Clickbait-Peak (CTR Prediction for ConnectSphere Digital)

Predicting which users are likely to click ads using real web analytics data and logistic regression.

See `documentation.md` for full modeling process, code snippets and EDA

Quickstart
1. Clone repo and install dependencies:
   pip install -r requirements.txt
2. Run EDA and modeling scripts.
3. Full results, plots, and performance details in `documentation.md`.

License
MIT

For questions: Nitish Sehgal / scholzisshit@gmail.com

EdiGlobe Minor Project Submission

Predictive Modeling for Click-Through Rate Optimization at ConnectSphere Digital

Problem Statement
ConnectSphere Digital manages many online ad campaigns but struggles with low ROI as ads are often shown to users unlikely to click. The goal is to create a data-driven model to identify and prioritize users most likely to engage, improving campaign effectiveness and client satisfaction.

Project Goal
Develop and evaluate a logistic regression model using historical user data—Age, Area Income, Daily Time Spent on Site, Daily Internet Usage, and demographics like Male—to classify users as 'likely to click' or 'unlikely to click'. Evaluate performance with accuracy, precision, and recall.

Business Objective
Enable ConnectSphere Digital to allocate ad budget more efficiently by targeting high-propensity users, thereby increasing overall Click-Through Rate (CTR) and boosting competitive advantage.

Link for Dataset
https://drive.google.com/file/d/1J5yqeokVKRrBZKrNxxnwXN_jGG4pUy6L/view

Dataset Overview
- 1000 samples with 10 features: usage patterns, demographics, ad topic, location, timestamp, and clicks (target).
- No missing values detected.
- Balanced target variable: 50% clicked, 50% not clicked.

Exploratory Data Analysis (EDA) Outputs and Inferences

- **Top 10 Cities and Countries by Frequency** show a diverse user base, no dominant location bias.
- **Unique Ad Topics** count = 1000, meaning each row has a unique topic, thus dropped for modeling to avoid noise.
- **Class balance** is perfect (50%-50%), aiding unbiased model training.

Model Performance

- **Accuracy: 95%** — Model predicts clicks correctly 95% of the time.
- **Precision: 95.5%** — When predicted "likely to click," correct 95.5% of the time, minimizing wasted ads.
- **Recall: 95.5%** — Identifies 95.5% of all actual clickers, reducing missed opportunities.

Confusion Matrix
[[89 5] # True Negatives, False Positives
 [5 106]] # False Negatives, True Positives

text

Libraries Used and Their Purposes

- **pandas**: Efficient data loading and manipulation.
- **matplotlib & seaborn**: Visualization for EDA (distributions, counts, boxplots).
- **scikit-learn**: Machine learning modeling, including data splitting, scaling, logistic regression, and evaluation metrics.

Code Snippets Generating Outputs

- Missing values check:
```
print(df.isnull().sum())
```

text

- Class distribution check:
```
print(df['Clicked on Ad'].value_counts(normalize=True))
```

text

- Top locations frequency:
```
print(df['City'].value_counts().head(10))
print(df['Country'].value_counts().head(10))
```

text

- Model training and evaluation:
```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train[num_features] = scaler.fit_transform(X_train[num_features])
X_test[num_features] = scaler.transform(X_test[num_features])

model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print('Accuracy:', accuracy_score(y_test, y_pred))
print('Precision:', precision_score(y_test, y_pred))
print('Recall:', recall_score(y_test, y_pred))
print('Confusion Matrix:\n', confusion_matrix(y_test, y_pred))
```

text

Final Thoughts
The model delivers robust predictions and can significantly improve targeted advertising at ConnectSphere Digital. Next steps might include error analysis on misclassifications, feature refinement, or testing advanced models. This repository contains full code, visualization, and evaluation to support deployment-ready insights.

This documentation was prepared for the EdiGlobe Minor Project submission, showcasing a real-world data science solution with a pinch of fun and full technical rigor.

Screenshots: