Ml

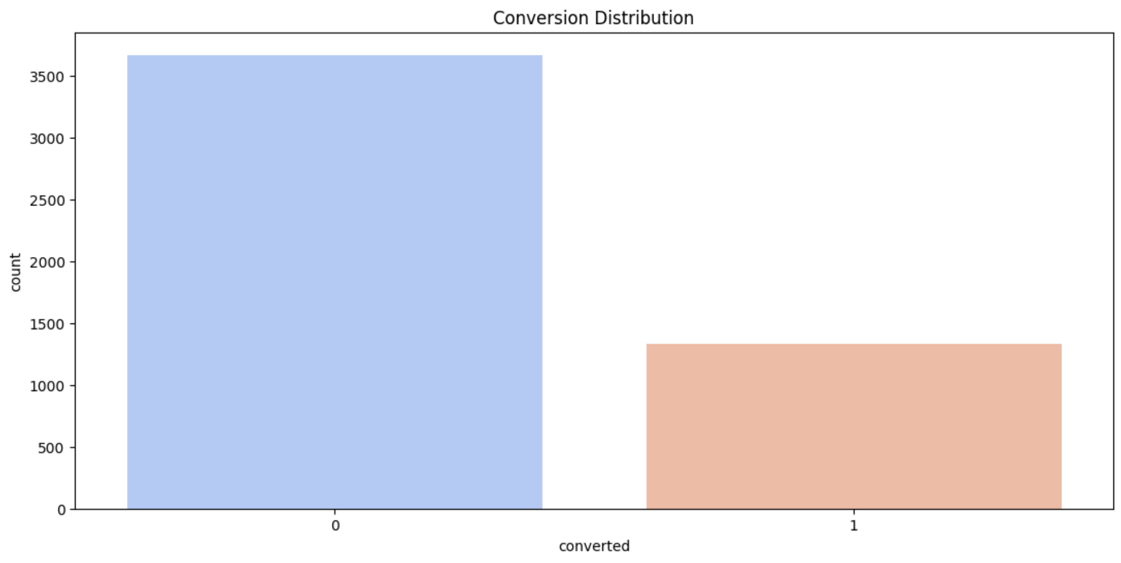
March 12, 2025

[14]:
```
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.preprocessing import StandardScaler, OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, roc_auc_score, classification_report

# Load dataset
df = pd.read_csv(r"C:\Users\Joshua.Mahada\Downloads\customer_conversion_data.
 ↪csv")

# Display basic info
display(df.head())
print(df.info())

# Visualise conversion distribution
plt.figure(figsize=(13,6))
sns.countplot(x='converted', data=df, palette='coolwarm')
plt.title("Conversion Distribution")
plt.show()

# Encoding categorical features
categorical_features = ["gender", "device_type", "ad_channel"]
encoder = OneHotEncoder(drop='first', sparse=False)
categorical_encoded = pd.DataFrame(encoder.
 ↪fit_transform(df[categorical_features]))
categorical_encoded.columns = encoder.get_feature_names_out()

# Standardising numerical features
numerical_features = ["age", "time_spent", "num_impressions", "num_clicks"]
scaler = StandardScaler()
numerical_scaled = pd.DataFrame(scaler.fit_transform(df[numerical_features]),␣
 ↪columns=numerical_features)

# Combine features
```

```python
X = pd.concat([numerical_scaled, categorical_encoded], axis=1)
y = df["converted"]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Train model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Evaluate model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("AUC-ROC:", roc_auc_score(y_test, y_pred))
print(classification_report(y_test, y_pred))

# Feature importance
feature_importances = pd.Series(model.feature_importances_, index=X.columns)
plt.figure(figsize=(13,6))
feature_importances.nlargest(10).plot(kind='barh')
plt.title("Top 10 Feature Importances")
plt.show()

print("Exploratory Data Analysis and Model Evaluation Complete!")
```

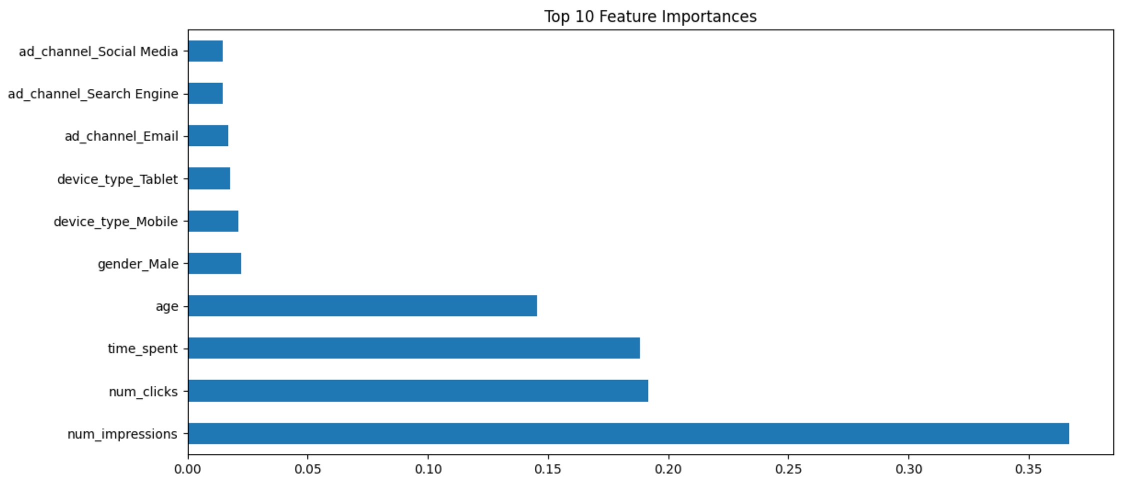
```
   user_id  age  gender device_type    ad_channel  time_spent  \
0        1   56    Male      Mobile         Email   78.364084
1        2   46  Female      Mobile       Display  132.356481
2        3   32    Male     Desktop       Display  192.044900
3        4   60    Male      Mobile  Social Media   80.810852
4        5   25    Male      Mobile       Display   20.394592

   num_impressions  num_clicks  converted
0               17           6          1
1               22           7          0
2               39           8          1
3               41           6          0
4               48           7          0

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   user_id          5000 non-null   int64
 1   age              5000 non-null   int64
 2   gender           5000 non-null   object
 3   device_type      5000 non-null   object
 4   ad_channel       5000 non-null   object
 5   time_spent       5000 non-null   float64
 6   num_impressions  5000 non-null   int64
 7   num_clicks       5000 non-null   int64
 8   converted        5000 non-null   int64
dtypes: float64(1), int64(5), object(3)
memory usage: 351.7+ KB
None
```

```
C:\Users\Joshua.Mahada\AppData\Local\Temp\ipykernel_17084\1394045602.py:19:
FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.countplot(x='converted', data=df, palette='coolwarm')
```



```
C:\Users\Joshua.Mahada\AppData\Local\Programs\Python\Python312\Lib\site-
packages\sklearn\preprocessing\_encoders.py:975: FutureWarning: `sparse` was
renamed to `sparse_output` in version 1.2 and will be removed in 1.4.
`sparse_output` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

```
Accuracy: 0.822
AUC-ROC: 0.7356150793650793
              precision    recall  f1-score   support

           0       0.84      0.93      0.88       720
           1       0.76      0.54      0.63       280

    accuracy                           0.82      1000
   macro avg       0.80      0.74      0.76      1000
weighted avg       0.82      0.82      0.81      1000
```



```
Exploratory Data Analysis and Model Evaluation Complete!
```