | -0.000216 | 0.008838 | -0.003112 | -0.000283 | -0.003010 | -0.003236 | -0.001533 | -0.004201 | -0.003804 | -0.000742 | -0.002030 | -0.001748 | -0.002576 |
| rpc_Contact centre | -0.000199 | 0.000287 | 0.000089 | -0.000091 | -0.000173 | -0.000216 | 1.000000 | -0.000381 | 0.002007 | -0.000046 | -0.000489 | 0.002408 | -0.000170 | 0.002335 | -0.000578 | -0.000125 | 0.002074 | -0.000316 | -0.000424 |
| rpc_Gifting | -0.001095 | -0.004185 | -0.004170 | 0.002734 | -0.001242 | 0.008838 | -0.000381 | 1.000000 | -0.003352 | -0.000502 | -0.005402 | -0.005645 | -0.001937 | -0.007218 | -0.005398 | -0.000993 | -0.001582 | -0.002527 | -0.004576 |
| rpc_In-Market | 0.424405 | 0.461802 | 0.419540 | 0.360423 | 0.177802 | -0.003112 | 0.002007 | -0.003352 | 1.000000 | 0.044959 | 0.561547 | 0.684270 | -0.003493 | 0.715517 | 0.647835 | 0.180826 | 0.685422 | 0.316757 | 0.135208 |
| rpc_Local Corporate | 0.100529 | 0.014622 | 0.012883 | 0.000185 | -0.000212 | -0.000283 | -0.000046 | -0.000502 | 0.044959 | 1.000000 | 0.043425 | 0.001406 | 0.001490 | 0.029042 | -0.001676 | -0.000163 | 0.022901 | 0.123922 | -0.000513 |
| rpc_Local Groups | 0.311400 | 0.588803 | 0.519053 | 0.275798 | 0.142347 | -0.003010 | -0.000489 | -0.005402 | 0.561547 | 0.043425 | 1.000000 | 0.724987 | -0.003747 | 0.810529 | 0.756712 | 0.104683 | 0.300017 | 0.247706 | 0.627596 |
| rpc_Local Schools | 0.247795 | 0.612738 | 0.536045 | 0.251251 | 0.154807 | -0.003236 | 0.002408 | -0.005645 | 0.684270 | 0.001406 | 0.724987 | 1.000000 | -0.003879 | 0.829190 | 0.773983 | 0.143326 | 0.436061 | 0.241961 | 0.522934 |
| rpc_Local Travel and tour operator | -0.000376 | -0.003594 | -0.002954 | -0.000775 | -0.000728 | -0.001533 | -0.000170 | -0.001937 | -0.003493 | 0.001490 | -0.003747 | -0.003879 | 1.000000 | -0.003905 | -0.003974 | 0.000031 | -0.002151 | -0.001736 | -0.002973 |
| rpc_Merlin online | 0.314278 | 0.684467 | 0.579154 | 0.271095 | 0.165918 | -0.004201 | 0.002335 | -0.007218 | 0.715517 | 0.029042 | 0.810529 | 0.829190 | -0.003905 | 1.000000 | 0.855485 | 0.154095 | 0.417325 | 0.255206 | 0.608942 |
| rpc_Online Travel Agents | 0.292970 | 0.624584 | 0.560742 | 0.256041 | 0.142872 | -0.003804 | -0.000578 | -0.005398 | 0.647835 | -0.001676 | 0.756712 | 0.773983 | -0.003974 | 0.855485 | 1.000000 | 0.150149 | 0.386714 | 0.234287 | 0.615126 |
| rpc_Other consumer | -0.000672 | 0.133010 | 0.093234 | -0.000370 | 0.016446 | -0.000742 | -0.000125 | -0.000993 | 0.180826 | -0.000163 | 0.104683 | 0.143326 | 0.000031 | 0.154095 | 0.150149 | 1.000000 | 0.112631 | -0.000838 | 0.215044 |
| rpc_Package Tours | 0.129148 | 0.287309 | 0.267279 | 0.145211 | 0.129746 | -0.002030 | 0.002074 | -0.001582 | 0.685422 | 0.022901 | 0.300017 | 0.436061 | -0.002151 | 0.417325 | 0.386714 | 0.112631 | 1.000000 | 0.106328 | 0.097919 |
| rpc_Retail | 0.434378 | 0.132695 | 0.158896 | 0.518764 | 0.000449 | -0.001748 | -0.000316 | -0.002527 | 0.316757 | 0.123922 | 0.247706 | 0.241961 | -0.001736 | 0.255206 | 0.234287 | -0.000838 | 0.106328 | 1.000000 | -0.003722 |
| rpc_Third party online | -0.002345 | 0.547208 | 0.459656 | -0.001246 | 0.223522 | -0.002576 | -0.000424 | -0.004576 | 0.135208 | -0.000513 | 0.627596 | 0.522934 | -0.002973 | 0.608942 | 0.615126 | 0.215044 | 0.097919 | -0.003722 | 1.000000 |

*Figure 10: correlation matrix for revenue per customer data, in order to identify high correlations and reduce inputs*

From the matrix, I can see that the only highly correlated channels are Local Groups and Merlin Online. I therefore got rid of Local Group tickets (as Merlin Online represent the majority of ticket sales and are therefore the better data source), as well as some RPCs which had many zero values (Other Consumer) and other less relevant ones such as Annual Passes, which will usually be near zero due to annual pass holders paying once a year, and Retail, which are not tickets but rather total spend per customer within the park. I then performed the same PCA analysis as earlier.

The last two sources for the POC were already in a usable format. Thanks to the nature of random forests, there is no need to scale values. However, I did impute some missing data with mean values, which is reasonable when the missing data is minimal.

## Explanatory Model

To build a robust explanatory model, which I will use for feature selection in my forecasting model, I built a random forest which takes in all data sources available to me and the total attendance/revenue for each park. The reason for this choice is that my data sources vary a lot and are not linear. A lasso approach would be good for feature selection due to its computational efficiency. However, a lot of my data isn't always linear, and contains a lot of highly correlated predictors, making this approach less relevant. A gradient boosting model would also be a good option, due to its highly accurate results thanks to its boosting mechanism and its flexibility in handling different data types. However, it is highly prone to overfitting when a model is too complex, which will be the case with a high number of very different or similar data sources that

I will be using. A random forest is instead robust to outliers, handles multicollinearity well and doesn't require linear data, therefore being the best option for my use-case.

```
def feature_importance(weekly_df, variable_lst):

    X = weekly_df.drop(columns=["y"])
    y = weekly_df["y"]

    # Fit the model
    rf = RandomForestRegressor(n_estimators=100, random_state=42)
    rf.fit(X, y)

    # Get feature importances
    importances = rf.feature_importances_
    indices = np.argsort(importances)[::-1]

    # Plot feature importances
    plt.figure(figsize=(10, 6))
    plt.title("Feature Importances")
    plt.bar(range(X.shape[1]), importances[indices], align="center")
    plt.xticks(range(X.shape[1]), X.columns[indices], rotation=90)
    plt.show()

    return indices
```

*Figure 11: random forest explanatory model build to extract feature importances*

Figure 11 shows how I fit my weekly attendance data to this model. I first split my weekly data by my predictor variables (X) and my outcome variable (y). I then created a random forest model with the correct hyperparameters, fit the data to the model, and extracted the feature importance generated by the model using coefficients. Figure 12 shows the feature importance for the London cluster when grabbed straight from the random forest model.

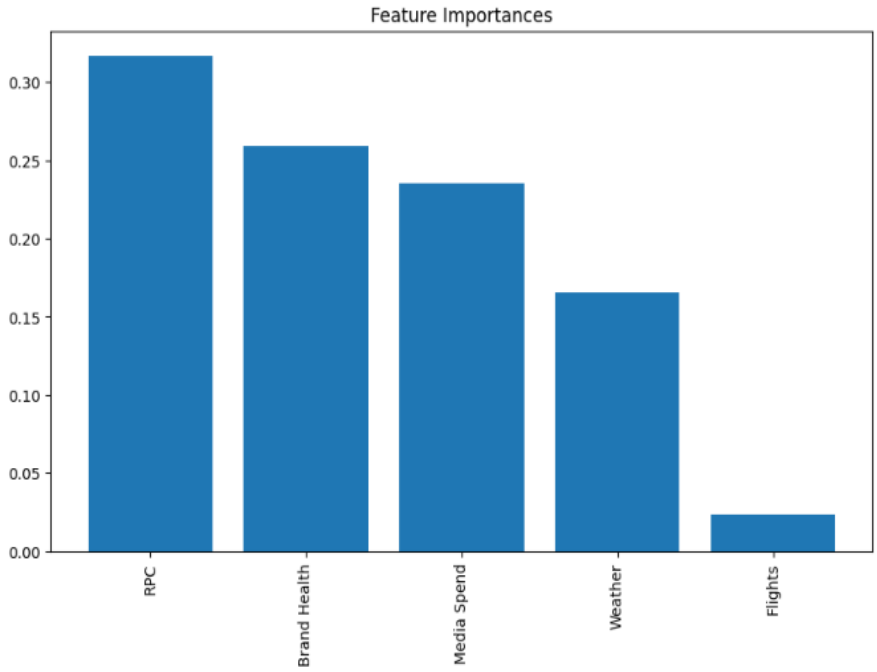



*Figure 12: random forest feature importances*

I also ran a permutation analysis, which will randomly shuffle the feature values around and check how much they affect prediction accuracy, therefore giving an overall score, which yielded similar results to those in figure 12. These methods, however, lack information of how features affect the data. I therefore used a package called SHAP (Shapley Additive Explanations), to determine feature importance by getting a unified measure of importance by calculating the contribution of each feature to the prediction. These SHAP values are derived from cooperative game theory and offer consistent and interpretable insights into model predictions (figure 13).

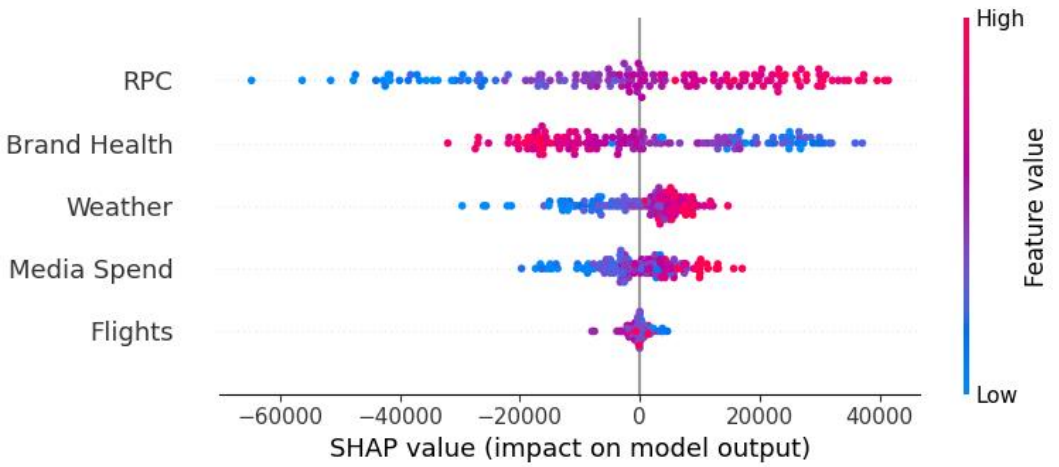



*Figure 13: Shapley Additive Explanations for random forest features*

From the results of our SHAP analysis, I noticed a few unexpected outcomes. The first, is that a higher price will lead to more attendance/revenue, and a lower price leads to lower attendance/revenue. This makes sense, as higher attendance days will usually be priced higher than less popular days (such as weekends vs weekdays). However, a better source for a project like this would be a calendar of price adjustments, to better understand pricing impacts. However, Merlin does not record these, forcing me to drop ticket price as a feature for now. In terms of Brand Health, higher NPS scores are highly correlated to lower attendance and revenue. While I expected a higher score to mean more attendance, this also makes sense. Crowded parks and big queues will have a detrimental effect to guest experience, leading guests to give their experience a lower score. Therefore, with higher attendance we will see lower scores. The reason for leaving weather split, is that the weather variables had very different structures. For example, we expect more rain to lead to less attendance, but higher temperatures to lead to more attendance. To make my SHAP analysis clearer and more explainable, I decided to leave these separate. Media spend's effect on attendance is instead as expected: when more is spent on marketing, parks will have a higher attendance and therefore higher revenue.

## Forecasting Model

Using the above list of features, I can now build a 12-month forecast for each of my parks. Figure 14 shows how I set up the training and running of my model.

The model runs slightly differently for each park. For each of these, I check how I have configured my model using model_config. Based on this, the function will run through various if statements to add certain features, such as seasonality. These are added to the prophet model as regressors. I then fit my training data to the model. Once this is done, I run my model using the run_model function. I then remove unwanted training data from my predictions dataframe and return it to my pipeline. For LEGOLAND Florida, I can only use seasonality, media spend and inbound flights, as they are the only forward looking values I have for the POC. If the accuracy is any good, I can then start adding more features in as the POC is moved to a fully fledged data product. I therefore ran a 12 month forecast with these features using prophet. Figure 15 shows the result of this forecast.

```python
def train_model(train_df, importance, model_config):

    seasonality = model_config.seasonality
    add_periods = model_config.add_periods
    add_flights = model_config.add_flights
    add_media_spend = model_config.add_media_spend

    if seasonality:
        model = Prophet(weekly_seasonality=True, yearly_seasonality=True)
        model.add_seasonality(name='monthly', period=30.5, fourier_order=5)
    else:
        model = Prophet()

    if add_flights:
        model.add_regressor("Flights")
    if add_media_spend:
        model.add_regressor("Media Spend")

    model.fit(train_df)

    return model

def run_model(future_df, model):

    future = future_df.drop(columns=["y"])
    forecast = model.predict(future)

    return forecast
```

*Figure 14: time-series forecasting model training using subset (for backtesting) or all weekly data (for forward looking)*


*Figure 15: 12-month visitation forecast for LEGOLAND Florida*

The forecast shows the expected peak season around august and the spring period, while showing a sharp dip in the winter period following Christmas. When compared to previous years, this looks realistic (figure 16). However, I will also perform a backtest to ensure that the model is trustworthy and accurate enough to make operational decisions.

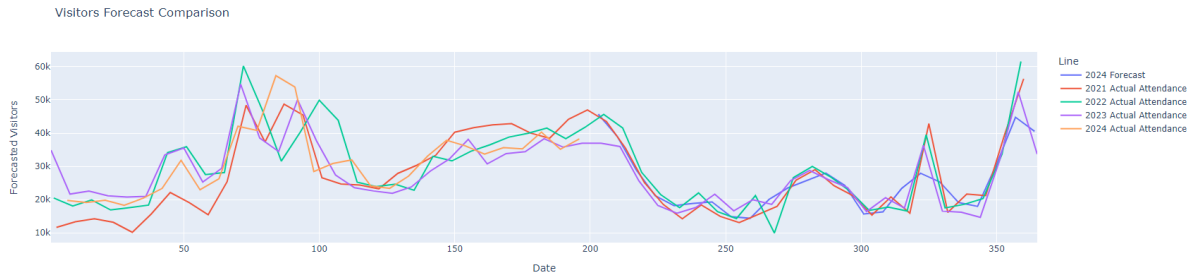



*Figure 16: year-to-year comparison of attendance at LEGOLAND Florida*

I therefore ran a 6 month backtest and the results are promising (figure 17). With an average weekly attendance in Florida of 31700, the average weekly error is 3628, with an MAPE of 11.7%. For my revenue model, the MAPE is 16%. These are promising results when taking into account that I have only used seasonality and media spend, therefore omitting many other data sources, not currently available, that will eventually be very useful in improving accuracy (such as school holidays and hotel occupancy).

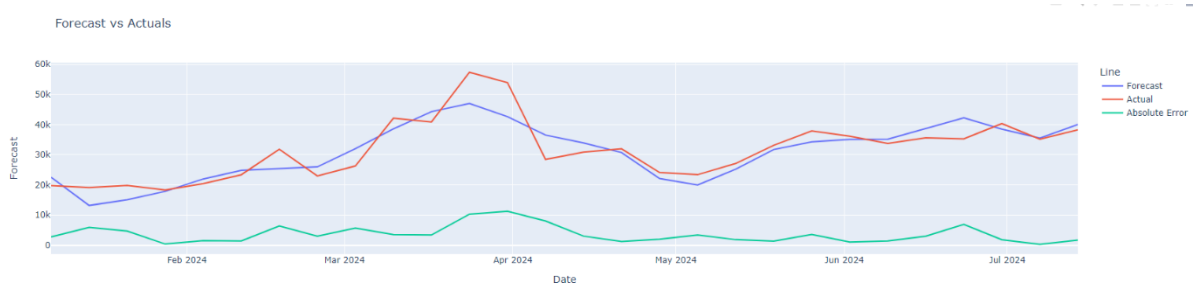
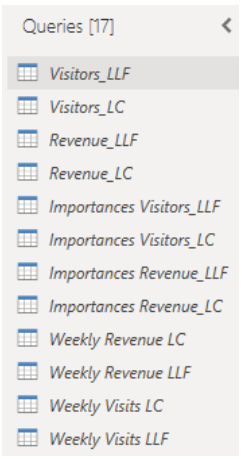



*Figure 17: 6 month backtest for LEGOLAND Florida time-series model*

## Dashboard

Now that I have results for both feature importance and a forecast for the next 12 months, I can start building a dashboard in power bi. All my data is stored in a Fabric Lakehouse, making it easy for me to extract. I start with one table per model and per park, meaning for the POC I have four tables total for the forecasts and four more for the explanatory model, as shown in figure 18a.

A




B

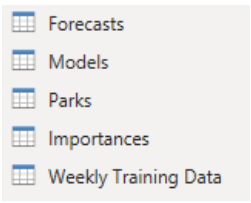



*Figure 18: A) pre-transformed model outputs in power query, pulled from Fabric Onelake. B) tables post transformation in power query*

To make these easier to work with, I appended the four forecast tables together and the four importances tables, giving me two big tables to work with. To differentiate from each park and type of model, I have a "Model ID" and "Park ID" column that map to lookup tables "Models" and "Parks" (figure 18b). I then exited power query and started working on the visualizations for the dashboard. I first built a dates table using DAX, to normalize dates through my weekly training and forecast tables. My data model looks good straight off the bat (figure 19), meaning I can move straight into the visualizations.

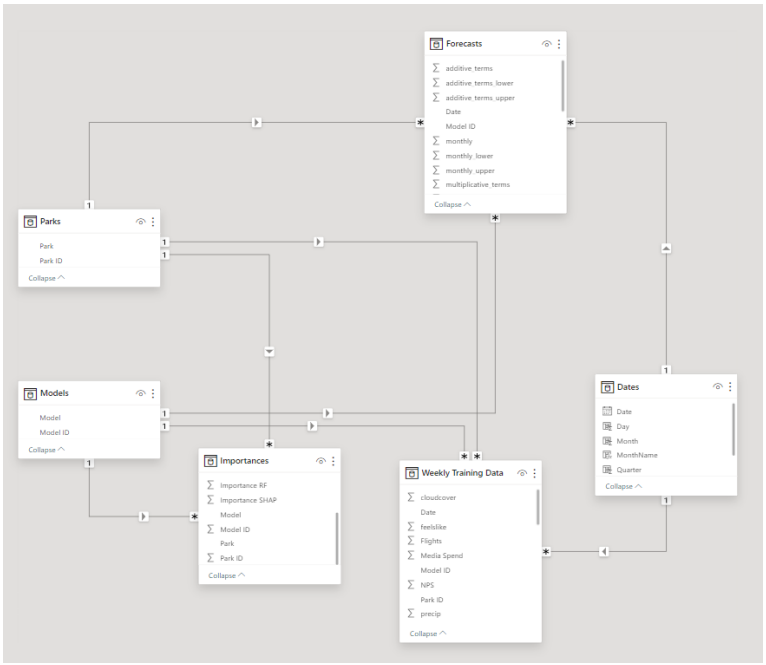



*Figure 19: power bi dashboard data model*

For my visuals, I used a colourblind safe theme, as there are a few colourblind employees working on revenue management and operations. I decided to give an overall ranking of all the features I used within the model by the type of model I used (permutation importance, SHAP importance and random forest importance) using pie charts. I then grouped some of the features (Seasonality & Weather) and averaged out all the three importance score methods for an overall ranking, which I show using a funnel visual. For the forecast, I used a simple line chart which gives a good idea of how attendance is changing throughout the year (based on our predictions). I also placed the budgeted values as a comparison. My forecast will go further in the future than the budget, which is why the budget line will stop in December 2024. These visuals together create my overview page for my dashboard (figure 20).

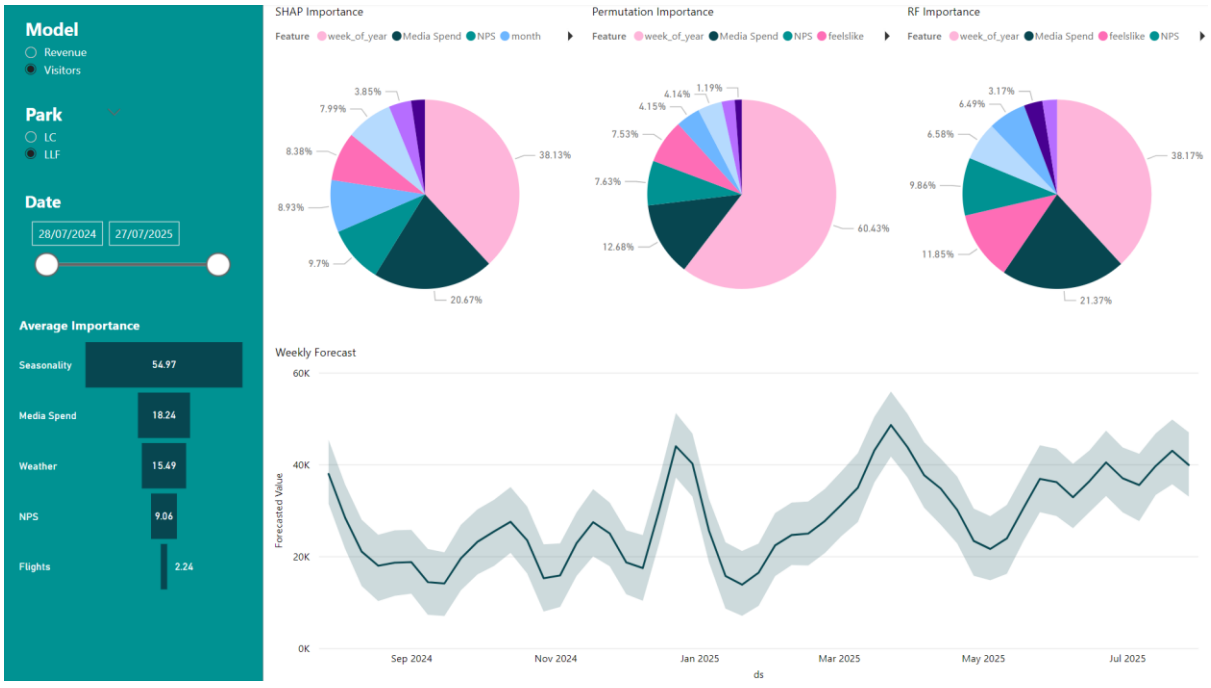



*Figure 20: power bi dashboard page 1 (Overview)*

I also created a separate page, called In-Depth View, which can be used to get specific numbers for the upcoming forecast, with the possibility of grouping by month or quarter, with a comparison to last year's numbers (figure 21). To get previous year numbers, I also connected to the weekly training dataframes I used for my models, which also sit in fabric. I treated these the same way to create a single weekly table with both parks and both model types.

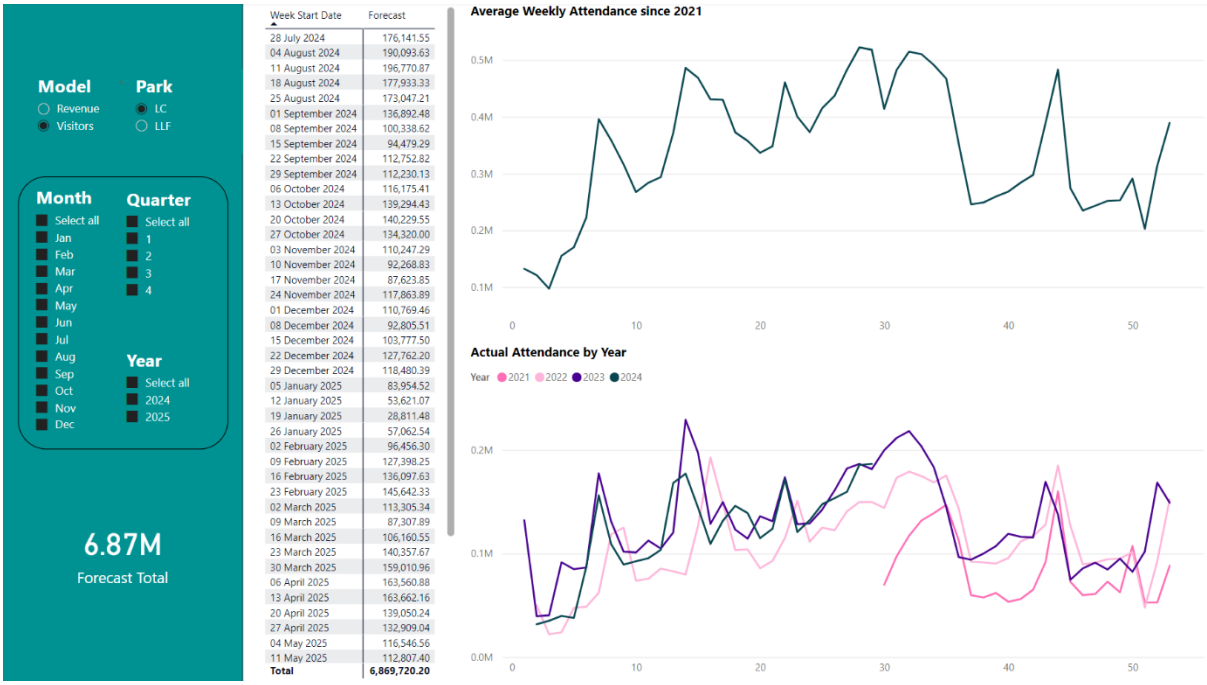



*Figure 21: power bi dashboard page 2 (In-Depth View)*

## Recommendations & Conclusions

The dashboard I generated in this project will give a quick and easy insight for revenue analysts to make operational decisions for the coming months. It will significantly speed up their day-to-day work and increase efficiency within the business. The explainability model I built will help the business make operational decisions, such as increasing media spend or putting in offers to increase attendance on rainy days. While the features are limited, the POC will soon expand and include a variety of other controllable features such as ticket discounts or park events, increasing the usefulness of the tool.

The current accuracy for yearly budget at LEGOLAND Florida is 15.2%, both for visitors and for revenue. My current forecasts are 11.7% for visitors and 16.2% for revenue. While my visitors forecast is hugely outperforming the yearly budget accuracy, my revenue forecast needs some work. However, as a starting point, this is a great result. The budget on file is the accuracy taken throughout the year, where managers will tweak budget as the year progresses, which means a true comparison of accuracy can only be calculated later in the year. I have recommended to the business that they still consider any unexpected events, such as a new ride opening, into their operational decisions, rather than relying purely on the product. There can always be unexpected events such as a ride opening we were not briefed on, meaning operational decisions cannot be automated just yet.

There are some immediate next steps, which include presenting the results to the senior stakeholders and, if the project gets approved, it will move from POC to being a fully fledged data product. This will require lots of work in terms of data engineering to make sure the data pipelines are robust and can be moved to production. There is also a lot of potential for more data sources to be included in the model. The readily available sources were limited, and some new ones include google trends data, other brand health metrics such as google reviews, and more granular media spend and flights data. Once these are added, we can start expanding to more parks and make this product a company-wide tool.

## Knowledge, Skills & Behaviour

K3 principles of the data analysis life cycle and the steps involved in carrying out routine data analysis tasks (Project Plan, Page 2-3)

K4 principles of data, including open and public data, administrative data, and research data (Data Transformation, Pages 5-8)

K8 quality risks inherent in data and how to mitigate/resolve these (SHAP Analysis of Explanatory Model, Page 9, risks of keeping revenue per customer as a data source)

K9 principal approaches to defining customer requirements for data analysis (Scope & Project Outcomes, Page 2)

K11 approaches to organisational tools and methods for data analysis (Legislation, Organisational Policies & Procedures, Page 3)

K12 organisational data architecture (Legislation, Organisational Policies & Procedures, Page 3)

S1 use data systems securely to meet requirements and in line with organisational procedures and legislation, including principles of Privacy by Design (Legislation, Organisational Policies & Procedures, Page 3)

S2 implement the stages of the data analysis lifecycle (Throughout project, Set up environment, pulled and transformed data, researched different model solutions, built model, displayed results and recommended actions in a dashboard)

S3 apply principles of data classification within data analysis activity, flexing approach as necessary (Legislation, Organisational Policies & Procedures, Page 3)

S4 analyse data sets taking account of different data structures and database designs (Data Transformation, Pages 5-8)

S6 identify and escalate quality risks in data analysis with suggested mitigation/resolutions as appropriate (Explanatory Model, RPC Data discussion, NPS Data discussion)

S7 undertake customer requirements analysis and implement findings in data analytics planning and outputs (Dashboard, Pages 12-14)

S8 identify data sources and the risks, challenges to combination within data analysis activity (Data Transformation, Pages 5-8)

S12 collaborate and communicate with a range of internal and external stakeholders using appropriate styles and behaviours to suit the audience (Team Structure, Page 3-4)

S15 select and apply the most appropriate data tools to achieve the best outcome (Legislation, Organisational Policies & Procedures, Page 3)

B3 Works independently and collaboratively (Team Structure, Page 3-4, worked independently in technical terms for POC but will require help past approval of project)

B4 Logical and analytical (Throughout project, took independent decisions on which data sources work and don’t introduce any target leakage or confusion)