Building Transparent Campaign Intelligence: A Machine Learning Dashboard Using Synthetic Advertising Data

CM3203 – BSc Project

BSc Computer Science

School of Computer Science and Informatics, Cardiff University

Jack Thurman

C21128004

Project Supervisor: Dr Wei Zhou

Date: 12/05/2025

Abstract

Digital advertising platforms increasingly rely on proprietary data and infrastructure to offer advanced campaign prediction and optimisation features, making such tools out of reach for smaller advertisers and independent researchers. This project explores whether similar capabilities can be achieved using only synthetic data and open-source technologies. A fully interactive Streamlit dashboard was developed to simulate a campaign intelligence platform, allowing users to visualise KPIs, generate predictions, and receive optimisation recommendations. The system leverages a synthetic dataset modelled on real advertising trends. It includes several machine learning models, such as linear regression, random forest, and XGBoost, for predicting success scores and click-through rates (CTR). Additionally, a TensorFlow-based neural network combined with SHAP explainability was used to provide interpretable post-run optimisation suggestions. While real-world benchmarking was beyond the scope due to the use of synthetic data, the project successfully demonstrates the technical feasibility and potential value of an open, transparent alternative to commercial campaign analytics platforms.

Acknowledgements

I would like to thank my supervisor, Wei Zhou, for guiding me through this project. I would also like to thank my father, Lea Thurman.

Table of Contents

1. Introduction 5
   1.1. Project Goal 5
   1.2. Motivation 5
   1.3. Project Objectives 6
   1.4. Target Audience 6
   1.5. Assumptions and constraints 6
   1.6. Ethical Considerations 7
   1.7. Structure of report 7
2. Background 8
   2.1. Wider context of the project 8
   2.2. Identified Problem 8
   2.3. Stakeholders and implications 9
   2.4. Relevant Theories and Concepts 10
   2.5. Constraints on the Approach 10
   2.6. Existing Solutions and Their Limitations 11
   2.7. Summary of the Background 11
3. Methodology 12
   3.1. Overview of System Architecture 12
      3.1.1. System Components 12
      3.1.2. Architecture Flow 13
   3.2. Dashboard Visualisations 14
   3.3. Dataset design and simulation 19
   3.4. Machine Learning Models 27
      3.4.1. Model Selection 27
      3.4.2. Model Implementation 28
      3.5. Definition of "Success Score" 31
   3.6. Tools and technologies 31
4. Results and Evaluation 32
   4.1. Evaluation Strategy 32
      4.1.1. Predictive Accuracy 32
      4.1.2. Generalisation Assessment 32
      4.1.3. Cross-validation for Robustness 33
      4.1.4. Residual Analysis 33
      4.1.5. Interpretability: SHAP Analysis 33
      4.1.6. Learning Dynamics for Neural Network 33

4.2. Quantitative Model Results 33
4.2.1. Performance Summary 33
4.2.2. Predicted vs Actual Fit 35
4.2.3. Residual Distribution 36
4.2.4. Parameter Tuning and Optimal Configuration 37
4.2.5. Neural Network Results 39
4.3. Qualitative Evaluation 40
4.3.1. Model Interpretability with SHAP 40
4.3.2. System Transparency and Trust 42
4.3.3. Dashboard Usability (Heuristic Review) 42
5. Conclusions and future work 43
5.1. Summary of project aims and achievements 43
5.2. Key findings and insights 44
5.3. Limitations for the current approach 44
5.4. Recommendations for next steps 45
5.5. Conclusion 45
6. Reflection on Learning 45
7. Appendices 47
8. References 47

1. Introduction

1.1. Project Goal

The goal of this project is to investigate whether it is feasible to replicate key functionalities of commercial advertising intelligence platforms, specifically prediction, visualisation, and optimisation of campaign performance, using only publicly available tools and synthetic data. The project aims to challenge the prevailing assumption that high-performance campaign intelligence requires access to proprietary datasets or advanced infrastructure. By designing a fully functional prototype system, including a predictive engine, a user-facing dashboard, and a data simulation framework, this project seeks to demonstrate the potential for building free, transparent, and accessible alternatives. The broader objective is to democratise access to campaign analytics and empower smaller advertisers, developers, and researchers who are often excluded from commercial solutions due to cost and data limitations.

1.2. Motivation

Modern digital advertising platforms rely heavily on opaque, proprietary infrastructure and data, which restricts access for smaller players and limits transparency. These systems often operate

as "black boxes," providing outcomes without exposing the logic behind predictions or recommendations. In light of increasing calls for algorithmic accountability and data ethics, this project explores the potential for open, interpretable tools that can support campaign intelligence without relying on private data or exclusive ecosystems. A more detailed discussion of this context and its implications is provided in Section 2 (Background).

1.3. Project Objectives

This project aims to meet the following objectives:

- Understand real-world advertising dynamics to inform dataset simulation, using common KPIs such as click-through rate (CTR), cost-per-impression (CPI), and user engagement.
- Create a synthetic dataset modelled on industry benchmarks to allow controlled yet realistic experimentation.
- Develop a Streamlit dashboard as the system's front-end, enabling interactive visualisation, prediction, and optimisation.
- Train predictive models (e.g., linear regression, random forest, XGBoost) to estimate success metrics from campaign configurations.
- Implement a deep learning-based optimiser using TensorFlow and SHAP to generate performance-improving recommendations.
- Evaluate the system's practicality and accuracy in replicating core functions of proprietary platforms within the constraints of synthetic data and open tools.

Each objective is explored in more technical depth throughout Sections 3 (Methodology) and 4 (Evaluation).

1.4. Target Audience

This project is designed for a broad range of users who may lack access to enterprise-level analytics tools but still require actionable advertising insights:

- Independent Advertisers and SMES: Those seeking affordable, user-friendly alternatives to platform-native tools.
- Researchers and Educators: Individuals focused on studying digital marketing trends or teaching data-driven decision-making.
- Data Science and ML Practitioners: Professionals interested in applying machine learning to campaign data using open, reproducible methods.

By promoting accessibility and transparency, the system aims to support independent experimentation, informed decision-making, and open research in digital advertising.

1.5. Assumptions and constraints

To guide development, the project operates under several key assumptions:

- Synthetic data can approximate the patterns of real campaign data sufficiently for modelling and evaluation.
- Users have basic familiarity with advertising KPIs.
- Open-source tools (e.g., Streamlit, scikit-learn, TensorFlow) are capable of supporting the intended system functionality.

Constraints include:

- No use of proprietary or personal data, due to ethical and regulatory considerations (see Sections 1.6 and 2.5).
- Limited computational resources, restricting model complexity and data volume.
- Academic timeline, requiring prioritisation of core features over full product refinement.

These assumptions and constraints are discussed further in the context of the system's design in Section 3 (Methodology).

1.6. Ethical Considerations

The project was designed with a privacy-first approach. No personal or sensitive data was collected or processed. Instead, all training and testing were conducted using synthetic data informed by public sources, including demographic patterns from the UK Office for National Statistics (ONS). The ONS data is freely available under the Open Government Licence, ensuring legal and ethical compliance. By avoiding real user data and using only open-source technologies, the project supports responsible AI development and data transparency. Broader ethical implications and industry concerns are further examined in Section 2.5 (Constraints) and Section 2.6 (Existing Solutions and Limitations).

1.7. Structure of report

This report is structured to provide a clear and logical progression from the project's motivation and objectives through to implementation, evaluation, and reflection. It consists of the following key sections:

- Section 2 – Background: This section contextualises the project within the broader digital advertising landscape, identifying existing problems with transparency, exploring relevant theories, and reviewing current solutions and their limitations.
- Section 3 – Methodology: Describes the technical implementation of the system, including the design of the synthetic dataset, system architecture, dashboard functionality, and machine learning models used for prediction and optimisation.
- Section 4 – Results and Evaluation: Presents a detailed analysis of the model performance using both quantitative and qualitative methods. This includes predictive accuracy, cross-validation, SHAP-based interpretability, and a usability review of the dashboard interface.

- Section 5 – Conclusions and Future Work: Summarises the main achievements of the project, discusses the limitations encountered, and proposes directions for future development and research.
- Section 6 – Reflection on Learning: Provides a personal reflection on the learning outcomes, challenges overcome, and skills developed throughout the project.
- Sections 7 and 8 – Appendices and References: Contain supplementary materials, including implementation instructions and source references, to support the reader's understanding and enable reproducibility.

This structure ensures that both technical and non-technical readers can follow the development, execution, and implications of the project.

2. Background

2.1. Wider context of the project

Over the past decade, digital advertising has rapidly become the dominant force in global ad spend. In 2024, Google and Meta accounted for over £26 billion of the UK's £43 billion advertising market. Their platforms have become essential to campaign deployment, optimisation, and analytics. [1]

However, these platforms operate with increasing opacity. Advanced tools for forecasting and optimisation are typically locked behind paywalls or accessible only to enterprise-level advertisers. Independent advertisers, researchers, and regulators are often excluded from meaningful engagement with these systems. The result is an ecosystem in which outcomes are visible, but the logic behind them remains hidden.

Critics like Shoshana Zuboff describe this shift as a symptom of "surveillance capitalism," where user behaviour becomes commodified without transparency. [2] The ACM further stresses the need for algorithmic accountability and redress mechanisms in digital systems. [3]

These dynamics fuel a growing demand for open, reproducible tools that allow users outside these commercial ecosystems to conduct independent analysis, verify results, and explore campaign strategies — goals that this project directly responds to.

2.2. Identified Problem

Despite offering robust features, major platforms like Google and Meta often fall short in transparency. The inner workings of their campaign analytics and optimisation tools are largely inaccessible to end-users. As the Mozilla Foundation notes, many ad oversight tools fail to deliver meaningful transparency, even when explicitly designed for public accountability. [4]

This creates two interconnected issues: a lack of insight into how campaigns are evaluated or optimised, and increasing pressure on advertisers to comply with data protection laws like the GDPR. Article 5 of the GDPR requires that personal data be processed transparently, lawfully, and fairly — conditions that many proprietary platforms do not meet. [5]

Together, these factors create a problematic environment where users cannot fully understand or trust the systems they rely on for campaign decision-making.

2.3. Stakeholders and implications

The challenges in digital advertising transparency and accessibility impact a diverse array of stakeholders. Understanding their unique needs and perspectives is crucial for developing inclusive and effective solutions.

SMES often lack the resources and expertise to navigate complex digital advertising platforms. According to a study published in Sustainability, "Digital marketing helps SMES gain financial returns, market visibility, and customer satisfaction." However, the study also notes that many SMES face difficulties in adapting to digital transformations due to limited resources and knowledge gaps. [6]

Access to transparent and comprehensive advertising data is essential for researchers studying the impacts of digital marketing. A paper on the transparency of digital native and embedded advertising emphasises, the lack of transparency in digital advertising poses significant challenges for regulation and education. [7]

Ensuring fair competition and protecting consumer rights in digital advertising are primary concerns for regulators. The European Union's Digital Markets Act (DMA) aims to address these issues by imposing obligations on large online platforms. The act seeks to "ensure fair competition and protect the consumer in the digital economy." [8]

Open-source initiatives play a pivotal role in democratising access to digital advertising tools. The IAB Tech Lab's Open-Source Initiative encourages collaboration to build solutions for common industry challenges, aiming to increase cross-industry transparency and collaboration. [9]

Figure 1 - Stakeholders Diagram (generated by ChatGPT)

2.4. Relevant Theories and Concepts

To address the challenges identified in the digital advertising landscape, this project leverages several theories and concepts:

As noted by Impression Digital, "Synthetic data refers to artificially generated data that mimics the characteristics of real-world data without containing any actual information from real individuals or events." [10] This approach is particularly valuable in scenarios where real data is scarce, sensitive, or subject to privacy regulations

Predictive modelling involves using statistical techniques and machine learning algorithms to analyse historical data and predict future outcomes. In the context of advertising, predictive modelling can forecast metrics like click-through rates (CTR) and conversion rates, enabling advertisers to optimise their campaigns. As highlighted by SAS, "Predictive analytics is driven by predictive modelling. It's more of an approach than a process. Predictive analytics and machine learning go hand-in-hand, as predictive models typically include a machine learning algorithm." [11]

SHAP (SHapley Additive exPlanations) is a method used in machine learning to explain the output of models. As described in the SHAP documentation, "SHAP values are a method used in machine learning for explaining the output of a model by attributing each feature's contribution to the final prediction." [12]

Interactive visual analytics combines data visualisation with analytical reasoning to help users explore and understand complex datasets. This approach facilitates better decision-making by allowing users to interact with data through dashboards and other visual tools. Qlik defines visual analytics as "the use of sophisticated tools and processes to analyse datasets using visual representations of the data. Visualising the data in graphs, charts, and maps helps users identify patterns and thereby develop actionable insights." [13]

2.5. Constraints on the Approach

The development of an open, transparent advertising analytics dashboard is influenced by several key constraints, which shape the project's scope, methodology, and implementation.

Access to real-world advertising data is often restricted due to proprietary concerns and stringent privacy regulations. The General Data Protection Regulation (GDPR) emphasises the principle of data minimisation, stating that personal data must be "adequate, relevant and limited to what is necessary in relation to the purposes for which you are processing it". This legal framework limits the use of actual user data, necessitating alternative approaches. [14]

To navigate these limitations, the project employs synthetic data generation. While synthetic data can mimic the statistical properties of real datasets, it may not capture all the nuances of actual user behaviour. As noted by AIMultiple, "Synthetic data may not fully capture the complexity and unpredictability of real-world conditions, potentially leading to performance gaps". [15]

The project relies exclusively on open-source tools to ensure accessibility and reproducibility. While these tools offer flexibility, they may lack certain features found in proprietary software.

For instance, open-source machine learning libraries may not provide the same level of optimisation or user support as commercial alternatives.

Additionally, the project is constrained by limited computational resources, which can affect the complexity of models and the volume of data processed. This limitation necessitates a balance between model sophistication and computational feasibility.

Operating within the confines of an academic timeline, the project prioritises the development of a functional prototype over a fully polished product. This time constraint influences decisions regarding feature implementation, testing, and documentation.

The scope is deliberately narrowed to focus on core functionalities that demonstrate the feasibility and value of an open, transparent advertising analytics tool. Future iterations may expand upon this foundation to include additional features and refinements.

2.6. Existing Solutions and Their Limitations

Several platforms and tools currently exist to provide advertisers with insights into digital advertising performance. However, these solutions often come with limitations that hinder comprehensive transparency and effectiveness.

Major digital advertising platforms like Google and Meta offer in-platform analytics tools. While these tools provide valuable metrics, they often lack the depth and transparency needed for a complete understanding of ad performance. As noted by Rockerbox, "Today, the industry as a whole relies heavily on in-platform reporting metrics provided by giants like Facebook, Google, and TikTok. While these metrics offer valuable insights, they often lack the granularity and transparency needed for a comprehensive understanding of ad performance." [16]

Platforms have introduced ad transparency tools, such as Meta's Ad Library, to provide visibility into advertising practices. However, these tools often offer limited access to data, restricting comprehensive analysis. A study published in the ACM FAccT conference proceedings highlights, "Overall, these tools only provide access to a restricted subset of advertisements, of which we do not know the sample population." [17]

Open-source analytics tools offer flexibility and community-driven development. However, they may lack certain features and support found in commercial solutions. Panoply notes, "While open-source is free to use, knowledge and labour costs are inescapable. Moreover, developing enough expertise around open-source software to use it effectively takes time." [18]

2.7. Summary of the Background

The digital advertising landscape is characterised by a lack of transparency and accessibility, particularly affecting small and medium-sized enterprises (SMES). While major platforms like Google and Meta offer analytics tools, these often lack the granularity and openness required for a comprehensive understanding.

Regulatory frameworks such as the General Data Protection Regulation (GDPR) impose strict guidelines on data usage, emphasising the need for transparency and user consent. This legal

environment necessitates the exploration of alternative data sources and methodologies that comply with privacy standards.

To address these challenges, the project leverages synthetic data generation, acknowledging its limitations in capturing the full complexity of real-world user behaviour.

The solution integrates open-source tools such as Streamlit and Plotly to facilitate customisable and transparent analytics. Streamlit offers a user-friendly interface for creating shareable dashboards, while Plotly provides a platform for generating a variety of data visualisations. [19], [20]

By combining these elements, the project aims to empower SMES with a transparent, compliant, and accessible advertising analytics dashboard. This tool seeks to bridge the gap left by existing solutions, offering a more inclusive approach to digital advertising analytics.

3. Methodology

3.1. Overview of System Architecture

The system developed in this project replicates key functionalities of commercial advertising platforms—namely prediction, visualisation, and optimisation—using a modular, open-source architecture. The design prioritises transparency, simplicity, and accessibility, in line with the project's goal to create a freely available alternative to proprietary tools.

3.1.1. System Components

The architecture consists of three primary components:

Synthetic Data Layer

Simulates realistic campaign data based on industry research and UK demographic distributions (see Section 3.3). This forms the foundation for training, testing, and user interaction.

Machine Learning Layer

A collection of predictive models trained on the synthetic dataset. These include:

- Regression models (linear, ridge, random forest, XGBoost, Gradient Boosting)
- A deep learning model (TensorFlow-based) for optimisation
- SHAP explainability module for post-hoc interpretation

Dashboard Interface (Streamlit)

The front-end interface through which users interact with the system. It includes three main tabs:

- Overview: Displays dataset-wide distributions and trends.
- Filtered View: Allows users to explore campaign-specific performance metrics.

- Optimise: Offers both pre-run predictions and post-run recommendations using the ML models.

Each component operates independently but is integrated to form a cohesive pipeline that mimics the data flow and analysis cycle of real-world advertising platforms.

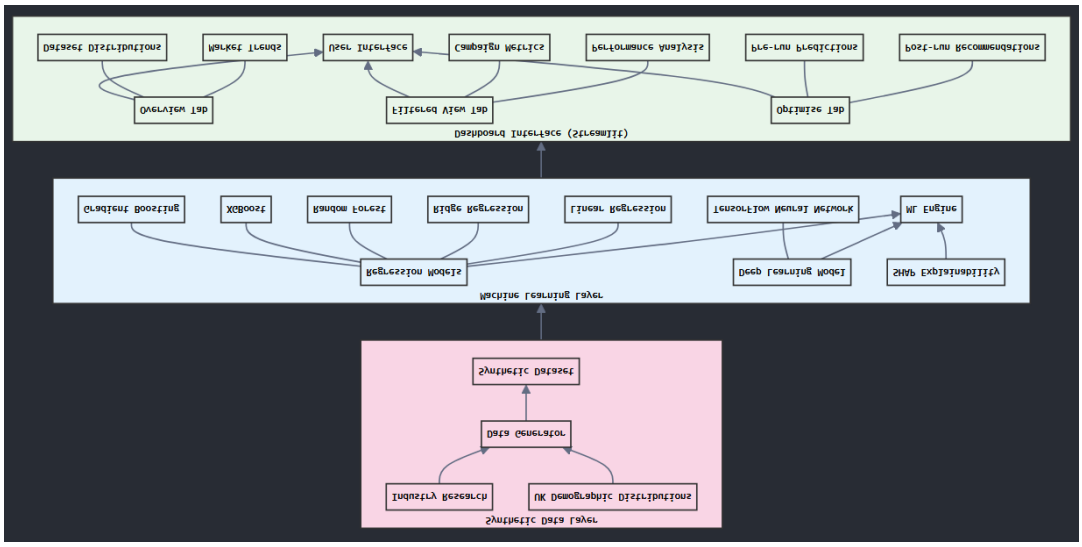

Figure 2 - System Components

3.1.2. Architecture Flow

The system follows a clear and repeatable flow:

- User Input or Selection (via dashboard)
- Data Processing (filtering, encoding, preparation)
- Model Prediction or Optimisation
- Visual Feedback (display of results, graphs, and SHAP interpretations)

This modularity not only supports reusability and scalability but also ensures that each part of the system can be improved or replaced without disrupting the entire workflow.

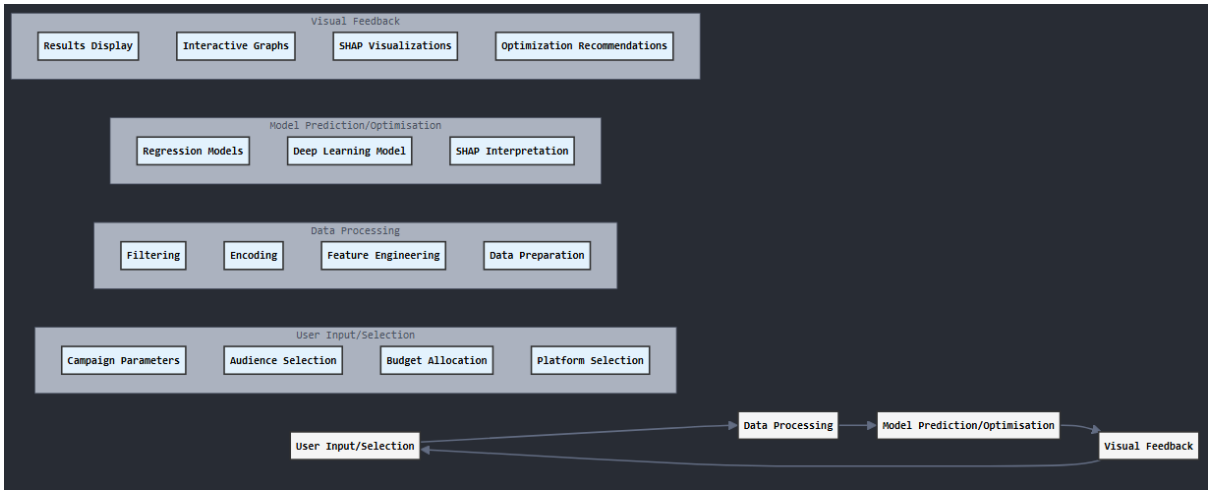

Figure 3 - Architecture Flow (generated by Claude)

3.2. Dashboard Visualisations

The system utilises the Python library Streamlit [19] and Plotly [20] to create a web application that allows advertisers to visualise current or past advertising campaign performance. It allows them to predict how well a future campaign may perform, and it allows them to optimise a currently running campaign. The dashboard contains three tabs: overview, explorer and optimise.

Overview:

This tab allows users to visualise the full synthetic dataset. Figure 4 shows a small section of the page and what kind of insights a user would be able to see.

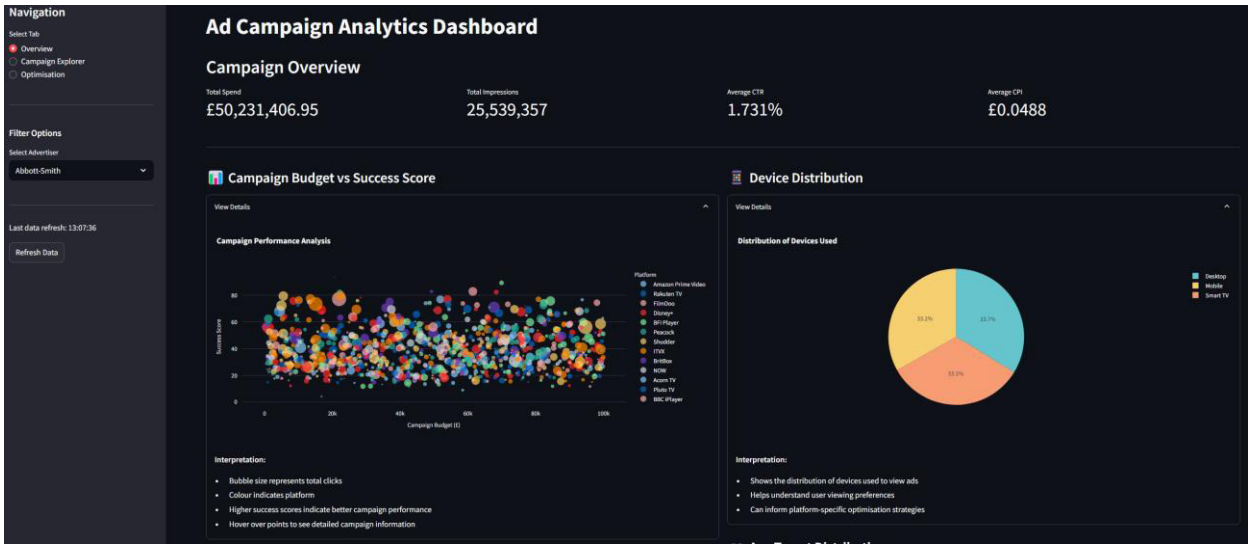

Figure 4 - Overview Page

This page would allow advertisers to see how their campaign performs compared to the full dataset.

The visualisations include:

- Campaign budgets vs success score
- Device distribution
- Engagement by Age and Income Targeting
- Age Target Distribution
- Campaign Duration Distribution
- Ad Spend over Time
- CTR Distribution
- Impressions by platform

Campaign Explorer:

This page allows the advertiser to see how their campaign is performing. The version displayed in Figure 5 allows the user to select an advertiser. This would not be available for customers if this system were released. They would only be able to see their own campaign(‘s).

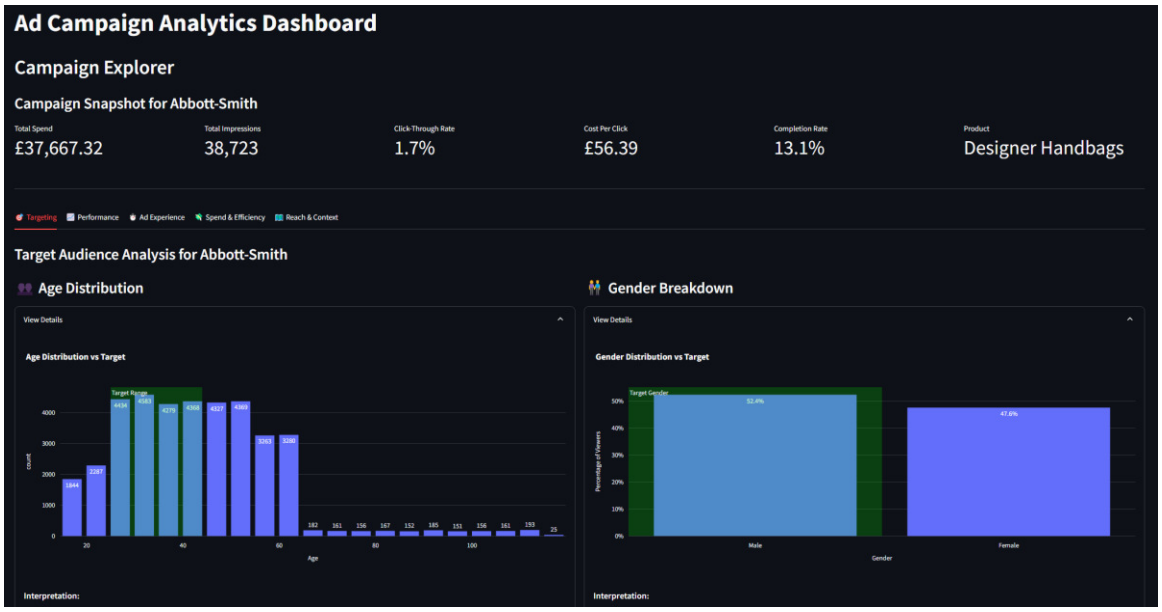

Figure 5 - Campaign Explorer Page

Campaign Explorer starts with basic KPI metrics at the top and then offers the user multiple tabs so that each aspect of their campaign can be thoroughly visualised.

Targeting:

Enables the user to see actual audience vs target audience which is beneficial when understanding why a campaign might be performing a certain way.

Visualisations:

- Age Distribution
- Gender Breakdown
- Income Distribution

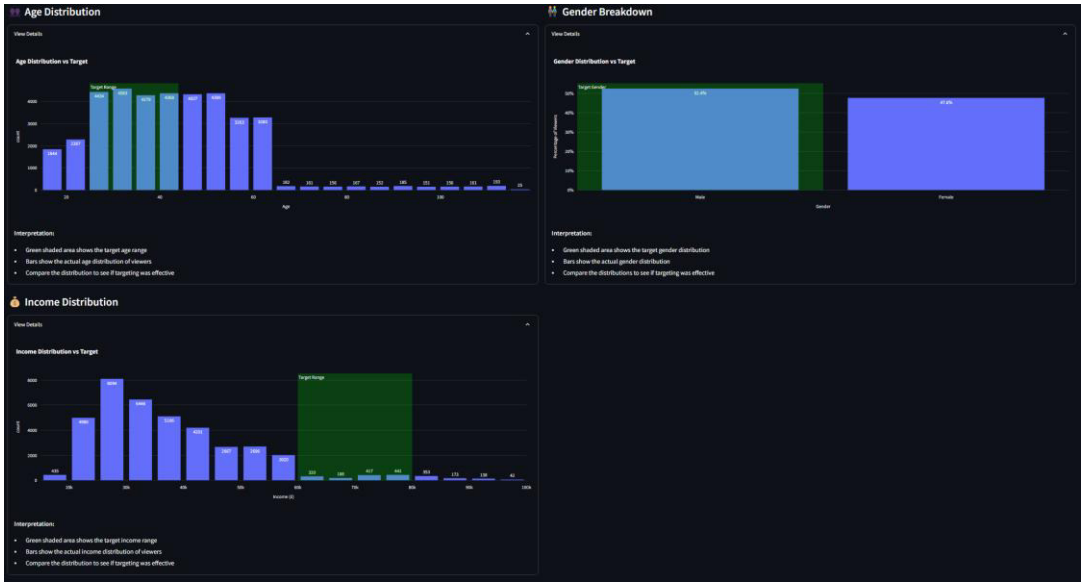

Figure 6 - Targeting Visualisations

Performance:

Displays how the campaign performed in different areas across the duration of the campaign. This would mean users can see how exactly the campaign performs on certain weekdays or at certain times of the month.

Visualisations:

- Daily Impressions
- Daily Engagement Probability
- Daily Clicks

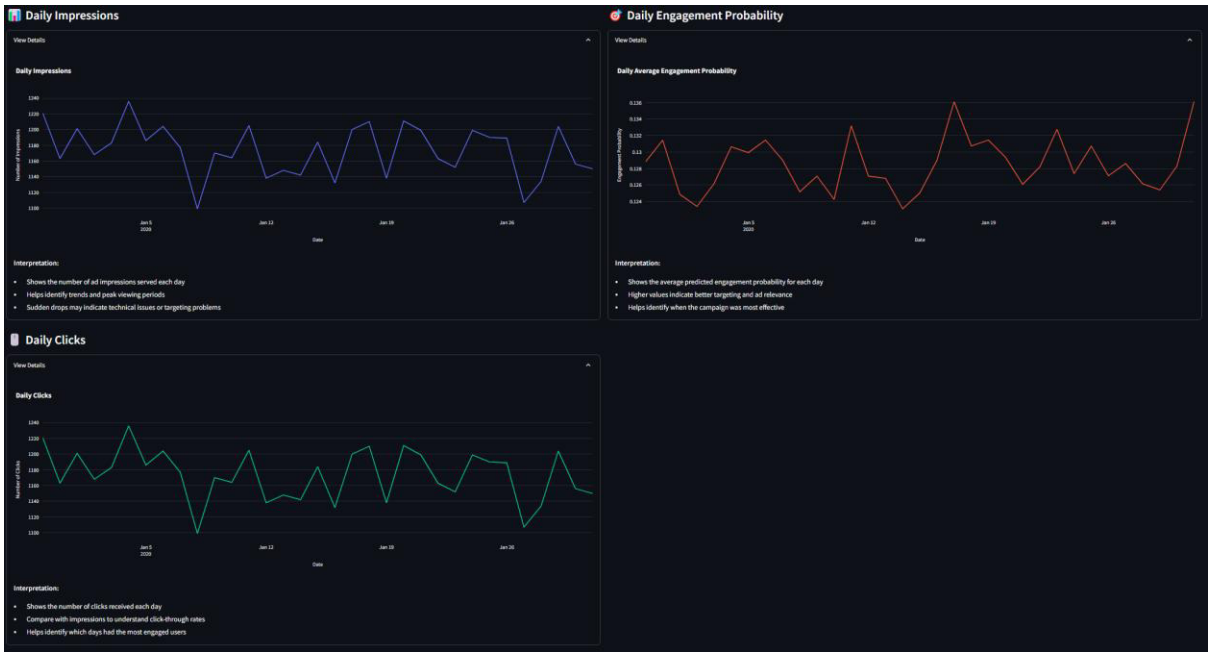

Figure 7 - Performance Visualisations

Ad Experience:

Displays how users are interacting with the ad itself. This could tell advertisers that the advert they are running is not being enjoyed.

Visualisations:

- Completion Rate Gauge
- Percent Watched Progress
- Session Disruption Analysis

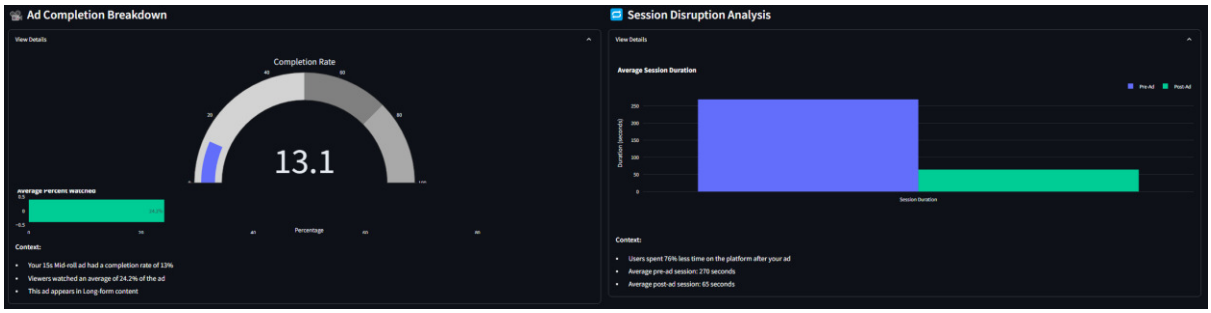

Figure 8 - Ad Experience Visualisations

Spend and Efficiency:

This page provides insight into how well the campaign budget is being spent, and if it is producing the desired results. It also contains insight into how well the campaign aligns with their products optimal advertising months.

Visualisations:

- Spend Breakdown KPIS
- Cost Efficiency Analysis
- Seasonality Analysis

Figure 9 - Spend and Efficiency Visualisations

Reach and context:

This tab provides information on the user's device and location. It also shows how well the platform they are advertising on performs.

Visualisations:

- Content Context
- Device Type Breakdown
- Geographic Reach

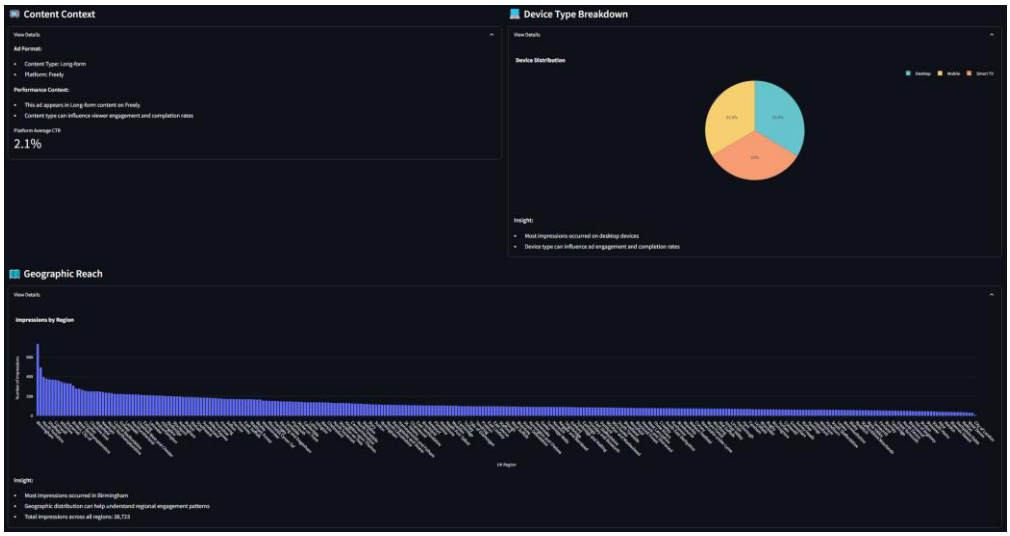

Figure 10 - Reach and Context Visualisations

Optimise:

This page is where users can input their campaign configurations and predict how the campaign might perform. Two options are available: pre-run predictions and post-run analysis. This refers to what kind of data the user would input.

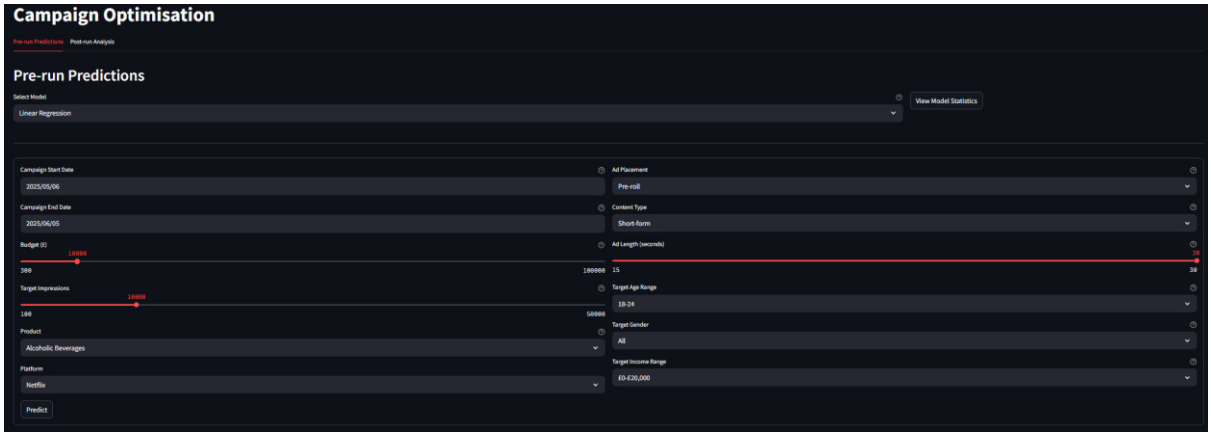

Figure 11 - Pre-run Prediction Menu

Figure 11 shows the pre-run prediction input form. It allows a user to input all campaign data that would be available before it has been run. Users may also select which model to predict with and see how that model performs on the test dataset.

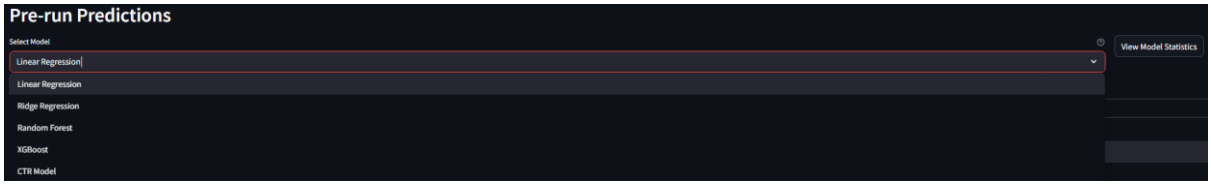

Figure 12 - Model selection

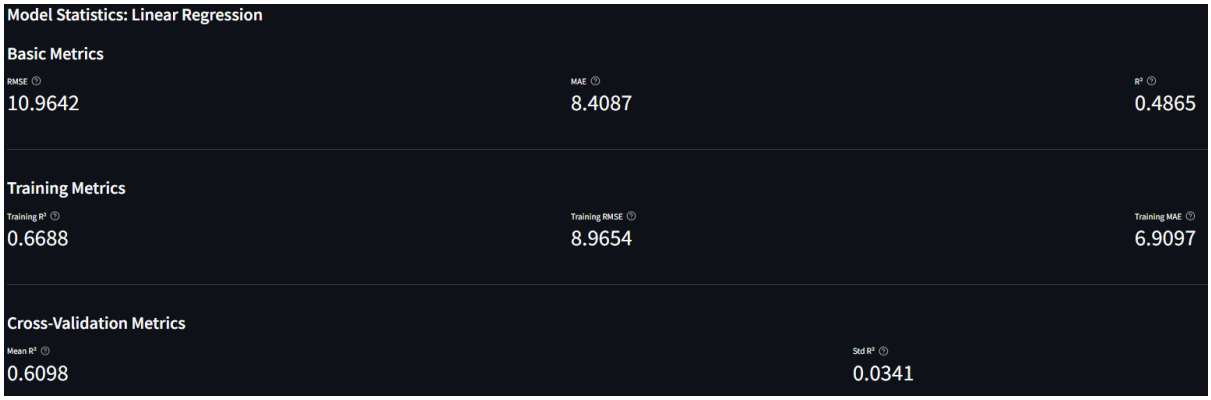

Figure 13 - Model Statistics

Figure 12 shows the models available for a user to choose from. Figure 13 shows the metrics displayed to provide transparency to the user.

3.3. Dataset design and simulation

This section describes the design and simulation of the synthetic dataset used in this project. The dataset was generated to model real-world advertising trends and UK demographics, providing a realistic yet accessible foundation for campaign analysis. Synthetic data was employed to address privacy concerns and the lack of publicly available datasets that accurately represent advertising trends. This approach allows for the evaluation of a campaign analysis tool without needing to pay for huge proprietary datasets.

The database connections are handled by a pymongo.MongoClient [21] singleton pattern, meaning all database trips are done through one instance. The dataset was generated using a series of Python scripts that simulate various aspects of advertising campaigns, user demographics, and interactions. The generation process leverages statistical distributions and real-world data sources to ensure realism.

Users' collection:

User data was generated to reflect UK population demographics, using distributions from UK census data. Attributes such as occupation, age, gender, ethnicity, location, and income were modelled using real-world distributions.

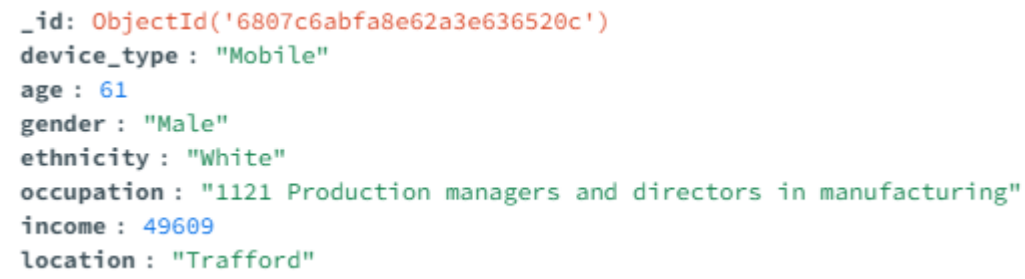

Figure 14 - A User Record

Campaign collection:

Campaigns were assigned random timeframes between 1 to 8 weeks, simulating typical advertising durations. Each campaign included random targeting parameters such as gender and income, as well as a random platform, ad length, ad placement and content type. The product is randomly chosen from an AI-generated set of product rules. This also determines the target age range.

The product rules constant is a list of products, optimal age target and optimal advertising months written by ChatGPT. These rules are then used to determine how well aligned a campaign is with a product's optimal months (seasonality score). In a polished system, these rules would be obtained from market research. However, this is out of scope for this project, so it was deemed unnecessary.

Prompt: 'Write me a list of UK products which includes what age range the product should be advertised to, and what months are optimal for the product to be advertised in.'

Budgets and impressions were randomly assigned within a specified range [22], which was used to derive a realistic cost-per-impression (CPI) metric. Advertisers' names were generated using the Python Faker library [23].

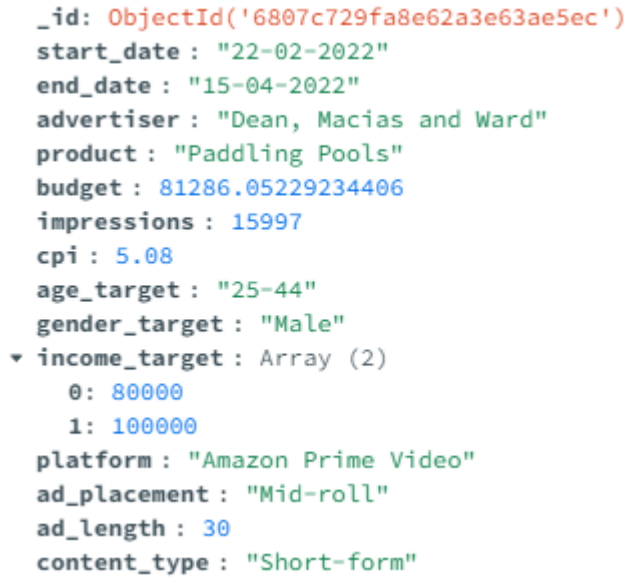

Figure 15 - A Campaign Record

Impressions collection:

Generated for each campaign, assigning a user, an engagement probability and a user affinity. These two scores are what influence how a user interacts with an impression. Engagement probabilities were calculated using a combination of factors such as ad placement, ad length, platform, time of day and content type. Each of these factors has an associated completion rate [24].


```
_id: ObjectId('6807c7bcfa8e62a3e63ae9d4')
campaign_id : ObjectId('6807c729fa8e62a3e63ae5f0')
user_id : ObjectId('6807c6abfa8e62a3e636541a')
dateTime : "01-07-2023 11:25:40"
engagement_probability : 0.13467824046119173
user_affinity : 0.1353909780043121
```


Figure 16 - An Impression Record

```python
def calculate_engagement_probability(impression, user, campaign):
    """
    Calculate the probability of user engagement with an ad impression.

    Args:
        impression: Impression data including timing and context
        user: User data containing demographic and device information
        campaign: Campaign data containing ad specifications

    Returns:
        float: Engagement probability between 0 and 1
    """
    placement_rate = sample_from_beta(PLACEMENT_COMPLETION[campaign['ad_placement']], confidence=5)
    length_rate = sample_from_beta(AD_LENGTH_COMPLETION[campaign['ad_length']], confidence=5)
    content_rate = sample_from_beta(CONTENT_COMPLETION[campaign['content_type']], confidence=5)

    platform_multiplier = sample_from_beta(PLATFORM_ENGAGEMENT[campaign['platform']], confidence=5)

    hour = int(impression['dateTime'].split(' ')[1].split(':')[0])
    if 6 <= hour < 9: time_slot = '06:00-09:00'
    elif 9 <= hour < 17: time_slot = '09:00-17:00'
    elif 17 <= hour < 23: time_slot = '17:00-23:00'
    else: time_slot = '23:00-06:00'

    time_multiplier = sample_from_beta(TIME_OF_DAY_ENGAGEMENT[time_slot], confidence=5)

    device_multiplier = sample_from_beta(DEVICE_TYPE_IMPACT[user['device_type']], confidence=5)

    user_affinity = calculate_product_user_affinity(user, campaign)

    base_probability = (placement_rate + length_rate + content_rate) / 3
    final_probability = base_probability * platform_multiplier * time_multiplier * device_multiplier * user_affinity

    return min(final_probability, 1.0)
```

Figure 17 - Engagement Probability Calculator

Figure 17 shows how the multipliers are combined. A beta distribution is used to add variation and uncertainty to the definite completion rates defined in Krishna's research. [25] This way, the data becomes more realistic.

```python
def sample_from_beta(mean, confidence=10):
    """
    Generate a random value from a beta distribution to add uncertainty to completion rates.

    Args:
        mean: The mean value of the distribution
        confidence: The confidence level of the distribution (higher = more certain)

    Returns:
        float: A random value from the beta distribution
    """
    mean = max(0.0001, min(0.9999, mean))

    alpha = mean * confidence
    beta = (1 - mean) * confidence

    alpha = max(0.1, alpha)
    beta = max(0.1, beta)

    return np.random.beta(alpha, beta)
```

Figure 18 - Beta Sampler

A user affinity score is calculated for each impression. This represents how well a user is suited to the targeting requirements of the campaign the impression is a part of.

```python
def calculate_product_user_affinity(user, campaign):
    """
    Calculate how well a user matches the campaign's target audience.

    Args:
        user: User data containing demographic information
        campaign: Campaign data containing targeting parameters

    Returns:
        float: Affinity score between 0 and 1
    """
    affinity = 0.1

    user_age = user['age']
    age_min, age_max = map(int, campaign['age_target'].split('-'))
    if age_min <= user_age <= age_max:
        affinity += 0.4

    if campaign['gender_target'] == 'All' or campaign['gender_target'] == user['gender']:
        affinity += 0.3

    income_min, income_max = campaign['income_target']
    if income_min <= user['income'] <= income_max:
        affinity += 0.3

    affinity = min(affinity, 1.0)

    confidence = 5 + (affinity * 15)
    affinity = sample_from_beta(affinity, confidence)

    return affinity
```

Figure 19 - User Affinity Calculator

Figure 19 shows the simple additive calculation that determines if each of the targeting requirements have been met, and then simply adds to the final score if they have. The beta sampler is used again to add uncertainty.

Interactions and Interaction Types Collections:

Simulated user interactions with ads, including metrics like watch time, clicks, and session durations. These were influenced by user affinity and engagement probabilities. Data was

modelled to improve flexibility and scalability. The amount and type of interaction were uncertain towards the start of the project. Modelling the database like this allowed for easy additions or deletions from interaction types.

```
_id: ObjectId('6807c9900f7c20b40b64aabf')
impression_id : ObjectId('6807c7bcfa8e62a3e63aeee5')
interaction_type_id : ObjectId('6807c6a8fa8e62a3e6365207')
value : 1
```

Figure 20 - An Interaction Record

```
_id: ObjectId('6807c6a8fa8e62a3e6365207')
name : "skipped"
type : 0
```

```
_id: ObjectId('6807c6a8fa8e62a3e6365208')
name : "percent_watched"
type : 1
```

```
_id: ObjectId('6807c6a8fa8e62a3e6365209')
name : "pre_session_duration"
type : 2
```

```
_id: ObjectId('6807c6a8fa8e62a3e636520a')
name : "post_session_duration"
type : 3
```

```
_id: ObjectId('6807c6a8fa8e62a3e636520b')
name : "clicked"
type : 4
```

Figure 21 - Interaction Types

Each interaction record has an associated interaction type and a value.

```python
def simulate_ad_interactions(impression, user_affinity, campaign):
    """
    Simulate user interaction with an ad based on engagement probability and user affinity.

    Args:
        impression: Impression data containing engagement probability
        user_affinity: User's affinity score for the campaign
        campaign: Campaign data containing ad specifications

    Returns:
        dict: Interaction data including watch time, clicks, and session durations
    """
    engagement_probability = impression['engagement_probability']

    # Determine if ad was skipped
    skipped = random.random() > engagement_probability

    # Calculate percent watched
    if not skipped:
        percent_watched = random.uniform(70, 100)
    else:
        max_skip_percent = 30 if campaign['ad_length'] == 15 else 25 if campaign['ad_length'] == 20 else 15
        percent_watched = random.uniform(0, max_skip_percent)

    # Generate session durations
    is_short_form = campaign['content_type'] == 'Short-form'
    pre_base = 120 if is_short_form else 600
    post_base = (60 if skipped else 180) if is_short_form else (120 if skipped else 300)

    # Add more variation to session durations
    pre_session_duration = max(0, random.normalvariate(pre_base, pre_base * 0.4) * user_affinity)
    post_session_duration = max(0, random.normalvariate(post_base, post_base * 0.4) * user_affinity)

    # Click-through rate calculation
    ctr_base = engagement_probability * (percent_watched / 100) * user_affinity
    clicked = random.random() < ctr_base

    return {
        'skipped': skipped,
        'percent_watched': percent_watched,
        'pre_session_duration': pre_session_duration,
        'post_session_duration': post_session_duration,
        'clicked': clicked,
    }
```

Figure 22 - Ad Simulator

Figure 22 simulates various aspects of user engagement with an ad, including whether the ad was skipped, the percentage of the ad watched, session durations before and after the ad, and whether the ad was clicked. These metrics are essential for understanding user behaviour and the effectiveness of ad campaigns.

The function determines if an ad was skipped based on a random comparison with the engagement probability. If the random value exceeds the engagement probability, the ad is considered skipped.

If the ad is not skipped, the percentage watched is randomly set between 70% and 100%. If skipped, the percentage watched is a random value up to a maximum determined by the ad length (e.g., shorter ads have a higher skip percentage).[24]

The function calculates session durations before and after the ad. These durations are influenced by whether the ad was skipped and the user's affinity score. Short-form content has different base durations compared to longer content.

The CTR is calculated based on the engagement probability, the percentage of the ad watched, and the user's affinity score. A random check against this calculated CTR determines if the ad was clicked.

Campaign Analysis Collection:

This collection is used to store all metrics associated with the campaign records. These metrics are mainly used by the dashboard when rendering visualisations. They are also used to train the TensorFlow post-run optimisation model. The analyse_campaign_success() function in figure 19 is used to determine how successful a campaign is based on all relevant details. This gives the models a target column.

The function calculates a seasonality score based on the campaign's start month and the optimal months for the product being advertised. This score reflects how well the campaign's timing aligns with the product's optimal advertising periods.

Interactions are aggregated by impression ID to facilitate quick look up and analysis. This aggregation helps in calculating metrics like completion rates and click-through rates. Completion rates are calculated as the inverse of the 'skipped' interaction value. Click rates are determined by the 'clicked' interaction value. Percent watched is the average percentage of the ad watched by users. Session Durations are the average durations before and after the ad.

Click-through rate (CTR) is the percentage of impressions that resulted in clicks. Cost per impression and cost per click are calculated to assess financial efficiency.

A base score is computed by weighing various metrics equally, including completion rates, percent watched, session durations, cost efficiency, and seasonality.

A random variation is applied to the base score to simulate real-world variability.

The final success score is scaled to a range between 0 and 100.

The function returns a dictionary containing the success_score and a metrics dictionary with detailed performance metrics.

```python
def analyse_campaign_success(campaign, impressions, interactions):
    """
    Analyses if a campaign was successful based on various factors.
    Each factor contributes equally to the final success score.

    Args:
        campaign (dict): Campaign data containing details like start date, product, and budget
        impressions (list): List of impressions for this campaign
        interactions (list): List of interactions for the impressions

    Returns:
        dict: Dictionary containing:
            - success_score (float): Overall success score between 0 and 100
            - metrics (dict): Detailed metrics including completion rates, click-through rates, and costs
    """
    campaign_start = datetime.strptime(campaign['start_date'], '%d-%m-%Y')
    product_rule = next((rule for rule in PRODUCT_RULES if rule[0] == campaign['product']), None)

    campaign_month = campaign_start.month
    optimal_months = product_rule[2] if product_rule else []

    if campaign_month in optimal_months:
        seasonality_score = random.uniform(0.8, 1.0)
    else:
        adjacent_months = []
        for month in optimal_months:
            adjacent_months.extend([(month-1) % 12 or 12, (month+1) % 12 or 12])

        if campaign_month in adjacent_months:
            seasonality_score = random.uniform(0.5, 0.8)
        else:
            seasonality_score = random.uniform(0.1, 0.5)

    impression_interactions = {}
    for interaction in interactions:
        imp_id = interaction['impression_id']
        if imp_id not in impression_interactions:
            impression_interactions[imp_id] = {}
        impression_interactions[imp_id][interaction['interaction_type_id']] = interaction['value']

    interaction_types = list(db_instance.get_collection('interaction_types').find())
    interaction_type_id_map = {it['name']: it['_id'] for it in interaction_types}

    completion_rates = []
    click_rates = []
    percent_watched_values = []
    pre_session_durations = []
    post_session_durations = []
    total_clicks = 0

    for impression in impressions:
        imp_id = impression['_id']

        if imp_id not in impression_interactions:
            continue

        interactions_for_imp = impression_interactions[imp_id]

        completion = 1 - interactions_for_imp.get(interaction_type_id_map['skipped'], 1)
        percent_watched = interactions_for_imp.get(interaction_type_id_map['percent_watched'], 0)
        pre_session_duration = interactions_for_imp.get(interaction_type_id_map['pre_session_duration'], 0)
        post_session_duration = interactions_for_imp.get(interaction_type_id_map['post_session_duration'], 0)
        clicked = interactions_for_imp.get(interaction_type_id_map['clicked'], 0)

        completion_rates.append(completion)
        percent_watched_values.append(percent_watched)
        pre_session_durations.append(pre_session_duration)
        post_session_durations.append(post_session_duration)
        click_rates.append(clicked)
        total_clicks += clicked

    try:
        avg_completion = statistics.mean(completion_rates)
        avg_percent_watched = statistics.mean(percent_watched_values)
        avg_pre_session = statistics.mean(pre_session_durations)
        avg_post_session = statistics.mean(post_session_durations)

        click_through_rate = (total_clicks / len(impressions)) * 100
        cost_per_impression = campaign['cpi']
        cost_per_click = campaign['budget'] / total_clicks if total_clicks > 0 else float('inf')

        base_score = (
            avg_completion * 0.166 +
            (avg_percent_watched / 100) * 0.166 +
            (avg_pre_session / 500) * 0.166 +
            (avg_post_session / 120) * 0.166 +
            min(1.0, (cost_per_impression / (cost_per_click + 1e-6)) if total_clicks > 0 else 0.1) * 0.166 +
            seasonality_score * 0.17
        )

        random_variation = np.random.normal(1.15, 0.25)
        raw_score = base_score * 100 * random_variation

        success_score = min(100, max(0, raw_score * 1.15))

        metrics = {
            'avg_completion_rate': avg_completion,
            'avg_percent_watched': avg_percent_watched,
            'click_through_rate': click_through_rate,
            'cost_per_impression': cost_per_impression,
            'cost_per_click': cost_per_click,
            'avg_pre_session_duration': avg_pre_session,
            'avg_post_session_duration': avg_post_session,
            'total_clicks': total_clicks,
            'total_impressions': len(impressions),
            'seasonality': seasonality_score,
            'placement': campaign['ad_placement'],
            'length': campaign['ad_length'],
            'content_type': campaign['content_type']
        }

        return {
            'success_score': success_score,
            'metrics': metrics
        }
```

Figure 23 - Campaign Success Calculator

3.4. Machine Learning Models

3.4.1. Model Selection

This section outlines the machine learning and deep learning models integrated into the dashboard, detailing the rationale behind their selection, key parameters, and training methodologies. The chosen models aim to balance interpretability, performance, and suitability for the synthetic advertising dataset.

Linear Regression

Why: Linear Regression was chosen for its simplicity and interpretability, making it a good baseline model for predicting the success score.

Prediction Target: Success Score.

Key Parameters/Tuning Strategies: Utilises a preprocessing pipeline to handle numerical and categorical features. No specific hyperparameters were tuned for this model.

Training Methodology: The model is trained using a standard train/test split. The preprocessing pipeline ensures that the data is appropriately scaled and encoded before training.

Ridge Regression

Why: Ridge Regression was chosen to address potential multicollinearity issues by adding a regularisation term, which helps in reducing model complexity and preventing overfitting. According to Chan, "A breakthrough method to solve multicollinearity came in the form of ridge regression." [26]

Prediction Target: Success Score.

Key Parameters/Tuning Strategies: The regularisation strength (α) is a key parameter, with a default value of 1.0. This can be tuned to find the optimal balance between bias and variance.

Training Methodology: Similar to Linear Regression, it uses a train/test split with a preprocessing pipeline.

Random Forest

Why: Random Forest was selected for its robustness and ability to handle non-linear relationships and interactions between features. As noted by Hengl et al, "Random forest is a generic framework for predictive modelling of spatial and spatio-temporal variables." [27]

Prediction Target: Success Score.

Key Parameters/Tuning Strategies: The number of trees (n_estimators) and the maximum depth of the trees (max_depth) are key parameters. These were tuned to optimise model performance.

Training Methodology: The model is trained using a train/test split, with preprocessing to handle feature scaling and encoding.

XGBoost

Why: XGBoost was chosen for its powerful boosting algorithm, which often results in superior performance on structured data. According to Hakkal, " XGBoost is a scalable tree-boosting algorithm designed for high performance, adaptability, and mobility," [28]

Prediction Target: Success Score.

Key Parameters/Tuning Strategies: Important parameters include the number of boosting rounds (n_estimators), maximum tree depth (max_depth), and learning rate (learning_rate). These parameters are crucial for controlling the model's complexity and learning process.

Training Methodology: Utilises a train/test split with a preprocessing pipeline to ensure data is in the correct format.

Gradient Boosting Regressor

Why: This model was chosen for its ability to predict the Click-Through Rate (CTR) by capturing complex patterns in the data. (Gradient Boosting Regressor Why)

Prediction Target: CTR.

Key Parameters/Tuning Strategies: The number of trees (n_estimators) and the maximum depth of the trees (max_depth) are key parameters that were tuned for better performance.

Training Methodology: Similar to other models, it uses a train/test split with preprocessing to handle feature scaling and encoding.

TensorFlow Neural Network + SHAP Explainer

Why: The TensorFlow Neural Network was chosen for its flexibility and ability to model complex, non-linear relationships. The SHAP Explainer is used for interpretability, providing insights into feature importance and model predictions. [29]

Prediction Target: Success Score.

Key Parameters/Tuning Strategies: The model architecture includes embedding layers for categorical features and dense layers for numerical features. The learning rate and batch size are parameters that can be tuned.

Training Methodology: The model is trained using a train/test split with a validation set. The training process includes monitoring of loss and mean absolute error (MAE) to ensure the model is learning effectively. The SHAP Explainer uses a KernelExplainer to provide interpretability by analysing the impact of each feature on the model's predictions.

3.4.2. Model Implementation

Model training, preprocessing and storage were implemented to be as modular as possible. The ModelConfig class is responsible for managing and registering traditional machine learning models. It provides a structured way to handle different models and their configurations.

The class encapsulates the configuration for a machine learning model, including its name, training, evaluation, saving, and loading functions, as well as the features and target column. The register_model function allows models to be registered either as a function or a decorator. It stores the model configurations in a dictionary, _models, for easy retrieval. Functions like get_available_models and get_model_config provide access to the registered models and their configurations. The file includes functions to save and load model metrics to and from JSON files, allowing for persistent storage of model performance data.

```python
class ModelConfig:
    """
    Sci-kit learn model config
    """
    def __init__(self, name, train_func, evaluate_func, save_func, load_func,
                 numerical_features, categorical_features, target_column):
        """
        Initialise model configuration.

        Args:
            name: Name of the model
            train_func: Function to train the model
            evaluate_func: Function to evaluate the model
            save_func: Function to save the model
            load_func: Function to load the model
            numerical_features: List of numerical feature column names
            categorical_features: List of categorical feature column names
            target_column: Name of the column to use as target
        """
        self.name = name
        self.train_func = train_func
        self.evaluate_func = evaluate_func
        self.save_func = save_func
        self.load_func = load_func
        self.numerical_features = numerical_features
        self.categorical_features = categorical_features
        self.target_column = target_column
        self.metrics = None

    def update_metrics(self, metrics):
        """
        Update the metrics for this model configuration.

        Args:
            metrics: Dictionary containing the model metrics
        """
        self.metrics = metrics
```

Figure 24 - Model Config Class

A separate script is used to train the machine learning models. It orchestrates the entire training process, from data preparation to model evaluation. Each model is defined in its respective module within the scikit directory. These modules contain functions for training, evaluating, saving, and loading the models. The models are registered in training.py using the register_model function, which links the model's functions and features to a ModelConfig object.

```python
def train_and_evaluate_model(model_config, X_train, X_test, y_train, y_test):

    model = model_config.train_func(X_train, y_train)

    basic_metrics = model_config.evaluate_func(model, X_test, y_test)

    y_pred = model.predict(X_test)
    y_train_pred = model.predict(X_train)

    cv_scores = cross_val_score(model, X_train, y_train, cv=5, scoring='r2')

    train_metrics = {
        'Training R²': r2_score(y_train, y_train_pred),
        'Training RMSE': np.sqrt(mean_squared_error(y_train, y_train_pred)),
        'Training MAE': mean_absolute_error(y_train, y_train_pred)
    }

    metrics = {
        'Basic Metrics': basic_metrics,
        'Training Metrics': train_metrics,
        'Cross-Validation': {
            'Mean R²': cv_scores.mean(),
            'Std R²': cv_scores.std()
        }
    }

    model_config.metrics = metrics
    save_model_metrics(model_config)

    model_config.save_func(model)

    return model, metrics
```

Figure 25 - Train and Evaluate Function

The TensorFlow model was not included in the model registry as there is a different training, evaluating and processing process.

The model.py file in the neural directory defines a TensorFlow neural network model for predicting campaign success scores.

- The preprocess_data function handles the encoding of categorical features and scaling of numerical features, using LabelEncoder and StandardScaler.
- The build_model function constructs a neural network with embeddings for categorical features and dense layers for numerical features. It compiles the model with an Adam optimiser and mean squared error loss.
- The train_model function trains the neural network using the pre-processed data, with options for batch size, epochs, and validation split.
- **The** evaluate_model function assesses the model's performance using RMSE, MAE, and R^2 metrics.
- Functions are provided to save and load the trained model and its evaluation metrics, ensuring that the model can be reused without retraining.

This structured approach allowed for efficient management, training, and integration of both traditional and neural network models within the project.

3.5. Definition of "Success Score"

The "success score" is a target variable engineered to represent the overall effectiveness of a campaign. It is a composite metric that combines various performance indicators such as Click-Through Rate (CTR), conversions, and other engagement metrics. The exact formulation of the success score is designed to reflect the specific goals and priorities of the campaign, ensuring that the model's predictions align with the desired outcomes.

3.6. Tools and technologies

The development of this dashboard leverages a variety of tools and technologies to ensure robust data processing, model training, and user interaction through a dashboard. Below is an overview of the key tools and technologies used across the project:

Python: The primary programming language used for developing the project. Python's extensive libraries and frameworks make it ideal for data analysis, machine learning, and web development.

NumPy and Pandas: These libraries are used for data manipulation and analysis. They provide powerful data structures and functions to handle large datasets efficiently. [30]

Scikit-learn: Utilised for implementing traditional machine learning models such as Linear Regression, Ridge Regression, Random Forest, and XGBoost. Scikit-learn offers a wide range of algorithms and tools for model training and evaluation. [31]

TensorFlow: Employed for building and training neural network models. TensorFlow's flexibility and scalability make it suitable for complex deep learning tasks. [29]

Streamlit: A framework used to create the interactive dashboard. Streamlit allows for rapid development of web applications with minimal effort, providing a user-friendly interface for visualising and interacting with data. [19]

Plotly: Used for creating interactive visualisations within the dashboard. Plotly's integration with Streamlit enables the rendering of dynamic charts and graphs that enhance data exploration. [20]

LabelEncoder and StandardScaler: Part of Scikit-learn, these tools are used for encoding categorical variables and scaling numerical features, respectively. They ensure that the data is in a suitable format for model training.

Joblib and Pickle: These libraries are used for serialising and deserialising Python objects, such as models and encoders. This allows for efficient storage and retrieval of trained models and preprocessing tools.

Modular Design: The project is organised into distinct modules, each responsible for specific tasks such as data loading, model training, and dashboard rendering. This modular approach enhances maintainability and scalability.

Model Registry: Implemented in registry.py, the model registry manages the configuration and registration of machine learning models. It provides a centralised way to handle different models and their associated functions.

Configuration Management: The config directory is used to store configuration files and settings, ensuring that the project can be easily configured and adapted to different environments or datasets.

Version Control: Git is used for version control, allowing for efficient tracking of changes and collaboration among team members.

Environment Management: Virtual environments are used to manage dependencies and ensure that the project runs consistently across different systems.

By leveraging these tools and technologies, the project achieves a high level of functionality and user engagement, providing valuable insights into advertising campaign performance.

4. Results and Evaluation

4.1. Evaluation Strategy

This section outlines the strategy used to evaluate the machine learning models developed in this project. It focuses on the rationale and methods chosen to assess model accuracy, generalisation, robustness, and transparency, given the limitations of working with synthetic data.

4.1.1. Predictive Accuracy

The primary evaluation criteria for the regression models were:

- R^2 (Coefficient of Determination): Measures the proportion of variance in the dependent variable explained by the model. A higher R^2 indicates better explanatory power.
- MAE (Mean Absolute Error): Captures the average magnitude of errors in predictions, without considering their direction.
- RMSE (Root Mean Squared Error): Similar to MAE but penalises larger errors more heavily, providing a more sensitive measure of error magnitude.

These were applied to predictions on a hold-out test set (20% of the dataset). These metrics were chosen for their complementary properties and widespread use in regression tasks [32].

4.1.2. Generalisation Assessment

Model overfitting was identified by comparing R^2 scores between training and test sets:

- Ridge and Linear Regression had relatively small gaps, indicating good generalisation.
- Random Forest and XGBoost demonstrated overfitting, as shown by large drops in R^2 from training to test sets.

This aligns with known risks of high-capacity models overfitting small or synthetic datasets [33].

4.1.3. Cross-validation for Robustness

All models underwent 5-fold cross-validation to evaluate stability. The mean and standard deviation of R^2 scores were recorded:

- Lower standard deviation suggests consistent performance across folds.
- Ridge Regression was the most stable, with the smallest cross-validation variance.

Cross-validation is a proven method to estimate model performance on unseen data [34].

4.1.4. Residual Analysis

The residuals of Ridge Regression were plotted:

- A random scatter around zero in the residuals vs predicted plot supports model appropriateness.
- The histogram of residuals approximated a normal distribution, validating key assumptions in linear modelling.

These visual checks reinforce the statistical validity of the Ridge Regression model.

4.1.5. Interpretability: SHAP Analysis

The TensorFlow model was analysed using SHAP to improve interpretability:

- The SHAP summary plot highlighted the most influential features on predictions.
- SHAP enables consistent explanations across complex models, which is especially important when using neural networks in high-stakes applications. (Goel 2023)

4.1.6. Learning Dynamics for Neural Network

To evaluate training dynamics, loss and MAE over epochs were plotted for the neural network:

- Rapid early convergence and plateauing validation curves suggest efficient but limited learning capacity.
- Slight underperformance on validation suggests a need for further tuning or architectural adjustments.

4.2. Quantitative Model Results

This section reports the performance outcomes of each machine learning model across the selected metrics, based on the evaluation methods outlined in Section 4.1.

4.2.1. Performance Summary

The following table summarises the test set results for all models:

Model	MAE	RMSE	R^2	Training R^2	Training RMSE	Training MAE	CV Mean R^2	CV Std R^2
ridge_regression	**8.3365**	**10.8829**	**0.494**	**0.6669**	**8.9908**	**6.895**	**0.6223**	**0.0346**
linear_regression	8.4087	10.9642	0.4865	0.6688	8.9654	6.9097	0.6098	0.0341
xgboost	8.5632	11.0956	0.4741	0.6776	8.8454	6.7236	0.6089	0.0388
random_forest	8.6772	11.2218	0.462	0.6465	9.2629	7.0673	0.6033	0.0263

Table 1 - Model Performance

Ridge Regression consistently outperformed others across all metrics, offering the best trade-off between accuracy and robustness. Linear Regression was a close second, indicating the problem structure may favour linear models. XGBoost and Random Forest performed well on training data but showed overfitting during testing.

Figure 26 - Training vs Test Bar

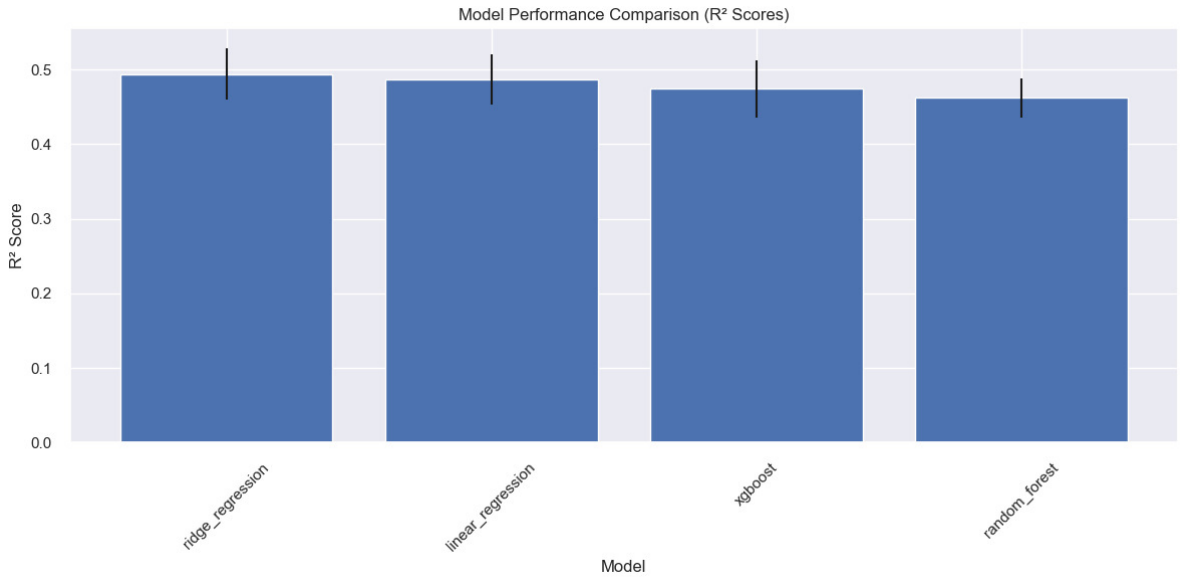

Figure 27 - R^2 Comparison

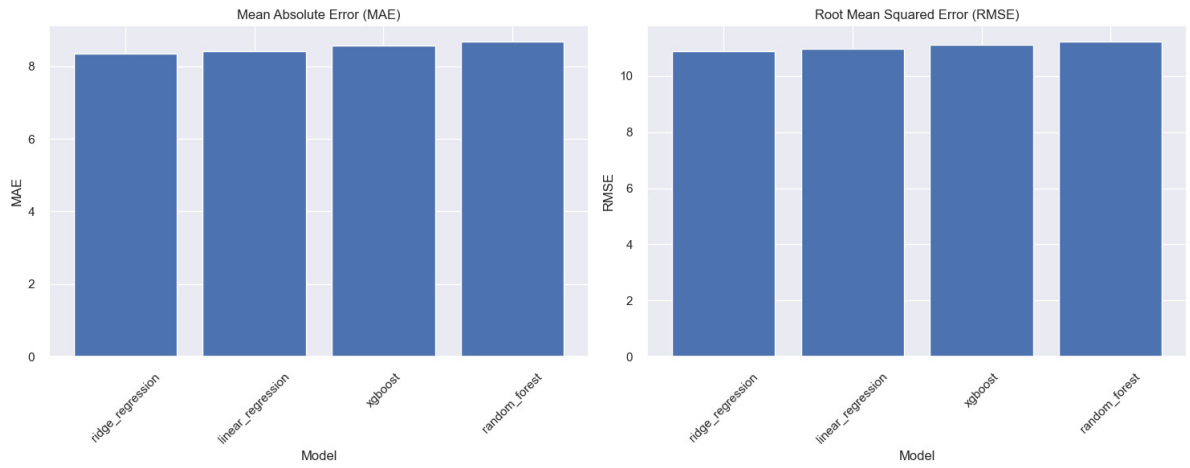

Figure 28 - MAE/RMSE Comparison

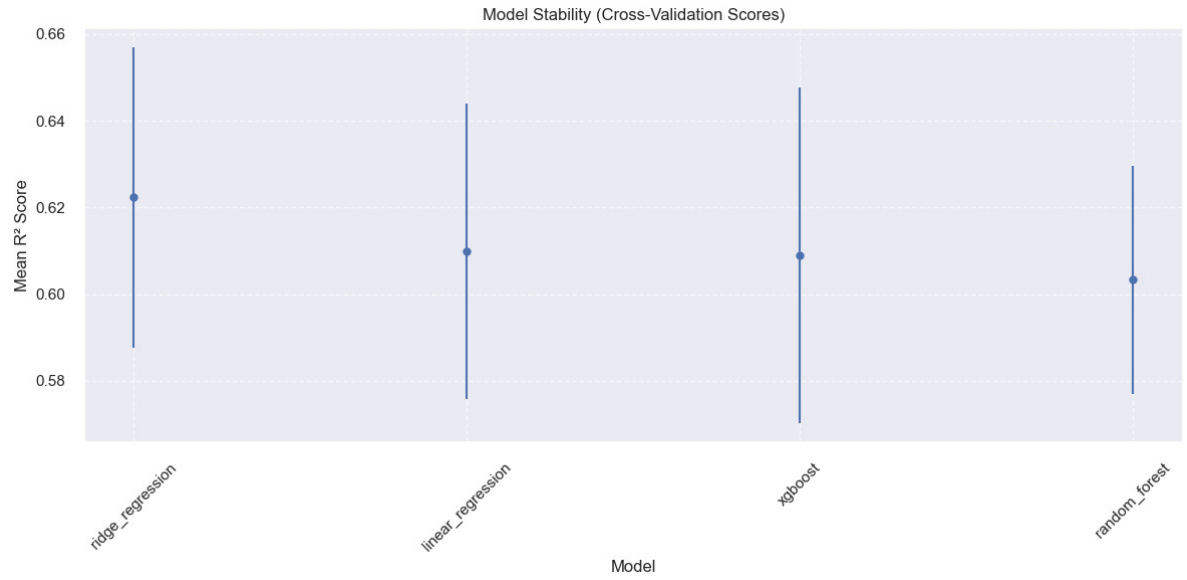

Figure 29 - Cross Validation Scores

4.2.2. Predicted vs Actual Fit

The predicted vs actual plot for Ridge Regression (Figure 31) shows a reasonably strong linear correlation, though some underestimation occurs at higher success scores. This supports the R^2 of 0.494 and shows the model captures broad trends but not extreme values.

Figure 30 - Ridge Predicted vs Actual

4.2.3. Residual Distribution

The residual plot (Figure 32) shows:

- No clear pattern, which is good
- Moderate heteroscedasticity
- A near-normal distribution around zero

These are favourable indicators for a linear model's reliability on this dataset.

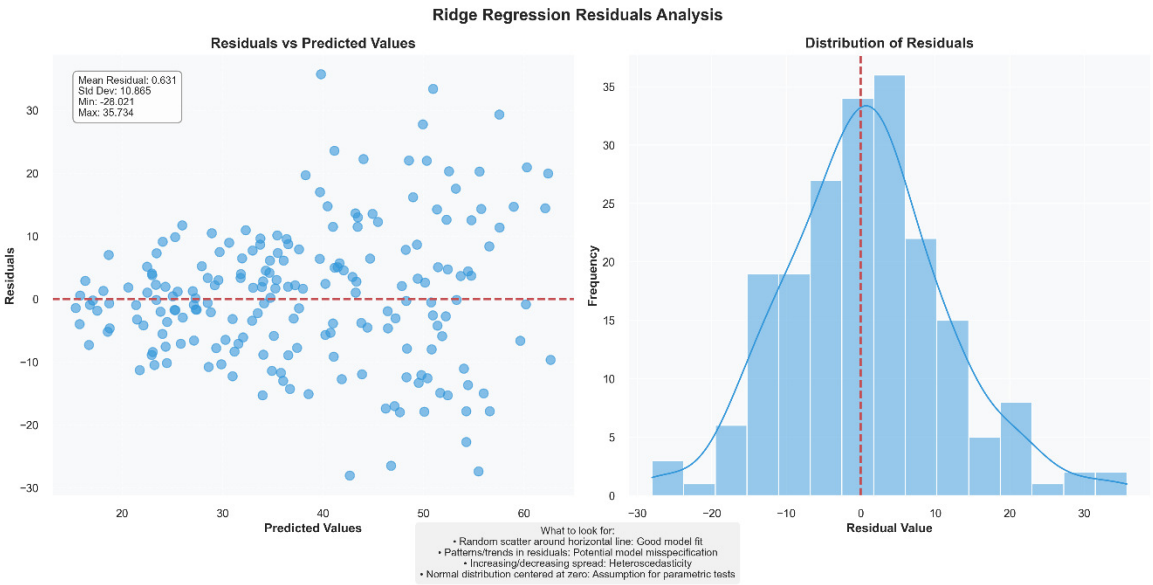

Figure 31 - Ridge Residuals

4.2.4. Parameter Tuning and Optimal Configuration

Each model underwent individual parameter tuning, using grid or manual search strategies:

- Ridge Regression: Optimised α = 10.0 yielded lowest RMSE and highest R^2 (see Figure 35).
- XGBoost: Best performance was achieved at max_depth=3 and learning_rate=0.05, improving balance between bias and variance. (see Figure 36 and 37)
- Random Forest: Stability improved with n_estimators=300, though overfitting remained a concern. (see Figure 33 and 34)

The figures below show all of the performance impacts when the models were tuned. With figure 39 showing the most optimal tuning parameters.

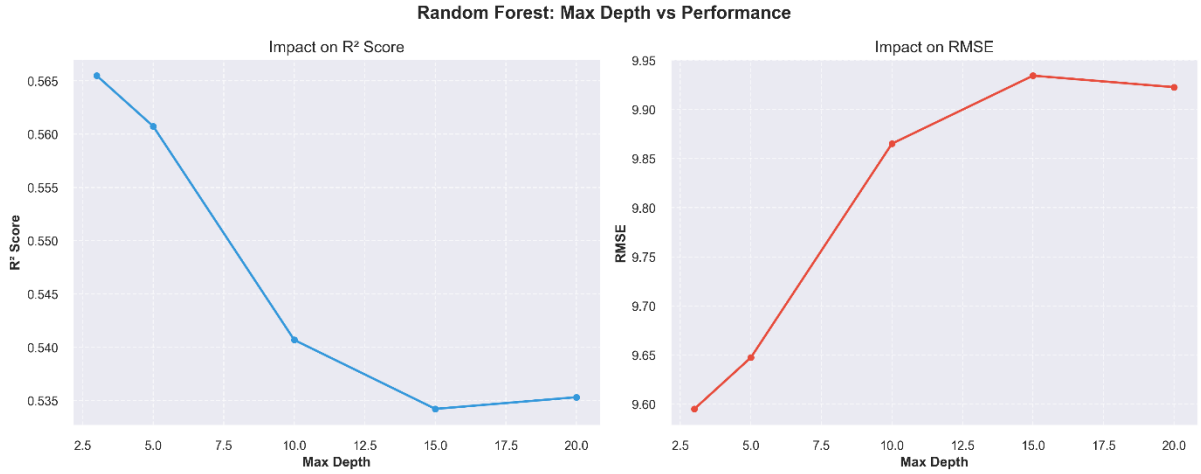

Figure 32 - RF Max Depth Impact

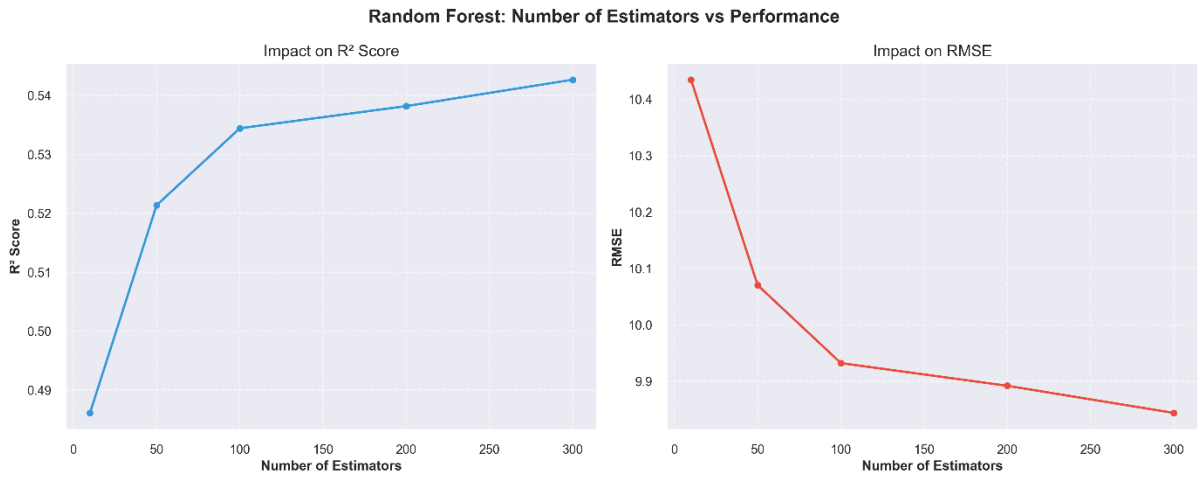

Figure 33 - RF Number of Estimators Impact

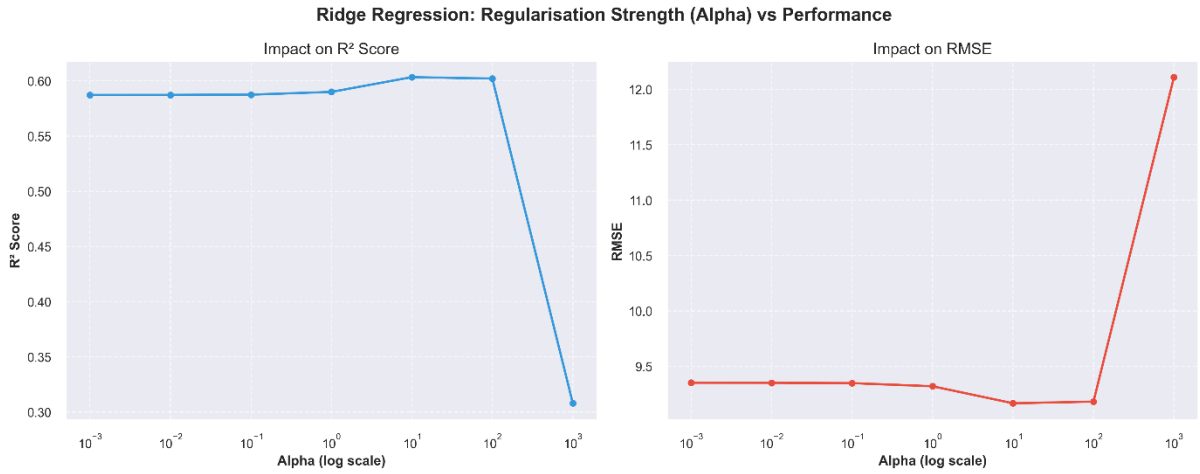

Figure 34 - Ridge Alpha Impact

Figure 35 - XGBoost Learning Rate Impact

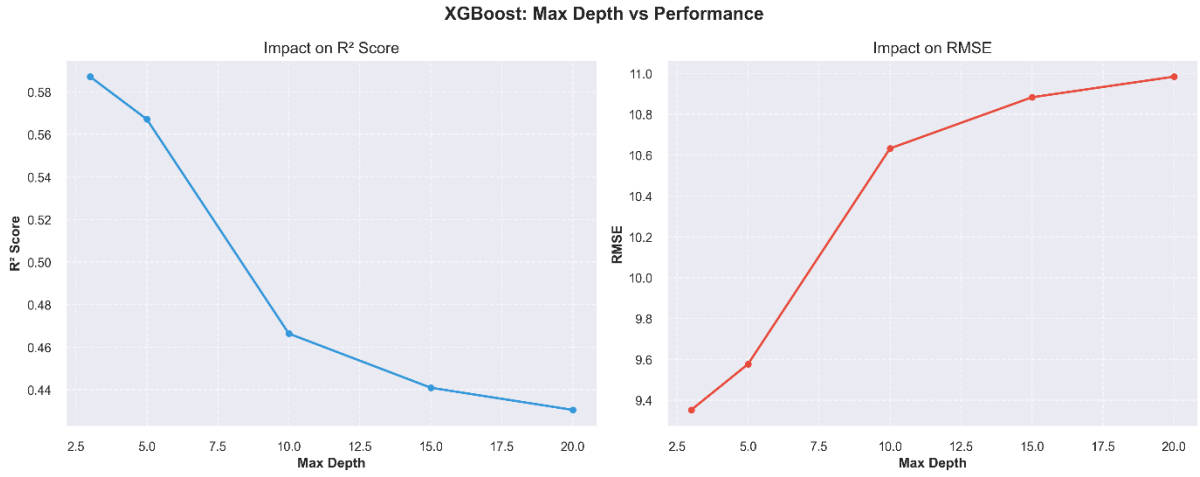

Figure 36 - XGBoost Max Depth Impact

Figure 37 - XGBoost Number of Estimators Impact

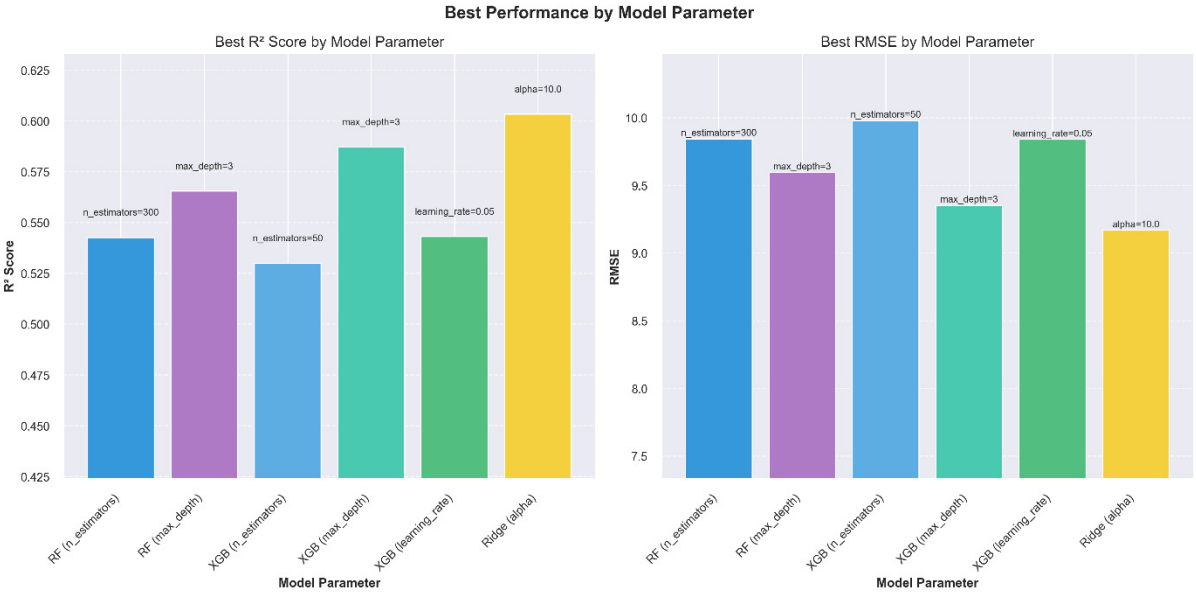

Figure 38 - Best Parameters

This process improved baseline performance in all models, demonstrating that even basic tuning has a measurable impact on regression outcomes [35].

4.2.5. Neural Network Results

As visualised in Figure 40, the model shows:

- Rapid convergence of both training and validation loss within the first 10 epochs.
- A plateau around epoch 20, with training MAE stabilising near 6.06 and validation MAE only slightly higher.
- A final test R^2 of 0.736, which was the highest among all models.

These metrics suggest the model fit the training data well and retained strong generalisation properties despite the complexity of neural architectures.

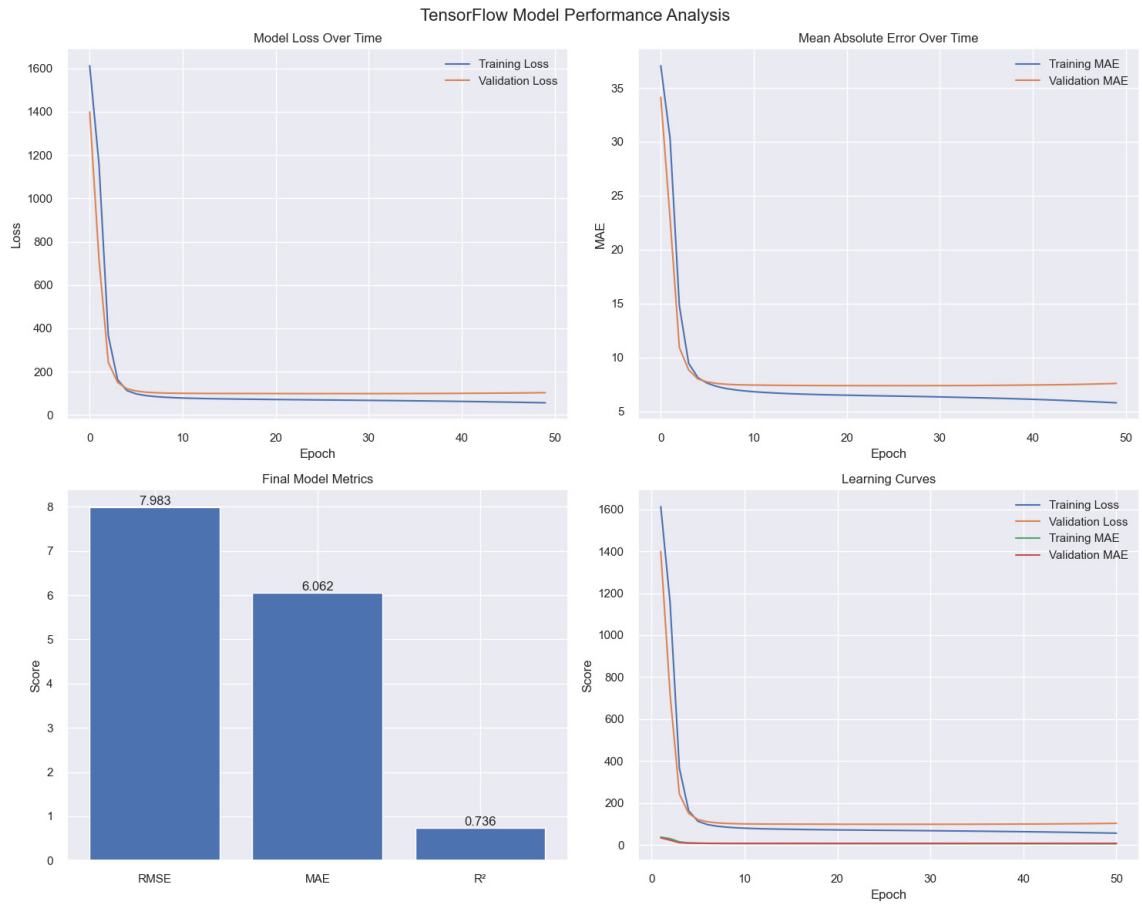

Figure 39 - TensorFlow Analysis

The neural network is the best choice for success score prediction.

4.3. Qualitative Evaluation

While quantitative metrics such as R^2 and RMSE provide insight into model accuracy, they do not capture how understandable or usable a model is to its end users. This section evaluates qualitative aspects of the system, including model interpretability, dashboard usability, and the alignment of explanations with domain expectations.

4.3.1. Model Interpretability with SHAP

The output of SHAP analysis is summarised in Figure 41, which presents the mean absolute SHAP values for all input features. These values reflect the average magnitude of influence each feature had across all predictions, regardless of direction (positive or negative).

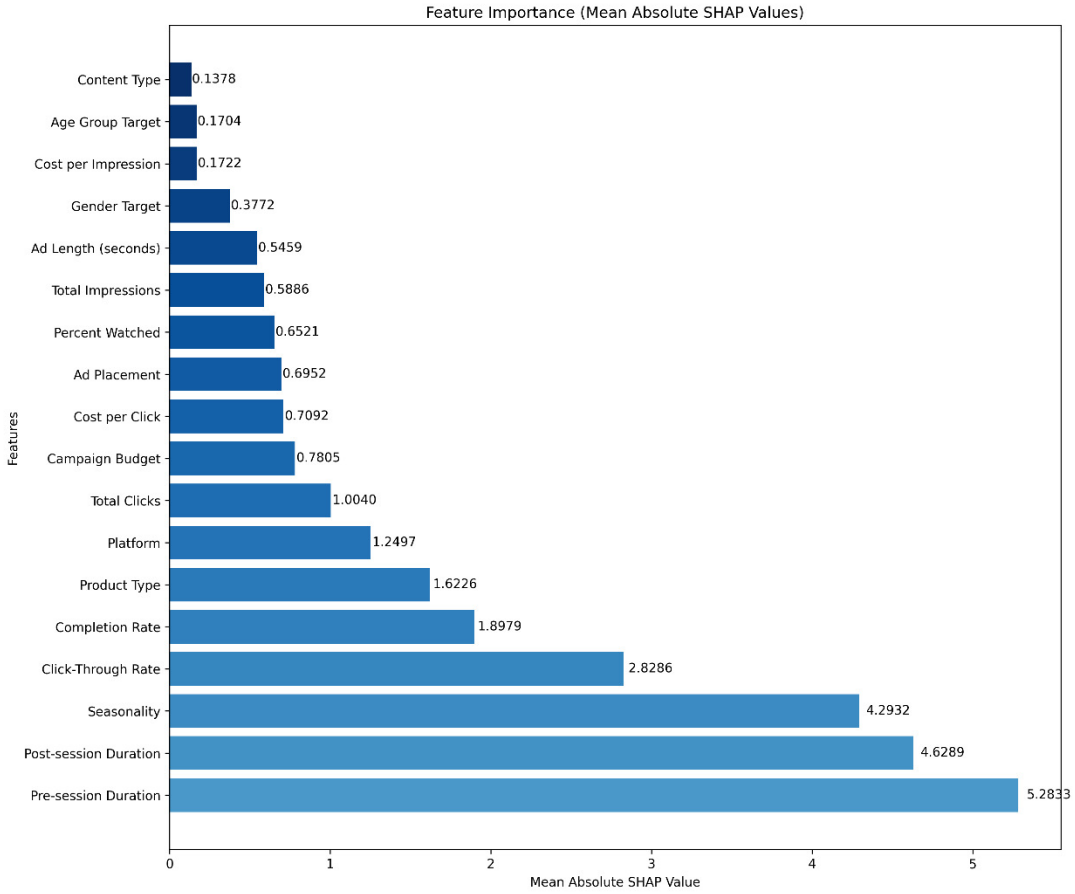

Figure 40 - Absolute SHAP Importance

Key Insights from Global Feature Importance

- Pre-session Duration, Post-session Duration, and Seasonality emerged as the three most influential features. Their high SHAP values suggest these user engagement metrics are crucial predictors of campaign success.
- Click-Through Rate (CTR) and Completion Rate also scored highly, reinforcing their relevance as outcome-linked behavioural metrics.
- Features such as Product Type, Platform, and Total Clicks contributed moderately to the predictions, indicating that content categorisation and delivery medium influence expected performance.
- Conversely, variables like Content Type, Age Group Target, and Cost per Impression had comparatively lower influence on model predictions, suggesting these parameters played a minimal role in shaping the outcome within the synthetic dataset.

This distribution aligns with advertising intuition: user engagement metrics and historical response indicators typically carry stronger signals than demographic or categorical features, especially in digital VOD environments where session behaviour reflects viewer satisfaction.

4.3.2. System Transparency and Trust

From a user-facing perspective, interpretability techniques like SHAP help build trust in the system:

- Even basic plots give users a sense of why the model recommends certain campaign strategies.
- The integration of these explanations into the backend of the dashboard ensures that predictions are not treated as "black boxes."

This aligns well with ACM's position that "transparency must be paired with processes for accountability that enable stakeholders impacted by an algorithmic system to seek meaningful redress for harms done" [3].

4.3.3. Dashboard Usability (Heuristic Review)

To evaluate the usability of the dashboard without access to external participants, a self-guided heuristic evaluation was conducted using Nielsen's 10 Usability Heuristics for user interface design (Nielsen 1994). These heuristics are a widely accepted framework for identifying common usability issues and assessing the intuitiveness, efficiency, and clarity of user interfaces.

Each heuristic was reviewed by interacting with the dashboard from the perspective of a first-time user, completing key tasks such as submitting campaign inputs, generating predictions, and interpreting results.

Heuristic	Description	Strengths Observed	Issues Identified / Comments
1. Visibility of system status	The system should always keep users informed about what is going on.	When a tab is loading, a marked loading indicator is displayed.	
2. Match between the system and the real world	The system should speak the users' language and follow real-world conventions.	Some industry terms are labelled and described. Each visualisation contains an interpretation.	There are still some that are not labelled.
3. User control and freedom	Users need a marked "emergency exit."	The system runs in a browser. Meaning users can close the tab or the browser itself.	There is no exit inside the web application itself.
4. Consistency and standards	Users should not have to wonder whether different elements mean the same thing.	The layout is consistent across the overview and explorer tab.	The Optimisation tab could have a more consistent input form on both the pre-run and post-run tabs

5. Error prevention	The interface should help prevent problems before they occur.	No errors were found when using the app. There is sufficient backend error checking, so if there is, the entire system does not break.	
6. Recognition rather than recall	Reduce memory load by making actions and options visible.	Form input fields contain a tooltip.	
7. Flexibility and efficiency of use	Support both novice and expert users.	Tabs are labelled descriptively, and buttons are easy to understand.	The system would be difficult for someone to use without any knowledge of advertising.
8. Aesthetic and minimalist design	Interfaces should not contain irrelevant or rarely used information.	The interface is aesthetically pleasing and does not instantly overwhelm the user.	The interface contains emojis to aid users, which could be considered unnecessary.
9. Help users recognise, diagnose, and recover from errors	Error messages should be clear and constructive.	Errors are displayed and sometimes contain information on how to fix the error.	Errors may be difficult to understand for a non-developer.
10. Help and documentation	Provide help when needed, even if the system is easy to use.	Tooltips are available.	

Table 2 – Usability Survey

This review helped identify not only strong areas of the dashboard — such as its clarity and simplicity — but also key usability gaps, particularly around user guidance and feedback mechanisms. These observations inform the recommendations for future improvement discussed in Section 4.6.

5. Conclusions and future work

5.1. Summary of project aims and achievements

The central aim of this project was to explore whether key functionalities of commercial advertising analytics platforms—prediction, visualisation, and optimisation—could be replicated using only open-source tools and synthetic data. Motivated by the increasing inaccessibility of proprietary tools for smaller advertisers and researchers, the project set out to deliver a transparent, interpretable, and accessible alternative through a modular dashboard system.

All six project objectives were successfully achieved:

- A statistically grounded synthetic dataset was created to emulate advertising campaign behaviours.
- Multiple machine learning models were developed to predict success score and CTR, with Ridge Regression and the TensorFlow neural network showing strong performance.
- A Streamlit dashboard was implemented to provide interactive campaign analytics and visual feedback.
- SHAP explainability tools were integrated to improve transparency and interpretability.
- Technical feasibility was validated through consistent performance across metrics and folds.
- Open-source components were used throughout to ensure reproducibility and accessibility.

These achievements confirm the project's hypothesis: it is feasible to develop a functional, transparent campaign analytics platform without relying on proprietary datasets or enterprise-scale infrastructure.

5.2. Key findings and insights

- Linear models performed robustly, with Ridge Regression offering the best generalisation and interpretability.
- Complex models like XGBoost and Random Forest overfit due to the relatively small synthetic dataset, highlighting the limits of high-capacity models in constrained data environments.
- Neural networks, when paired with SHAP, provided the best predictive performance (R^2 = 0.736) while retaining some degree of interpretability.
- Synthetic data generation can approximate real-world distributions well, but lacks the complexity needed to fully mimic behavioural noise and edge cases.
- Dashboard usability is strong overall, though a few issues (e.g., terminology clarity, user support mechanisms) remain for improvement.
- Interpretability tools significantly enhanced user trust, making model predictions more transparent and actionable.

5.3. Limitations for the current approach

Despite the positive outcomes, several limitations were identified:

- Synthetic Data Constraints: While realistic, the dataset may not capture the full variability and unpredictability of actual user interactions. This limits external validity.
- Overfitting in Complex Models: Ensemble methods like Random Forest struggled to generalise, reinforcing the need for careful model selection when working with limited or synthetic datasets.
- Scalability Concerns: The current system has not been tested on large-scale datasets or with concurrent users, which may reveal performance bottlenecks.

- Lack of Real User Testing: Usability evaluation was based on self-guided heuristics rather than real advertiser feedback, limiting insights into true end-user needs and satisfaction.
- Limited Model Types for Optimisation: Post-run recommendations were only implemented using a single TensorFlow model, rather than evaluating multiple optimisation strategies.

5.4. Recommendations for next steps

To build upon the current system and address its limitations, several areas for future development are suggested:

- Collaborating with advertisers to obtain anonymised or partially synthetic datasets could enhance realism, improve model calibration, and expand the system's practical relevance.
- Implementing model ensembling or stacking methods could balance generalisation and accuracy. Additionally, exploring Bayesian models or quantile regression could improve uncertainty estimation.
- Introduce user-specific guidance (e.g., tooltips, tutorials, dynamic recommendations) based on actual campaign behaviour. Adding a feedback loop where users can rate model suggestions may also improve learning over time.
- Formal usability testing with real advertisers or students would provide richer insights than heuristic evaluation alone. This could guide interface changes and feature prioritisation.
- Future iterations could support real-time campaign data ingestion and alerting mechanisms, increasing the tool’s relevance for live performance tracking.
- Introduce features that simulate GDPR compliance checks, bias detection in targeting, or ethical flagging systems—useful for advertisers looking to audit their campaign strategies.

5.5. Conclusion

This project has demonstrated that it is technically and practically feasible to deliver a transparent, predictive advertising analytics system without relying on proprietary data or infrastructure. While there are challenges to overcome—particularly around data realism and system scalability—the foundational system provides a compelling alternative for education, research, and small-scale commercial use. It offers a springboard for further research and development in democratising algorithmic advertising insights.

6. Reflection on Learning

- Gained a deeper understanding of the end-to-end machine learning pipeline, from data simulation and preprocessing to model training, evaluation, and deployment within an interactive system.

- Developed strong proficiency in structuring modular Python code for maintainability and scalability, particularly through the creation of model registries and configuration management systems.
- Improved my ability to critically assess model performance beyond basic metrics by incorporating residual analysis, cross-validation, and SHAP interpretability.
- Learned how to design and simulate realistic synthetic datasets that preserve key behavioural distributions, while recognising the limitations of such data in capturing edge-case behaviour and real-world noise.
- Strengthened my understanding of user interface design principles through the heuristic evaluation process and by integrating visual analytics into the Streamlit dashboard.
- Gained valuable experience in managing project scope and prioritisation under academic deadlines, balancing technical ambition with pragmatic implementation.
- Developed a more nuanced appreciation for the ethical and regulatory considerations in AI development, particularly regarding data transparency, fairness, and user trust.
- Learned how to communicate technical concepts to a mixed audience, combining detailed machine learning explanations with intuitive dashboard interfaces for non-technical users.
- Realised the importance of iterative development and testing, and how early prototyping can uncover design issues or edge cases that influence downstream modelling choices.
- Gained confidence in using tools like TensorFlow, XGBoost, SHAP, and MongoDB in a cohesive system, rather than in isolation, enhancing my ability to build integrated ML-powered applications.

7. Appendices

```
1. Create environment

python -m venv .venv

2. Install dependencies

pip install -r requirements.txt

3. Generating dataset

python src/data_generation/generate_data.py

(This will take some time)

4. Train ML models

python src/models/training.py

5. Train DL model

python src/models/neural/train.py

6. Start dashboard

streamlit run dashboard.py
```

Figure 41 - Instructions to run the dashboard

8. References

[1] T. Witherow, "Half of UK advertising spend went to Google and Meta last year," *Thetimes.com*, 2025. https://www.thetimes.com/uk/technology-uk/article/half-of-uk-advertising-spend-went-to-google-and-meta-last-year-vx7lvnh5x (accessed May 07, 2025).

[2] P. S. Zuboff, *The Age of Surveillance Capitalism: The Fight for the Future at the New Frontier of Power*, Main. Profile Books, 2019, pp. 12–14.

[3] A. T. Policy Council, "Statement on Principles for Responsible Algorithmic Systems," ACM, 2022. Accessed: Jan. 01, 2025. [Online]. Available: https://www.acm.org/binaries/content/assets/public-policy/final-joint-ai-statement-update.pdf?

[4] N. Lomas, "Big Tech's ad transparency tools are still woeful, Mozilla research report finds | TechCrunch," *TechCrunch*, 2024.

https://techcrunch.com/2024/04/15/mozilla-checkfirst-ads-transparency-report/ (accessed Jan. 01, 2025).

[5] P. Brace, “Privacy Policies Explained: Ensuring Transparency Under the GDPR,” *Urmconsulting.com*, 2025. https://www.urmconsulting.com/blog/privacy-policies-explained-ensuring-transparency-under-the-gdpr (accessed Jan. 01, 2025).

[6] A.-A. A. Sharabati, A. Ali, M. I. Allahham, A. A. Hussein, A. F. Alheet, and A. S. Mohammad, “The Impact of Digital Marketing on the Performance of SMEs: An Analytical Study in Light of Modern Digital Transformations,” *Sustainability*, vol. 16, no. 19, pp. 8667–8667, 2024.

[7] E. A. van Reijmersdal and E. Rozendaal, “Transparency of digital native and embedded advertising: Opportunities and challenges for regulation and education,” *Walter de Gruyter GmbH*, 2020, doi: 10.1515/commun-2019-0120.

[8] E. Union, “The Digital Markets Act: Fair and competitive digital markets,” *European Union*, Mar. 25, 2024. https://data.europa.eu/en/news-events/news/digital-markets-act-fair-and-competitive-digital-markets (accessed May 07, 2025).

[9] S. Singh, “Open Source Repository for Industry Collaboration, Adoption, and Trust,” *IAB. Tech Lab*. https://iabtechlab.com/iab-tech-lab-launches-open-source-initiative-to-increase-cross-industry-transparency-and-collaboration-and-trust/ (accessed May 07, 2025).

[10] A. Dicks and J. Piekarski, “What is synthetic data? A definition for digital marketers,” *Impression*, Feb. 23, 2024. https://www.impressiondigital.com/blog/what-is-synthetic-data/ (accessed May 07, 2025).

[11] K. Wakefield, “Predictive Modeling Analytics and Machine Learning,” *SAS UK*. https://www.sas.com/en_gb/insights/articles/analytics/a-guide-to-predictive-analytics-and-machine-learning.html (accessed May 07, 2025).

[12] A. Goel, “Model Explainability using SHAP (SHapley Additive exPlanations) and LIME (Local Interpretable Model-agnostic Explanations),” *Medium*, Oct. 13, 2023. https://medium.com/@anshulgoel991/model-exploitability-using-shap-shapley-additive-explanations-and-lime-local-interpretable-cb4f5594fc1a#id_token=eyJhbGciOiJSUzI1NiIsImtpZCI6IjA3YjgwYTM2NTQyODUyNWY4YmY3Y2QwODQ2ZDc0YThlZTRlZjM2MjUiLCJ0eXAiOiJKV1QifQ.eyJpc3MiOiJodHRwczovL2FjY291bnRzLmdvb2dsZS5jb20iLCJhenAiOiIyMTYyOTYwMzU4MzQtazFrNnFlMDYwczJ0cDJhMmphbTRsamRjbXMwMHN0dGcuYXBwcy5nb29nbGV1c2VyY29udGVudC5jb20iLCJhdWQiOiIyMTYyOTYwMzU4MzQtazFrNnFlMDYwczJ0cDJhMmphbTRsamRjbXMwMHN0dGcuYXBwcy5nb29nbGV1c2VyY29udGVudC5jb20iLCJzdWIiOiIxMDY4NDEyMzc1NjE4OTQ3NjIyMzIiLCJlbWFpbCI6ImphY2t

0aHVybWFuOEBnbWFpbC5jb20iLCJlbWFpbF92ZXJpZmllZCI6dHJ1ZSwibmJmIjo
xNzQ2NjYzMzg0LCJuYW1lIjoiSmFjayBUaHVybWFuIiwicGljdHVyZSI6Imh0dHBzO
i8vbGgzLmdvb2dsZXVzZXJjb250ZW50LmNvbS9hL0FDZzhvY0tuSVFEcXRVb1dW
Yl9TRi1WTmFUVHctQ0JBdTdPNUp6WDh1YlBVcm9zbEQ0TlJSY3c9czk2LWMiLC
JnaXZlbl9uYW1lIjoiSmFjayIsImZhbWlseV9uYW1lIjoiVGh1cm1hbiIsImlhdCI6MTc
0NjY2MzY4NCwiZXhwIjoxNzQ2NjY3Mjg0LCJqdGkiOiJlZTVlMWQ3MWM0OWI0Y2
NhMTQ3MjFkNDBhNWMxNGZiOTA3ZmRkNDI1In0.ZO1_s_WDrhJBOB16i1Y9ks-
SXT70o4Oeyx3dMVtvFQABwulDXGDhZdbHQ1R1Q_1BGPRRNwdWfvYFj4Ydfwgc
6dBl9xZnQYGIMl5ZWpWrvOEd9-WGWud7ugqOiDocbNJ6XFeTn-
rlgQUU5NT7JkbxzT7bDGjbTNgFQ5MbIJ8edgiEK0996dpiQj3yWL4DAs4RF1O6Tby
xBO3tV9nUyLLhfOTMEeds1riKVjRnyDYFvXLgBPsQBrst9VUhOSVHpHoIdRPXcCC
SRWCxF-lfv7c9-
XlFLv_0YYzETvOQpct_auqXMvoQ2uUCUarfe0gHvXT5zCwxgwF8EWEDYJgvjyhye
Q (accessed May 08, 2025).

[13] A. Undru, “Visual analytics: Tools, benefits, types, and more,” *ThoughtSpot*, May 08, 2024. https://www.thoughtspot.com/data-trends/analytics/visual-analytics (accessed May 08, 2025).

[14] ico., “Purpose limitation, data minimisation and storage limitation,” *Information Commissioners Office*. https://ico.org.uk/for-organisations/direct-marketing-and-privacy-and-electronic-communications/guidance-for-the-use-of-personal-data-in-political-campaigning-1/purpose-limitation-data-minimisation-and-storage-limitation (accessed May 08, 2025).

[15] A. Ataman, “Synthetic Data vs Real Data: Benefits, Challenges in 2025,” *AIMultiple Research*, Mar. 21, 2025. https://research.aimultiple.com/synthetic-data-vs-real-data/ (accessed May 08, 2025).

[16] A. McAlpin, “Transparency in Advertising: The Push for Greater Clarity and Accountability,” *RockerBox*, Jul. 09, 2024. https://www.rockerbox.com/blog/transparency-in-advertising-the-push-for-greater-clarity-and-accountability (accessed May 08, 2025).

[17] R. M. Santini, D. Salles, B. M. Martins, A. Moreira, and J. G. Haddad, “Seeing through opacity: The limitations of digital ad transparency in Brazil,” in *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, New York, NY, USA, Jun. 2024, pp. 2209–2221, doi: 10.1145/3630106.3659034.

[18] C. Haury, “The Pros and Cons of Open-source Software in Data Analytics,” *Panoply*, Dec. 06, 2021. https://blog.panoply.io/pros-and-cons-of-open-source-in-data-analytics (accessed May 08, 2025).

[19] S. Inc, *Streamlit*. Snowflake Inc, 2025.

[20] P. Python, *Plotly Python Graphing Libraries*. Plotly, 2025.

[21] M. Inc, *PyMongo*. PyMongo: MongoDB, 2008.

[22] T. V. Advertising, “VOD Advertising,” *tvadvertising.co.uk*. https://tvadvertising.co.uk/vod-advertising/ (accessed May 12, 2025).

[23] D. Faraglia, *Faker*. Daniele Faraglia, 2014.

[24] S. S. Krishnan and R. K. Sitaraman, “Understanding the effectiveness of video ads: A measurement study,” in *Proceedings of the 2013 conference on Internet measurement conference*, New York, NY, USA, Oct. 2013, pp. 149–162, doi: 10.1145/2504730.2504748.

[25] K. P. Murphy, *Machine learning: A probabilistic perspective*. Cambridge, MA, 2012.

[26] J. Y.-L. Chan *et al.*, “Mitigating the multicollinearity problem and its machine learning approach: A review,” *Mathematics*, vol. 10, no. 8, p. 1283, Apr. 2022, doi: 10.3390/math10081283.

[27] T. Hengl, M. Nussbaum, M. N. Wright, G. B. M. Heuvelink, and B. Gräler, “Random forest as a generic framework for predictive modeling of spatial and spatio-temporal variables.,” *PeerJ*, vol. 6, p. e5518, Aug. 2018, doi: 10.7717/peerj.5518.

[28] S. Hakkal and A. A. Lahcen, “Xgboost to enhance learner performance prediction,” *Computers and Education: Artificial Intelligence*, vol. 7, p. 100254, Dec. 2024, doi: 10.1016/j.caeai.2024.100254.

[29] O. TensorFlow, *TensorFlow Documentation*. 2025.

[30] D. NumPy, *NumPy Documentation*. NumPy, 2008.

[31] L. SciKit, *SciKit Learn*. SciKit Learn Developers, 2007.

[32] T. Chai and R. R. Draxler, “Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature,” *Geosci. Model Dev.*, vol. 7, no. 3, pp. 1247–1250, Jun. 2014, doi: 10.5194/gmd-7-1247-2014.

[33] D. M. Hawkins, “The problem of overfitting.,” *J. Chem. Inf. Comput. Sci.*, vol. 44, no. 1, pp. 1–12, Feb. 2004, doi: 10.1021/ci0342472.

[34] P. Refaeilzadeh, L. Tang, and H. Liu, “Cross-Validation,” in *Encyclopedia of database systems*, L. Liu and M. T. Özsu, Eds. New York, NY: Springer New York, 2016, pp. 1–7.

[35] P. Probst, M. N. Wright, and A. Boulesteix, “Hyperparameters and tuning strategies for random forest,” *WIREs Data Mining Knowl Discov*, vol. 9, no. 3, p. e1301, May 2019, doi: 10.1002/widm.1301.