# **Content-Monetization-Modeler**

## **🎬 Project Overview: YouTube Ad Revenue Prediction**

## **🔹 Problem Statement**

### **As video creators and media companies increasingly rely on platforms like YouTube for income, predicting potential ad revenue becomes essential for business planning and content strategy. The task is to build a Linear Regression model that can accurately estimate YouTube ad revenue for individual videos based on various performance and contextual features, and implement the results in a simple Streamlit web application.**

### **🔹 Objective**

The goal of this project is to **predict YouTube ad revenue (in USD)** using both numerical and categorical features derived from video performance metrics and metadata. By building a regression-based machine learning model and deploying it as a web app, the project provides a tool for estimating ad revenue given engagement statistics.

### 

### **🔹 Dataset**

The dataset (from your notebook) includes the following key features:

* **Numerical Features**
  + views
  + likes
  + comments
  + watch\_time\_minutes
  + engagement\_rate *(derived or included in some versions)*
* **Categorical Features**
  + category (e.g., Entertainment, Education, Technology, Lifestyle)
  + device (e.g., Mobile, Desktop, Tablet, TV)
  + country (e.g., India, USA, UK, Germany, Canada)
* **Target Variable**
  + ad\_revenue\_usd → the revenue to be predicted.

### **🔹 Methodology**

1. **Data Preprocessing**
   * Numerical features are standardized (StandardScaler).
   * Categorical features are one-hot encoded (OneHotEncoder).
   * Missing values and irrelevant columns are handled in preprocessing.
2. **Modeling**
   * Initial experiments with **Support Vector Regression (SVR)** for capturing non-linear patterns.
   * Final deployed model uses **Linear Regression Pipeline** (for faster training and prediction).
   * Model evaluation metrics include **R², MAE, and RMSE**.
3. **Deployment**
   * A **Streamlit application** (model.py) was built.
   * Users can input YouTube video stats (views, likes, comments, etc.).
   * The app predicts expected **ad revenue** instantly.
   * Model pipeline is serialized using **Pickle** (linear\_regression\_pipeline.pkl).

### **🔹 Deliverables**

* **Jupyter Notebook (youtube.ipynb)**: Data analysis, feature engineering, model training, evaluation.
* **Streamlit Web App (model.py)**: User-friendly interface for predictions.
* **Pickled Model (linear\_regression\_pipeline.pkl)**: Trained machine learning pipeline.

### **🔹 Key Insights**

* Engagement metrics (likes, comments, engagement rate) are **positively correlated** with ad revenue.
* Device type and category also influence monetization patterns.
* Linear Regression provides interpretability and speed, while SVR was explored for accuracy but is slower on larger datasets.

## **⚙️ Technical Stack**

### **🔹 Programming Language**

* **Python 3.9+** → Main language for data processing, modeling, and app development.

### **🔹 Data Handling & Analysis**

* **Pandas** → For loading, cleaning, and manipulating structured data.
* **NumPy** → For numerical computations and efficient array operations.

### **🔹 Machine Learning**

* **scikit-learn** → Core library used for:  
  + Train-test split (train\_test\_split)
  + Preprocessing (StandardScaler, OneHotEncoder, ColumnTransformer)
  + Modeling (LinearRegression, SVR, LinearSVR)
  + Evaluation (r2\_score, mean\_absolute\_error, mean\_squared\_error)
* **Pickle** → For saving and loading the trained machine learning pipeline.

### **🔹 Model Development Environment**

* **Jupyter Notebook (youtube.ipynb)** → Used for exploration, feature engineering, training, evaluation, and visualization during development.

### **🔹 Deployment**

* **Streamlit** → To build the interactive web application (model.py).  
  + Provides input forms for video stats (views, likes, comments, etc.).
  + Calls the trained ML model pipeline for real-time prediction.
  + Displays predicted ad revenue in an intuitive UI.

### **🔹 Visualization (if used in Notebook)**

* **Matplotlib / Seaborn** → For exploratory data analysis, correlations, and performance plots.

### **🔹 Infrastructure**

* Local machine or cloud (can run on **Anaconda / VSCode / PyCharm**) for development.
* Deployment-ready via **Streamlit Cloud** or other hosting platforms (Heroku, AWS, GCP, etc.).

# **✨ Features, Future Model Ideas & Conclusion**

## **🔹 Features Used**

The project uses a mix of **numerical** and **categorical** features to predict YouTube ad revenue:

### **1. Numerical Features**

* **Views** → Number of times the video was watched.
* **Likes** → Viewer approval/positive engagement.
* **Comments** → Level of discussion and community activity.
* **Watch Time (minutes)** → Total duration viewers spent on the video.
* **Engagement Rate** → A derived metric (likes + comments) / views, reflecting interaction quality.

### **2. Categorical Features**

* **Category** → Type of content (e.g., Entertainment, Education, Lifestyle, etc.).
* **Device** → Access device (Mobile, Desktop, Tablet, TV).
* **Country** → Geographical location of the audience.

### **3. Target Variable**

* **Ad Revenue (USD)** → Continuous variable representing the earnings from YouTube ads.

## **🔹 Future Model Ideas**

To enhance prediction accuracy and make the model more robust:

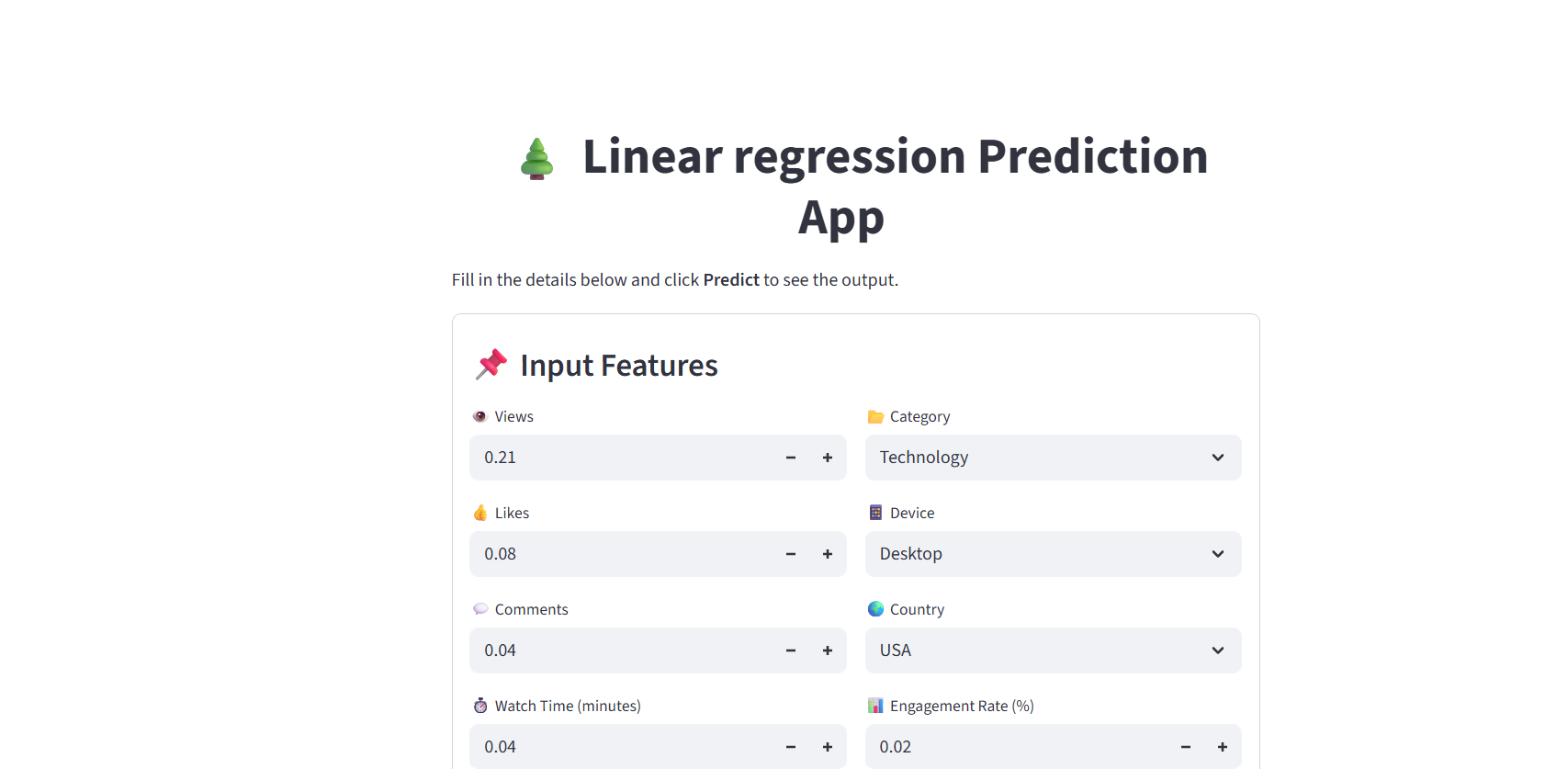
1. **Feature Expansion**
   * Include **subscriber count**, **video length (seconds)**, **upload frequency**, and **channel age**.
   * Add **CTR (Click-Through Rate)**, **impressions**, and **ad types**.
2. **Advanced Models**
   * Try **Tree-based models** (Random Forest, XGBoost, LightGBM) → better handling of non-linear patterns.
   * Use **Neural Networks (Deep Learning)** for capturing complex feature interactions.
   * Explore **Regularized Regression** (Ridge, Lasso, ElasticNet) for feature selection.
3. **Time-Series / Trend Analysis**
   * Incorporate historical data to predict revenue trends over time.
   * Add **seasonality effects** (festivals, holidays, trending events).
4. **Explainability**
   * Use **SHAP or LIME** to explain feature importance (which factors drive revenue most).
   * Provide creators with actionable insights, not just predictions.
5. **Deployment Improvements**
   * Scale the app with **Streamlit Cloud / AWS / GCP / Heroku** for public access.
   * Add **visual dashboards** with performance analytics and trend charts.
   * Implement **batch predictions** where creators can upload CSVs and get bulk predictions.

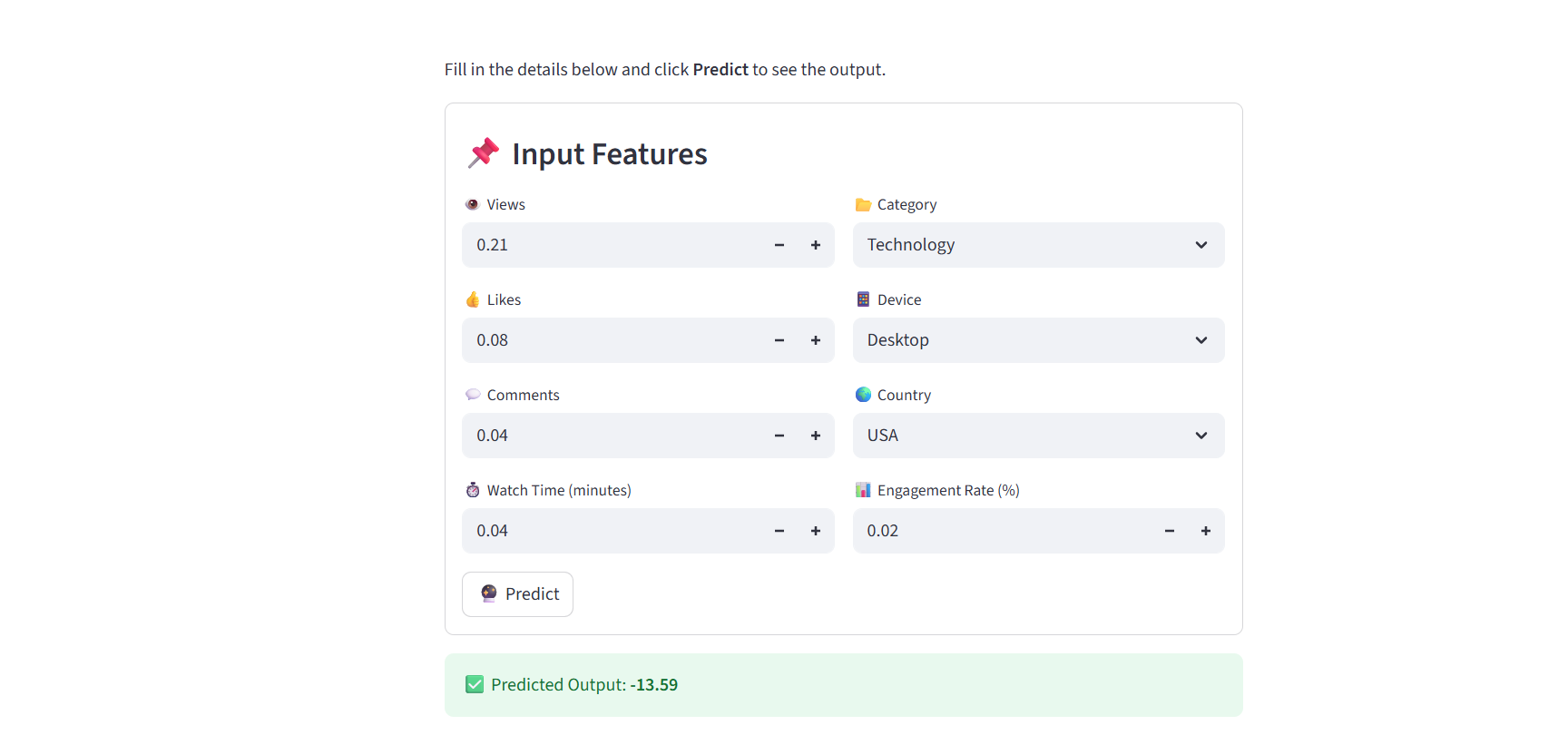
### **▶️ How to Run the Content-Monetization-Modeler Dashboard**

### **Launch the Streamlit Dashboard**

**Run the app using: bash streamlit run model.py 🌐 Open your browser and go to:** [**http://localhost:8503/**](http://localhost:8503/)

**Output:**

****

****

## 

## **🔹 Conclusion**

This project successfully demonstrates how **machine learning** can be leveraged to predict **YouTube ad revenue** using a combination of engagement and contextual features.

* We built a **Linear Regression pipeline** for fast, interpretable predictions.
* The model was deployed in a **Streamlit web app** for ease of use, making it accessible to video creators and businesses.
* Insights from the model highlight the importance of **engagement metrics (likes, comments, watch time)** and **audience demographics (country, device)** in influencing revenue.

With additional features, advanced models, and better deployment, this system can evolve into a powerful **decision-support tool** for YouTubers and media companies, helping them **optimize content strategy and maximize earnings**.