📊 AI Campaign Cost 2023–24 Project

Overview

This project focuses on predicting campaign costs using data from the 2023–24 fiscal year. It involves cleaning, analyzing, and preparing the dataset for AI-driven cost prediction. Insights are drawn from campaign strategies, video formats, and influencer metrics. Emphasis is placed on robust data cleaning, visual exploration, and correlation analysis to optimize model input quality.

📁 Dataset

Name

G-Sheet Name - New_AI_Final Data

Source

(MyDrive/Colab Notebooks/1. Project 1: AI CAMPAIGN COST 2023-24 PROJECT (Done)/AI Cost Data.gsheet)

The dataset is hosted on Google Sheets and accessed programmatically by exporting it as a CSV file.

Explanation: Instead of manually downloading the file, a direct link is used to fetch the data in `.csv` format via the Google Sheets export URL. This method ensures the data is always up-to-date and can be integrated directly into data pipelines or notebooks.

🔍 Feature Descriptions

Feature Description

Brand_name - The advertiser or company running the campaign (e.g., mStock, Octa).

Product_name - Specific product or service promoted in the video.

yt_username - YouTube influencer’s channel name or handle.

Overall_videos_viewcount_of_channel - Total lifetime views across all videos on the influencer's channel.

Channels_subscriber - Total number of subscribers on the influencer’s channel.

First_five_avg_video_of_channel - Average performance metrics (like views or likes) from the first five videos on the channel.

Engagement_rate_of_channel - Engagement level of the audience, typically derived from subscribers and first 5 video’s avg of channel

Video_views - Number of views for the specific campaign video.

Video_duration - Length of the campaign video (usually in seconds).

Category - Type of content (e.g., Finance, Tech, Lifestyle).

Video_type - Format of video (e.g., Dedicated, Integrated, YouTube Shorts, Live).

🧰 Tools & Libraries Used

- `pandas` – data manipulation
- `numpy` – numerical operations
- `matplotlib` & `seaborn` – visualizations (e.g., heatmaps, box plots, scatter plots)

🔍 Analysis Workflow

Data Cleaning Process
1. Data Cleaning with pandas

2. Heatmap-Based Correlation Analysis
video_cost vs video_type
video_cost vs category
video_cost vs engagement_rate_of_channel

3. Outlier Detection with Scatter & Box Plots

4. Brand-Specific Outlier Cleaning
Each brand had its own subset of outliers removed using IQR (Interquartile Range)

Notable Features

- **Outlier Removal Strategy**:
  Each category of campaign is analyzed and cleaned individually.
  A sequence of before-and-after visuals highlights the effect of outlier filtering.

5. Feature Engineering Before Feeding to Model
After testing with many models, both RandomForestRegressor and XGBRegressor provided good accuracy for predicting campaign costs.

🧬 Feature Encoding Techniques
Tree-based models such as RandomForestRegressor and XGBRegressor can handle both One-Hot Encoding and Label Encoding effectively.

Both encodings were tested independently on the same dataset using the same models, to normalize the comparison and identify the better fit for cost prediction.

Here strategy to try both encodings and compare results was absolutely correct and aligned with best practices in machine learning

ML Model Learning & Evaluation

Several regression models were applied and evaluated using error metrics:

🔹 Linear Regression

- Result: Poor accuracy on test data.

🔹 Ensemble Models - Bagging

- RandomForestRegressor:
  - Provided stable and accurate results.
  - Tested with both One-Hot and Label Encoded datasets.
- ExtraTreesRegressor:
  - Similar ensemble behavior; tested alongside RF.

🔹 Ensemble Models - Boosting

- GradientBoostingRegressor : Result: Poor accuracy on test data.
- XGBRegressor:
  - Slightly lower performance than RF but still strong.
  - Evaluated with both encoding techniques.

🔹 Support Vector Regression (SVR)

- Result: Poor accuracy on test data.

Model Evaluation Metrics
Performance of each model was assessed using:

- R^2 Score, MAE, RMSE, MSE

Model Comparison

Model	R^2 Score	MAE	RMSE	MSE
RF (One-Hot)	0.75	1836.31	2837.26	8.05M
RF (Label Enc.)	0.75	1850.80	2813.08	7.91M
XGB (One-Hot)	0.72	1841.21	2964.96	8.79M
XGB (Label Enc.)	0.73	1849.40	2930.10	8.58M

In your project, testing both encoding strategies allowed you to observe the effect of feature space size on model training time, error metrics, and generalization.

Result: RandomForestRegressor with Label Encoding showed the best overall performance.

Explanation:
Label Encoding: provides a compact feature set for memory/performance efficiency.
One-Hot Encoding: Increases feature dimensions and may overfit with smaller datasets, Risk overfitting if data is not large enough.

6. Model Testing
In the final model testing phase, the trained RandomForestRegressor model was loaded and used to predict campaign costs based on new input data, confirming its effectiveness and readiness for deployment.

7. API Endpoint Details
Route: /predict (http://localhost:8000/predict)

Method: GET

Input Parameters:

1. brand_name,
2. product_name,
3. for_username,
4. overall_videos_viewcount_of_channel, subscribers_channel,
5. first_five_avg_video_of_channel,
6. engagement_rate_of_channel,
7. video_views,
8. video_duration (in HH:MM:SS),
9. category,
10. video_type

Output: Predicted video_cost (in float type)

📁 Project Structure

```
AI_Campaign_Cost_Prediction_Project/
├── app.py                  # FastAPI app with /predict endpoint
├── Dockerfile              # For containerizing the API
├── requirements.txt        # Python dependencies
├── RF_trained_model.pkl    # Trained RandomForestRegressor model
```

⚙️ Key Features

- Trained model loaded from RF_trained_model.pkl
- Input preprocessing (duration conversion, label encoding)
- Handles unseen category values using fallback encoding (-1)
- Uses joblib, pandas, FastAPI

▶️ Run the API

```
uvicorn app:app --reload
```