```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder

# Reload the dataset
df = pd.read_csv("customer_purchase_prediction.csv")
df.head()

   Age  Annual_Income  Gender Marital_Status  Browsing_Time  \
0   56          21920    Male        Married       4.589678
1   69         126121  Female       Divorced      14.026686
2   46          97219  Female        Married       9.317701
3   32          96872  Female         Single      12.214115
4   60         101132  Female        Married       6.560902

   Previous_Purchases  Clicked_Ad  Customer_Rating  Purchase
0                  17           0         2.070361         0
1                   5           1         1.610790         0
2                   8           1         4.073605         1
3                   5           0         2.744720         0
4                  17           0         1.708211         0

# Encode categorical variables
label_encoder = LabelEncoder()
df["Gender"] = label_encoder.fit_transform(df["Gender"])  # Male: 1,
Female: 0
df["Marital_Status"] =
label_encoder.fit_transform(df["Marital_Status"])

# Define features (X) and target (y)
X = df.drop(columns=["Purchase"])
y = df["Purchase"]

# 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)

# Initialize and train the decision tree classifier
classifier = DecisionTreeClassifier(random_state=42)
classifier.fit(X_train, y_train)

# Make predictions
y_pred = classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
```

```
accuracy, classification_rep

(1.0,
 '              precision    recall  f1-score   support\n\n
0       1.00      1.00      1.00       920\n           1       1.00
1.00      1.00        80\n\n    accuracy
1.00      1000\n   macro avg       1.00      1.00      1.00      1000\
nweighted avg       1.00      1.00      1.00      1000\n')

from sklearn.model_selection import cross_val_score

# Perform cross-validation to evaluate the model's performance on
unseen data
cv_scores = cross_val_score(classifier, X, y, cv=5,
scoring='accuracy')

# Calculate mean and standard deviation of cross-validation scores
cv_mean = np.mean(cv_scores)
cv_std = np.std(cv_scores)

cv_mean, cv_std, cv_scores

(0.9996, 0.0004898979485566361, array([1.   , 1.   , 0.999, 0.999, 1.
]))
```

USING UNSEEN DATA

```
# Generate unseen data with similar characteristics as the original
dataset
n_unseen = 5000  # Number of rows for unseen data

unseen_data = {
    "Age": np.random.randint(18, 70, size=n_unseen),
    "Annual_Income": np.random.randint(20000, 150000, size=n_unseen),
    "Gender": np.random.choice(["Male", "Female"], size=n_unseen),
    "Marital_Status": np.random.choice(["Single", "Married",
"Divorced", "Widowed"], size=n_unseen),
    "Browsing_Time": np.random.uniform(1, 15, size=n_unseen),
    "Previous_Purchases": np.random.randint(0, 20, size=n_unseen),
    "Clicked_Ad": np.random.choice([0, 1], size=n_unseen, p=[0.7,
0.3]),
    "Customer_Rating": np.random.uniform(1, 5, size=n_unseen),
}

# Add the target variable "Purchase" using similar logic
unseen_data["Purchase"] = np.where(
    (unseen_data["Browsing_Time"] > 5) &
    (unseen_data["Annual_Income"] > 50000) &
    (unseen_data["Previous_Purchases"] > 2) &
    (unseen_data["Customer_Rating"] > 3) &
    (unseen_data["Clicked_Ad"] == 1),
    1,
    0
)

# Convert to a DataFrame
unseen_df = pd.DataFrame(unseen_data)

# Save unseen data to CSV
unseen_df.to_csv('unseen_df.csv', index=False)
unseen_df
```

```
      Age  Annual_Income  Gender Marital_Status  Browsing_Time  \
0      19          29964  Female         Single       3.740645
1      22         130043  Female         Single      14.909457
2      54          44838  Female         Single       9.805637
3      41         102474  Female        Married       2.083550
4      22          30597  Female        Widowed      11.168576
...   ...            ...     ...            ...            ...
4995   41          87046    Male       Divorced      14.117459
4996   33         120775    Male        Married       9.392444
4997   63          98042    Male        Married       5.628072
4998   39          87968  Female         Single       9.236977
4999   48          95977  Female        Married      11.941578

      Previous_Purchases  Clicked_Ad  Customer_Rating  Purchase
0                      0           0         1.429187         0
1                     17           0         3.285423         0
2                      9           0         4.362337         0
3                     17           0         1.598685         0
4                     15           0         2.754389         0
...                  ...         ...              ...       ...
4995                   7           0         3.181555         0
4996                  15           1         2.078366         0
4997                   7           1         3.615173         1
4998                   5           1         4.616142         1
4999                  19           0         1.781354         0

[5000 rows x 9 columns]
```

```
# Prepare unseen data
unseen_df = pd.read_csv("unseen_df.csv")

# Encode categorical variables in the unseen data
label_encoder = LabelEncoder()
unseen_df["Gender"] = label_encoder.fit_transform(df["Gender"])  #
Male: 1, Female: 0
unseen_df["Marital_Status"] =
label_encoder.fit_transform(df["Marital_Status"])
```

```python
# Define features (X_unseen) and target (y_unseen) for unseen data
X_unseen = unseen_df.drop(columns=["Purchase"])
y_unseen = unseen_df["Purchase"]

# Make predictions on unseen data
y_unseen_pred = classifier.predict(X_unseen)

# Evaluate the model's performance on unseen data
unseen_accuracy = accuracy_score(y_unseen, y_unseen_pred)
unseen_classification_report = classification_report(y_unseen,
y_unseen_pred)

unseen_accuracy, unseen_classification_report
```

```
(0.9992,
 '              precision    recall  f1-score   support\n\n
0       1.00      1.00      1.00      4665\n           1       1.00
0.99      0.99       335\n\n    accuracy
1.00      5000\n   macro avg       1.00      0.99      1.00      5000\
nweighted avg       1.00      1.00      1.00      5000\n')
```

The decision tree classifier performed exceptionally well on the dataset, achieving an accuracy of 100% on the test set