Analysis and Development Report: "AI-Powered Daily Revenue Prediction Model"

Prepared by:-

- Engineer / Mohamed Hassan
- Engineer / Mohamed Khalaf
- Engineer / Mohamed Eissa
- Engineer / Rawan Ali
- Engineer / Nouran Mohamed
- Engineer / Asmaa Esmaeil

****

Produced As :- Acamedic Project

Date:- December 2025

Abstract

This report presented the development and implementation of an **AI-powered Revenue Prediction System** designed for **Artificial Intelligene *AI* Acamedic** Project which aimed to predict daily revenue, deployed within an interactive dashboard using Streamlit. It covers the stages of Exploratory Data Analysis (**EDA**), the application of advanced feature engineering techniques (Polynomial Features), model training (**XGBoost**), and critically, the resolution of critical coding issues encountered during deployment to ensure the accuracy and integrity of the dashboard functionality.

The solution integrates advanced data **preprocessing**, feature engineering, and a finely tuned XGBoost regression model to accurately forecast daily revenue and identify key performance drivers.

Through the deployment of an interactive Streamlit **dashboard**, the system transforms complex data into actionable insights, empowering management to make data-driven strategic decisions with greater confidence and agility.

The project demonstrates how artificial intelligence can effectively enhance business intelligence processes and revenue forecasting accuracy.
Future improvements may include the integration of real-time data streams, automated model retraining, and enhanced visualization capabilities to further strengthen long-term business growth and performance monitoring.

Table of Contents

1. Exploratory Data Analysis (EDA)

The `source_code.py` script was utilized for data processing, preparation, and identifying the
relationships between variables. The data analysis revealed several key insights:

1.1 Statistical Feature Analysis

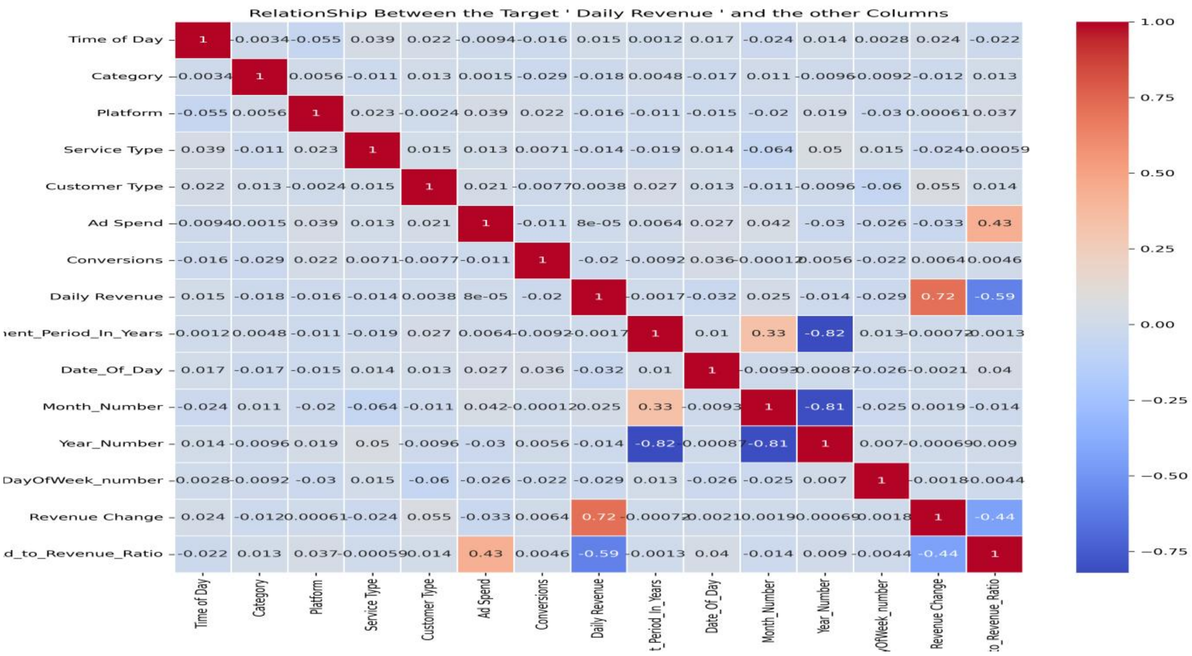

Figure 1: Correlation Matrix between Daily Revenue and other variables.

The matrix highlights

● Moderate Correlation: Most derived temporal variables (e.g., `Date_Of_Day`,`Month_Number`, `DayOfWeek_number`) show relatively low correlations with Daily
Revenue.

● Strongest Predictors: `Ad_to_Revenue_Ratio` and `Revenue Change` exhibit the
strongest correlation with Daily Revenue, underscoring their critical importance in the prediction process.

1.2 Categorical Feature Distribution

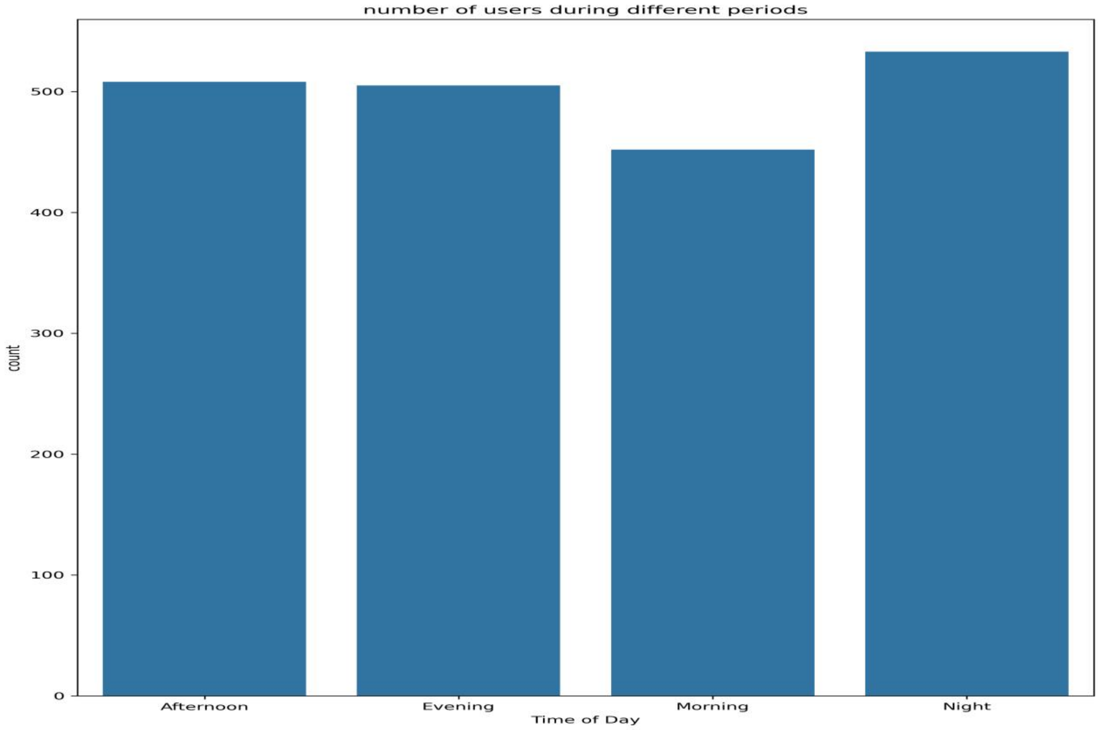

Figure 2: User count distribution by Time of Day

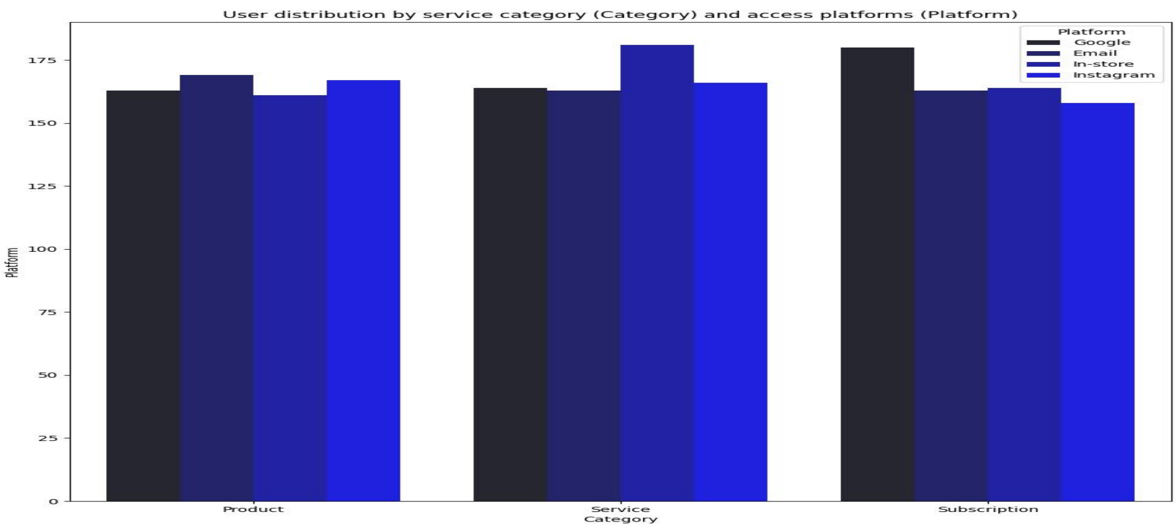

Figure 3: User distribution by (Category) and access platforms (Platform).

1.3 Effect of Platform Type on Revenue

The figure below shows that different platforms contribute variably across different revenue ranges,confirming the necessity of including this feature in the model.

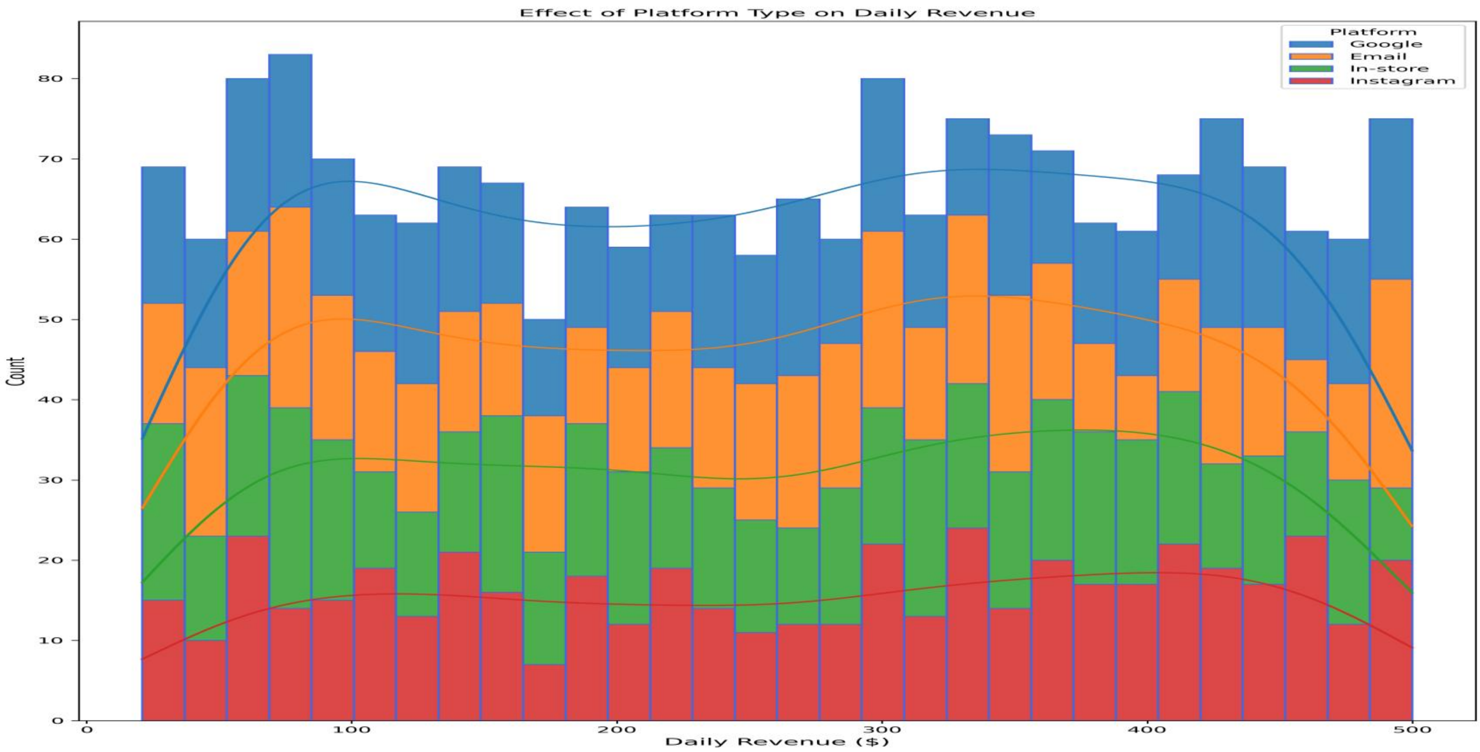

Figure 4: The effect of Platform Type on Daily Revenue.

2. Feature Engineering and Model Construction

2.1 Feature Engineering Steps

The following steps were applied to prepare the data:

1. **Date Transformation**: Multiple temporal features were derived, such as `Date_Of_Day`,`Month_Number`, `Year_Number`, and `DayOfWeek_number`.

2. **Derived Variables**: `Revenue Change` and `Ad_to_Revenue_Ratio` were calculated.

3. **Encoding:** Categorical variables (e.g., `Time of Day`, `Category`) were converted into numerical values.

4. **Polynomial Features:** The `PolynomialFeatures` function with a degree of two was used to create interaction terms among the 14 original features, increasing the total feature count to 120. This step is crucial for the XGBoost model to capture complex, non-linear relationships.

2.2 Training and Results

The XGBoost Regressor model was trained with the following parameters: n_estimators=300, learning_rate=0.05, max_depth=6.

- Training R^2 Score: 0.94
- Testing R^2 Score: 0.89

Technical Note: The `poly_transformer.pkl` and `scaler_project.pkl` were saved to ensure that the same transformations are applied to the input data within the Streamlit production environment.

3. Documentation of Streamlit-dashboard Adjustments and Fixes

The Streamlit dashboard application encountered a critical error, and fundamental modifications were made to the code to resolve the issue and refine the logic for the Key Performance Indicators (KPIs).

3.1 Fixing the Feature Mismatch Error (14 vs 120)

The error arose because the `StandardScaler` expected 120 features (created by`PolynomialFeatures` , but only 14 raw features were being passed from the user interface.Key Adjustments Made:

1. In `source_code.py`: The polynomial feature transformer (`poly`) was saved to a new filenamed `poly_transformer.pkl`. ```python

source_code.py (Adjustment)

poly = PolynomialFeatures(degree=2) x = poly.fit_transform(x) joblib.dump(poly,
'poly_transformer.pkl') ```

- In `main.py`: The `poly_transformer.pkl` was loaded and applied to the 14 raw input features before passing them to the `StandardScaler`. ```python

main_py (Adjustment)

poly_transformer = joblib.load('poly_transformer.pkl')
...
raw_input_data = raw_input_data[feature_names] input_poly
=

poly_transformer.transform(raw_input_data) input_scaled = scaler.transform(input_poly) ```

3.2 Improving the "Change \Delta" Metric Logic

The "Change \Delta" metric consistently displayed a static value (0.000). The logic was overhauled to reflect the true meaning of a performance indicator: the difference between the current day's predicted revenue and the last known daily revenue.Key Adjustments Made:

1. Defining Last Known Revenue: The last known daily revenue
(`LAST_KNOWN_REVENUE`) is determined using the `last_value_added()` functiononce at startup.

2. **New Change Calculation Logic:** The formula within the prediction button was changed to **Enhanced Visualization** as A visual indicator (up arrow ▲ or down arrow ▼) and dynamic colors green/red) were added for clarity. ```python

main_py (Updated Change Metric Logic)

... after calculating pred_value ...
change_value = pred_value - LAST_KNOWN_REVENUE
if change_value >= 0: change_symbol = "▲"
change_color = "text-green-500" else:
change_symbol = "▼" change_color = "text-red-500"
c3.markdown(f""" <div class='metric-card'> <h3>Change Δ</h3> <p class='{change_color}'>{change_symbol} {abs(change_value):.3f} $</p> </div> """,unsafe_allow_html=True) ```

4. Conclusion and Recommendations

The development and deployment of the Daily Revenue Prediction Model using **XGBoost** was successful. Technical issues related to feature dimension mismatch were resolved using the saved `**PolynomialFeatures**` transformer, and the logic of the "Change \Delta" metric was improved to reflect the actual expected change compared to previous performance.

Future Recommendations:

● **Integrate** a Continuous Learning feature to automatically update the model with the latest revenue data.

● **Explore** alternative models, such as Recurrent Neural Networks (RNNs), which are specialized for time-series data.

● Add a **"What-If Analysis" feature**, allowing users to adjust inputs like "Ad Spend" and immediately see its impact on the prediction.

User Guide

To operate and interact with the AI Revenue Prediction Dashboard, please follow the steps below:

- **Open the [AI Revenue Prediction]() Dashboard , provided in the deployment section.**
- **Select the desired business parameters, such as Date, Time of Day, Ad Spend, Category, Platform, Service Type, and Customer Type.**
- **Click on “Predict Daily Revenue” to generate an AI-based forecast.**
- **Review the Key Performance Indicators (KPIs) displayed on the dashboard, including the predicted revenue, advertising ratio, and the daily change (Δ) indicator.**
- **Use these insights to make informed, data-driven business decisions and adjust marketing or service strategies accordingly.**