# **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.

### 

## **📊 Feature Explanation**

* **Views**: Number of views on the video
* **Likes**: Number of likes
* **Comments**: Number of comments
* **Watch\_Time\_Minutes**: Total watch time across all viewers
* **Subscribers**: Channel subscriber count at the time of upload
* **Year, Month, DayOfWeek**: Extracted from upload date
* **Category**: Video type (categorical: Education, Music, Tech, etc.)
* **Device**: Access device (Mobile, Desktop, Tablet, TV)
* **Country**: Audience location (IN, US, UK, DE, CA, etc.)
* **Ad\_Revenue (USD)**: Target variable, estimated YouTube ad earnings

### **🔹 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.).

# **✨Future Model Ideas & Conclusion**

## **🔹 Future Model Ideas**

## **🚀 Future Enhancements**

1. **Advanced ML Models**
   * Replace Linear Regression with **XGBoost, Random Forest, or Neural Networks** for better prediction accuracy.
   * Use **AutoML frameworks** to automatically tune hyperparameters.
2. **Time-based Predictions**
   * Add a **time-series forecasting model** (e.g., ARIMA, Prophet, LSTM) to predict **future revenue growth** over weeks/months.
   * Helps creators plan content release schedules.
3. **CPM (Cost Per Mille) Integration**
   * Include **country-specific CPM values** (since ad rates differ by region).
   * More realistic revenue prediction based on audience geography.
4. **Content Type Analysis**
   * Expand categories (Tech, Gaming, Music, Vlogs, etc.).
   * Predict revenue differently for each **content niche**.
5. **Ad Type & Monetization Options**
   * Add factors like **skippable ads, non-skippable ads, sponsorships, memberships, Super Chats**.
   * Model can predict not only ad revenue but **total monetization revenue**.
6. **Channel Growth Features**
   * Add features like **channel age, total subscribers, posting frequency, average video length**.
   * This gives a more **holistic view** of revenue potential.
7. **Visualization Dashboard**
   * Use **Plotly/Matplotlib** inside Streamlit to show:  
     + Trend of views vs. revenue
     + Engagement vs. revenue
     + Country/device-wise revenue breakdown
8. **Deployment & Accessibility**
   * Deploy on **Streamlit Cloud / AWS / Heroku / GCP** for global access.
   * Add **user authentication** so creators can save predictions and track their own channel.
9. **Recommendation System**
   * Suggest **optimal video length, upload time, or engagement strategies** to maximize revenue.
   * Example: "Videos uploaded on Friday evenings in Gaming category earn 20% more revenue."
10. **Real-Time YouTube API Integration**
    * Connect with **YouTube Data API** to automatically fetch views, likes, comments, etc.
    * One-click prediction without manual data entry.

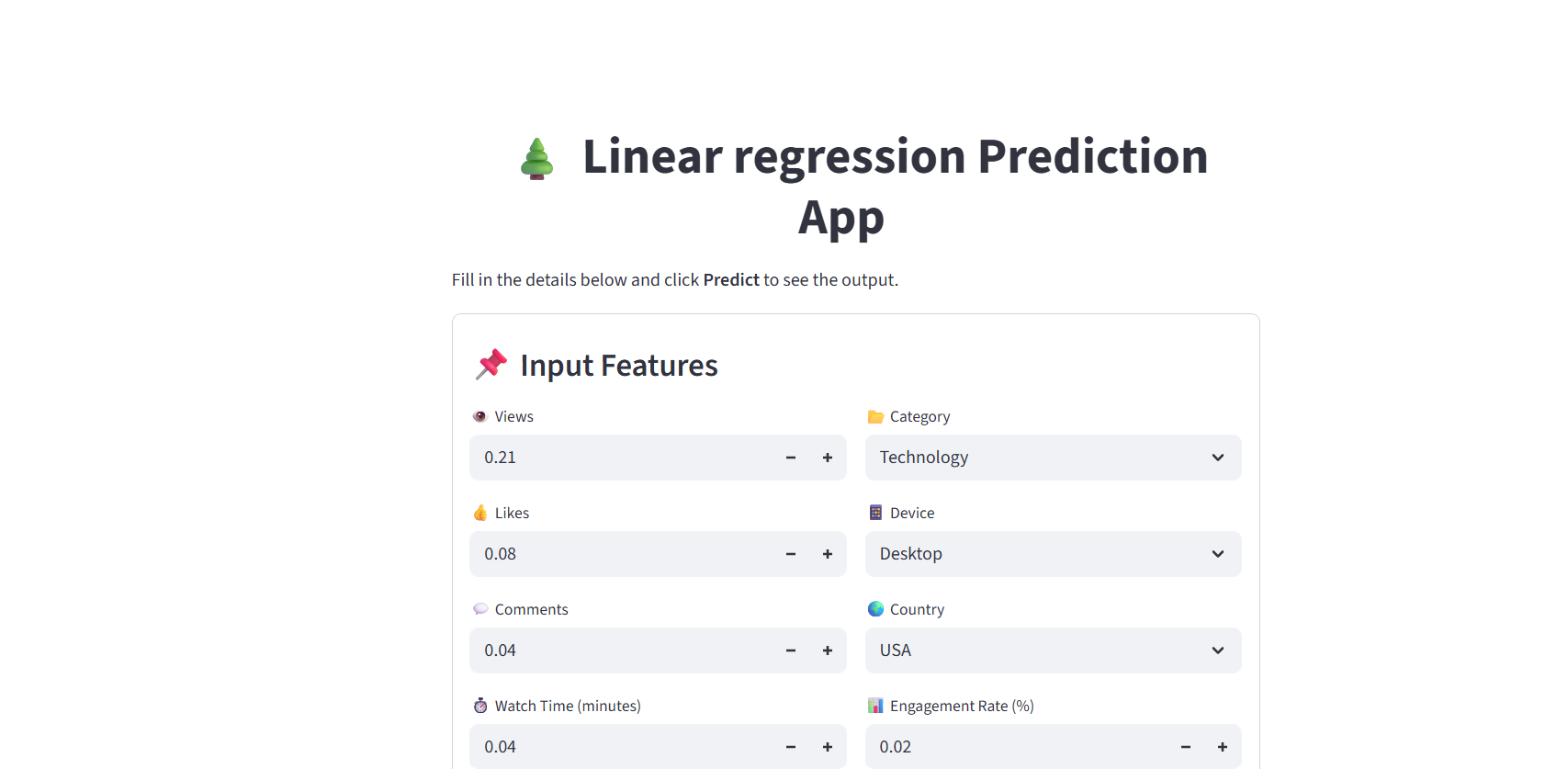
✨ These enhancements will make your p

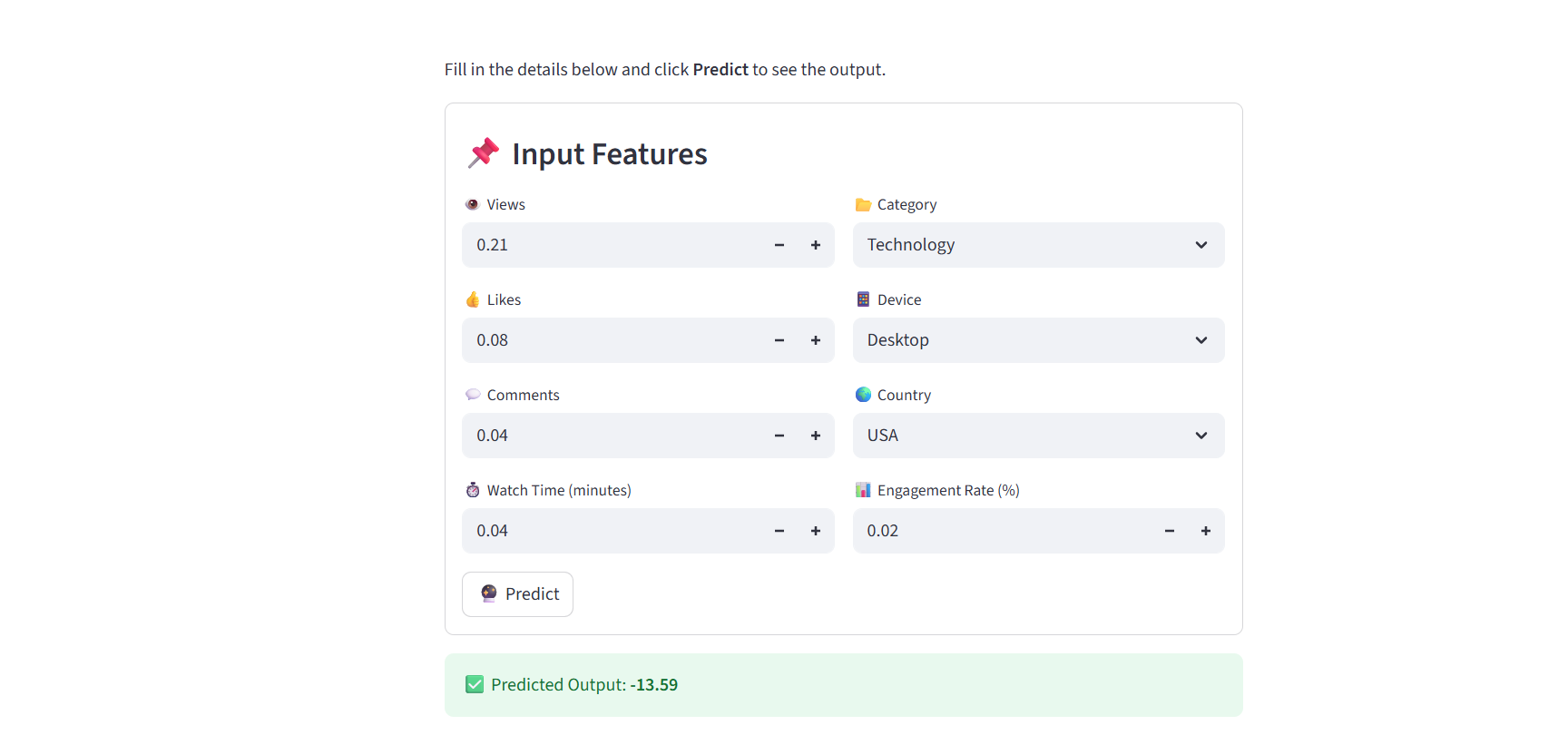
### **▶️ How to Run the Content-Monetization-Modeler Dashboard**

### **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**.