Dataset link: https://www.kaggle.com/datasets/marius2303/ad-click-prediction-dataset

```
!gdown --id 1SHoeE6p0RQADbTPHXI7G7be26XeFayco
```

```
/usr/local/lib/python3.10/dist-packages/gdown/__main__.py:132: FutureWarning: Option `--id` was deprecated in version 4.3.1 and will be
  warnings.warn(
Downloading...
From: https://drive.google.com/uc?id=1SHoeE6p0RQADbTPHXI7G7be26XeFayco
To: /content/ad_click_dataset.csv
100% 465k/465k [00:00<00:00, 44.1MB/s]
```

```
import pandas as pd
df = pd.read_csv('/content/ad_click_dataset.csv')
```

```
df.head()
```

	id	full_name	age	gender	device_type	ad_position	browsing_history	time_of_day	click
0	670	User670	22.0	NaN	Desktop	Top	Shopping	Afternoon	1
1	3044	User3044	NaN	Male	Desktop	Top	NaN	NaN	1
2	5912	User5912	41.0	Non-Binary	NaN	Side	Education	Night	1
3	5418	User5418	34.0	Male	NaN	NaN	Entertainment	Evening	1
4	9452	User9452	39.0	Non-Binary	NaN	NaN	Social Media	Morning	0

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 9 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   id                10000 non-null  int64
 1   full_name         10000 non-null  object
 2   age               5234 non-null   float64
 3   gender            5307 non-null   object
 4   device_type       8000 non-null   object
 5   ad_position       8000 non-null   object
 6   browsing_history  5218 non-null   object
 7   time_of_day       8000 non-null   object
 8   click             10000 non-null  int64
dtypes: float64(1), int64(2), object(6)
memory usage: 703.2+ KB
```

Turn 1

Here's how you can preprocess your dataset and use PyCaret for an AutoML approach to predict ad clicks:

Step 1: Install PyCaret If you haven't installed PyCaret, you can do so using pip

```
pip install pycaret
```

```
Collecting pycaret
  Downloading pycaret-3.3.2-py3-none-any.whl.metadata (17 kB)
Requirement already satisfied: ipython>=5.5.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (7.34.0)
Requirement already satisfied: ipywidgets>=7.6.5 in /usr/local/lib/python3.10/dist-packages (from pycaret) (7.7.1)
Requirement already satisfied: tqdm>=4.62.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (4.66.5)
Requirement already satisfied: numpy<1.27,>=1.21 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.26.4)
Requirement already satisfied: pandas<2.2.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (2.1.4)
Requirement already satisfied: jinja2>=3 in /usr/local/lib/python3.10/dist-packages (from pycaret) (3.1.4)
Collecting scipy<=1.11.4,>=1.6.1 (from pycaret)
  Downloading scipy-1.11.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (60 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.4/60.4 kB 966.3 kB/s eta 0:00:00
```

```
  Collecting joblib<1.4,>=1.2.0 (from pycaret)
    Downloading joblib-1.3.2-py3-none-any.whl.metadata (5.4 kB)
  Collecting scikit-learn>1.4.0 (from pycaret)
    Downloading scikit_learn-1.5.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
  Collecting pyod>=1.1.3 (from pycaret)
    Downloading pyod-2.0.2.tar.gz (165 kB)
       ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 165.8/165.8 kB 2.8 MB/s eta 0:00:00
    Preparing metadata (setup.py) ... done
  Requirement already satisfied: imbalanced-learn>=0.12.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (0.12.3)
  Collecting category-encoders>=2.4.0 (from pycaret)
    Downloading category_encoders-2.6.3-py2.py3-none-any.whl.metadata (8.0 kB)
  Requirement already satisfied: lightgbm>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (4.4.0)
  Requirement already satisfied: numba>=0.55.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (0.60.0)
  Requirement already satisfied: requests>=2.27.1 in /usr/local/lib/python3.10/dist-packages (from pycaret) (2.32.3)
  Requirement already satisfied: psutil>=5.9.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (5.9.5)
  Requirement already satisfied: markupsafe>=2.0.1 in /usr/local/lib/python3.10/dist-packages (from pycaret) (2.1.5)
  Requirement already satisfied: importlib-metadata>=4.12.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (8.4.0)
  Requirement already satisfied: nbformat>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (5.10.4)
  Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (from pycaret) (2.2.1)
  Collecting deprecation>=2.1.0 (from pycaret)
    Downloading deprecation-2.1.0-py2.py3-none-any.whl.metadata (4.6 kB)
  Collecting xxhash (from pycaret)
    Downloading xxhash-3.5.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
  Requirement already satisfied: matplotlib<3.8.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (3.7.1)
  Collecting scikit-plot>=0.3.7 (from pycaret)
    Downloading scikit_plot-0.3.7-py3-none-any.whl.metadata (7.1 kB)
  Requirement already satisfied: yellowbrick>=1.4 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.5)
  Requirement already satisfied: plotly>=5.14.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (5.15.0)
  Collecting kaleido>=0.2.1 (from pycaret)
    Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl.metadata (15 kB)
  Collecting schemdraw==0.15 (from pycaret)
    Downloading schemdraw-0.15-py3-none-any.whl.metadata (2.2 kB)
  Collecting plotly-resampler>=0.8.3.1 (from pycaret)
    Downloading plotly_resampler-0.10.0-py3-none-any.whl.metadata (13 kB)
  Requirement already satisfied: statsmodels>=0.12.1 in /usr/local/lib/python3.10/dist-packages (from pycaret) (0.14.2)
  Collecting sktime==0.26.0 (from pycaret)
    Downloading sktime-0.26.0-py3-none-any.whl.metadata (29 kB)
  Collecting tbats>=1.1.3 (from pycaret)
    Downloading tbats-1.1.3-py3-none-any.whl.metadata (3.8 kB)
  Collecting pmdarima>=2.0.4 (from pycaret)
    Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl.metadata (7.8 kB)
  Collecting wurlitzer (from pycaret)
    Downloading wurlitzer-3.1.1-py3-none-any.whl.metadata (2.5 kB)
  Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from sktime==0.26.0->pycaret) (24.1)
  Collecting scikit-base<0.8.0 (from sktime==0.26.0->pycaret)
    Downloading scikit_base-0.7.8-py3-none-any.whl.metadata (8.8 kB)
```

Step 2: Load and Preprocess the Data

```
import pandas as pd
import numpy as np
from pycaret.classification import *

# Load the dataset
df = pd.read_csv('/content/ad_click_dataset.csv')

# Preprocessing
# Drop unnecessary columns
df = df.drop(columns=['id', 'full_name'])

# Handle missing values
# For categorical variables, we'll use 'Unknown' as a placeholder
categorical_features = ['gender', 'device_type', 'ad_position', 'browsing_history', 'time_of_day']
for feature in categorical_features:
    df[feature] = df[feature].fillna('Unknown')

# For numerical variables, we'll use median
df['age'] = df['age'].fillna(df['age'].median())

# Encode categorical variables
# PyCaret will handle encoding, but for manual approach:
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for column in categorical_features:
    df[column] = le.fit_transform(df[column])

# Normalize numerical features if needed (PyCaret will handle this, but for manual approach):
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
```

```
df['age'] = scaler.fit_transform(df[['age']])

# Setup PyCaret environment
clf_setup = setup(data=df, target='click', session_id=123,
                  normalize=True,
                  html=False, # Use html=False to avoid interactive prompts
                  categorical_features=categorical_features)
```

	Description	Value
0	Session id	123
1	Target	click
2	Target type	Binary
3	Original data shape	(10000, 7)
4	Transformed data shape	(10000, 25)
5	Transformed train set shape	(7000, 25)
6	Transformed test set shape	(3000, 25)
7	Numeric features	1
8	Categorical features	5
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Maximum one-hot encoding	25
14	Encoding method	None
15	Normalize	True
16	Normalize method	zscore
17	Fold Generator	StratifiedKFold
18	Fold Number	10
19	CPU Jobs	-1
20	Use GPU	False
21	Log Experiment	False
22	Experiment Name	clf-default-name
23	USI	6ba8

Step 3: Compare Models

```
# Compare models
best_model = compare_models()
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.7346	0.7659	0.9147	0.7391	0.8175	0.3502	0.3785	0.353
lightgbm	Light Gradient Boosting Machine	0.7134	0.7233	0.9400	0.7118	0.8100	0.2696	0.3192	1.988
dt	Decision Tree Classifier	0.7089	0.6922	0.8048	0.7611	0.7822	0.3439	0.3454	0.197
rf	Random Forest Classifier	0.7017	0.7252	0.8464	0.7350	0.7867	0.2988	0.3073	0.866
et	Extra Trees Classifier	0.6801	0.7216	0.7982	0.7333	0.7643	0.2694	0.2720	1.153
gbc	Gradient Boosting Classifier	0.6769	0.6387	0.9789	0.6728	0.7975	0.1177	0.1995	0.697
ada	Ada Boost Classifier	0.6610	0.5800	0.9824	0.6610	0.7902	0.0587	0.1235	0.721
lr	Logistic Regression	0.6503	0.5569	0.9967	0.6508	0.7875	0.0047	0.0303	1.094
ridge	Ridge Classifier	0.6501	0.5568	0.9976	0.6506	0.7875	0.0032	0.0277	0.431
dummy	Dummy Classifier	0.6500	0.5000	1.0000	0.6500	0.7879	0.0000	0.0000	0.172
lda	Linear Discriminant Analysis	0.6499	0.5568	0.9960	0.6507	0.7871	0.0039	0.0250	0.246
knn	K Neighbors Classifier	0.6470	0.6120	0.8310	0.6897	0.7537	0.1493	0.1578	0.254
svm	SVM - Linear Kernel	0.6401	0.5148	0.9653	0.6504	0.7769	0.0021	0.0026	0.250
nb	Naive Bayes	0.6296	0.5557	0.8635	0.6658	0.7518	0.0672	0.0773	0.180
qda	Quadratic Discriminant Analysis	0.5307	0.5269	0.5587	0.6656	0.6020	0.0354	0.0363	0.207

Step 4: Finalize and Evaluate the Model

```
# Finalize the model
final_best = finalize_model(best_model)
```

```
# Evaluate the model
evaluate_model(final_best)
```

Plot Type:

Pipeline Plot | Hyperparameters | AUC | Confusion Matrix | Threshold | Precision Recall
Prediction Error | Class Report | Feature Selection | Learning Curve | Manifold Learning | Calibration Curve
Validation Curve | Dimensions | Feature Importance | Feature Importance… | Decision Boundary | Lift Chart
Gain Chart | Decision Tree | KS Statistic Plot

Step 5: Save and Use the Model

```
# Save the model
save_model(final_best, 'ad_click_prediction_model')
```

```
Transformation Pipeline and Model Successfully Saved
(Pipeline(memory=Memory(location=None),
          steps=[('numerical_imputer',
                  TransformerWrapper(exclude=None, include=['age'],
                                     transformer=SimpleImputer(add_indicator=False,
                                                               copy=True,
                                                               fill_value=None,
                                                               keep_empty_features=False,
                                                               missing_values=nan,
                                                               strategy='mean'))),
                 ('categorical_imputer',
                  TransformerWrapper(exclude=None,
                                     include=['gender', 'device_type',
                                              'ad_position'...
                                  importance_type=None,
                                  interaction_constraints=None, learning_rate=None,
                                  max_bin=None, max_cat_threshold=None,
                                  max_cat_to_onehot=None, max_delta_step=None,
                                  max_depth=None, max_leaves=None,
                                  min_child_weight=None, missing=nan,
                                  monotone_constraints=None, multi_strategy=None,
                                  n_estimators=None, n_jobs=-1,
                                  num_parallel_tree=None,
                                  objective='binary:logistic', ...))],
          verbose=False),
 'ad_click_prediction_model.pkl')
```

Start coding or generate with AI.