UNIFIED MENTOR

YOUR SKILL, SUCCESS & JOURNEY

SUBMITTED BY : ABISHEK S
UNDER THE GUIDANCE OF : UNIFIED MENTOR

Project Title :
Unlocking YouTube Channel Performance Secrets

Tools:Visual Studio code / jupyter notebook

Domain: Data Analyst

Project Difficulties level : Intermediate

UNIFIED MENTOR
YOUR SKILL, SUCCESS & JOURNEY

Dataset

This dataset provides detailed insights into YouTube video performance, audience engagement, revenue generation, and viewer behavior. It includes key metrics that help analyze how factors such as video duration, publish time, ad impressions, and viewer interactions impact monetization and channel growth.
The dataset covers:
Video Details: Duration, publish time, days since publish, and day of upload.
Revenue Metrics: Revenue per 1000 views (RPM), estimated revenue, ad impressions, and ad revenue sources like AdSense and DoubleClick.
Engagement Metrics: Views, likes, dislikes, shares, comments, average view duration, average view percentage, and thumbnail CTR.
Audience Data: New subscribers, unsubscribes, unique viewers, returning viewers, and new viewers.
Monetization Metrics: Monetized playbacks, playback-based CPM, YouTube Premium revenue, orders, and total sales volume.

Major Machine Learning Project: Unlocking YouTube Channel Performance Secrets Analysis

This project focuses on analyzing YouTube channel performance using Machine

Learning techniques. It uses key metrics such as views, subscribers, revenue, and engagement data to identify patterns, trends, and meaningful insights.
The project includes:
Exploratory Data Analysis (EDA) to understand the dataset
Data Visualization to highlight trends and relationships
Predictive Modeling to estimate revenue or subscriber growth
The main goal is to uncover actionable insights that help improve channel performance and make data-driven decisions.

Import Libraries :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

Load and Explore the Dataset

```
# Load the dataset
data = pd.read_csv("youtube_channel_data.csv")

# Display basic information about the dataset
print(data.info())

# Check for null values
print(data.isnull().sum())
```

```
# Preview the dataset
print(data.head())
```

Data Cleaning

```
# Drop rows with missing values (for simplicity)
data = data.dropna()

# Optional: Reset index after dropping rows
data.reset_index(drop=True, inplace=True)
```

Convert 'Video Duration' to Seconds

```
# Convert 'Video Duration' (ISO 8601 format) into seconds
import isodate

data['Video Duration'] = data['Video Duration'].apply(
   lambda x: isodate.parse_duration(x).total_seconds()
)
```

Exploratory Data Analysis (EDA)

```
# Pairplot to visualize relationships
sns.pairplot(
   data[['Revenue per 1000 Views (USD)',
        'Views',
        'Subscribers',
        'Estimated Revenue (USD)']]
)
```

```
plt.show()
```

Correlation Heatmap

```
plt.figure(figsize=(12, 8))

sns.heatmap(
    data.corr(numeric_only=True),
    annot=True,
    fmt=".2f",
    cmap="coolwarm"
)

plt.title("Correlation Heatmap")
plt.show()
```

Top Performers by Revenue

```
# Sort videos by Estimated Revenue (highest first) and get top 10
top_videos = data.sort_values(
    by='Estimated Revenue (USD)',
    ascending=False
).head(10)

# Display selected columns
print(top_videos[['ID', 'Estimated Revenue (USD)', 'Views', 'Subscribers']])
```

Feature Engineering

```
# Create Revenue per View
data['Revenue per View'] = data['Estimated Revenue (USD)'] /
data['Views']

# Create Engagement Rate (%)
data['Engagement Rate'] = (
    (data['Likes'] + data['Shares'] + data['Comments'])
    / data['Views']
) * 100

data['Revenue per View'] = np.where(
    data['Views'] != 0,
    data['Estimated Revenue (USD)'] / data['Views'],
    0
)

data['Engagement Rate'] = np.where(
    data['Views'] != 0,
    (data['Likes'] + data['Shares'] + data['Comments']) / data['Views'] *
100,
    0
)
```

Data Visualization

```
plt.figure(figsize=(10, 6))

sns.histplot(
    data['Estimated Revenue (USD)'],
    bins=50,
    kde=True,
    color='green'
)
```

```
plt.title("Revenue Distribution")
plt.xlabel("Revenue (USD)")
plt.ylabel("Frequency")

plt.show()
```

Revenue vs Views

```
plt.figure(figsize=(10, 6))

sns.scatterplot(
    x=data['Views'],
    y=data['Estimated Revenue (USD)']
)

plt.title("Revenue vs Views")
plt.xlabel("Views")
plt.ylabel("Estimated Revenue (USD)")

plt.show()
```

Predictive Model: Estimate Revenue

```
# Select features and target
features = [
    'Views',
    'Subscribers',
    'Likes',
    'Shares',
    'Comments',
    'Engagement Rate'
]

target = 'Estimated Revenue (USD)'

# Define X and y
X = data[features]
y = data[target]

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42
)
```

Train Random Forest Regressor

```
# Initialize the model
model = RandomForestRegressor(
    n_estimators=100,
    random_state=42
)

# Train the model
model.fit(X_train, y_train)

# Predict on test data
y_pred = model.predict(X_test)
```

Evaluate the Model

```
# Calculate performance metrics
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared: {r2:.2f}")
```

Insights and Recommendations

```
# Get feature importances from the trained model
importances = model.feature_importances_

# Create a DataFrame
feature_importance_df = pd.DataFrame({
    'Feature': features,
    'Importance': importances
})

# Sort by importance
feature_importance_df = feature_importance_df.sort_values(
    by='Importance',
    ascending=False
)

# Plot feature importance
plt.figure(figsize=(10, 6))

sns.barplot(
    x='Importance',
    y='Feature',
    data=feature_importance_df
)

plt.title("Feature Importance")
plt.xlabel("Importance Score")
plt.ylabel("Features")

plt.show()
```

Deployment and Presentation

```
import joblib

# Save the trained model
joblib.dump(model, "youtube_revenue_prediction_model.pkl")

print("Model saved successfully!")
```

Example:

```
# ==================================
# 1. Import Required Libraries
# ==================================

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import isodate
import joblib

from sklearn.model_selection import
train_test_split
from sklearn.ensemble import
RandomForestRegressor
from sklearn.metrics import
mean_squared_error, r2_score


# ==================================
# 2. Load Dataset
# ==================================

data = pd.read_csv("youtube_channel_data.csv")

print("Dataset Info:")
print(data.info())

print("\nMissing Values:")
print(data.isnull().sum())
```

```
# ==============================
# 3. Data Cleaning
# ==============================

# Drop missing values
data = data.dropna()
data.reset_index(drop=True, inplace=True)

# Convert Video Duration (ISO 8601 format like
PT5M30S) into seconds
if 'Video Duration' in data.columns:
    data['Video Duration'] = data['Video
Duration'].apply(
        lambda x:
isodate.parse_duration(x).total_seconds()
    )


# ==============================
# 4. Feature Engineering
# ==============================
# Revenue per View
data['Revenue per View'] = np.where(
    data['Views'] != 0,
    data['Estimated Revenue (USD)'] /
data['Views'],
    0
)

# Engagement Rate (%)
data['Engagement Rate'] = np.where(
    data['Views'] != 0,
    (data['Likes'] + data['Shares'] +
data['Comments']) / data['Views'] * 100,
    0
)


# ==============================
# 5. Exploratory Data Analysis
# ==============================

# Revenue Distribution
plt.figure(figsize=(8, 5))
sns.histplot(data['Estimated Revenue (USD)'],
bins=30, kde=True)
plt.title("Revenue Distribution")
plt.show()

# Correlation Heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(data.corr(numeric_only=True),
annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()


# ==================================
# 6. Prepare Data for Model
# ==================================

features = [
    'Views',
    'Subscribers',
    'Likes',
    'Shares',
    'Comments',
    'Engagement Rate'
]
```

```
target = 'Estimated Revenue (USD)'

X = data[features]
y = data[target]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)


# ==================================
# 7. Train Model
# ==================================

model =
RandomForestRegressor(n_estimators=100,
random_state=42)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)


# ==================================
# 8. Model Evaluation
# ==============================

mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("\nModel Performance:")
print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared Score: {r2:.2f}")


# ==============================
# 9. Feature Importance
# ==============================

importances = model.feature_importances_

feature_importance_df = pd.DataFrame({
    'Feature': features,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

plt.figure(figsize=(8, 5))
sns.barplot(x='Importance', y='Feature',
data=feature_importance_df)
plt.title("Feature Importance")
plt.show()


# ==================================
# 10. Save Model
# ==================================

joblib.dump(model,
"youtube_revenue_model.pkl")
print("Model Saved Successfully!")
```

Conclusion

This project analyzed YouTube channel performance using key metrics such as views, watch time, engagement rate, click-through rate (CTR), audience retention, upload timing, and revenue factors. The findings show that consistent content strategy, optimized video structure, and data-driven decisions significantly impact channel growth and monetization.

By leveraging analytics, creators can identify high-performing content patterns, understand audience behavior, and optimize future videos for better reach, engagement, and revenue generation.

Overall, YouTube success is not random — it is driven by strategy, consistency, and continuous performance monitoring.