E-COMMERCE BOOK PURCHASE PREDICTION

BRIEF

In this project, I scraped data from an open-source e-commerce website and created a custom dataset of books.

After analyzing key patterns using exploratory data analysis (EDA), I simulated realistic purchase behavior based on price and rating, and trained a machine learning model to predict whether a user will purchase a book.

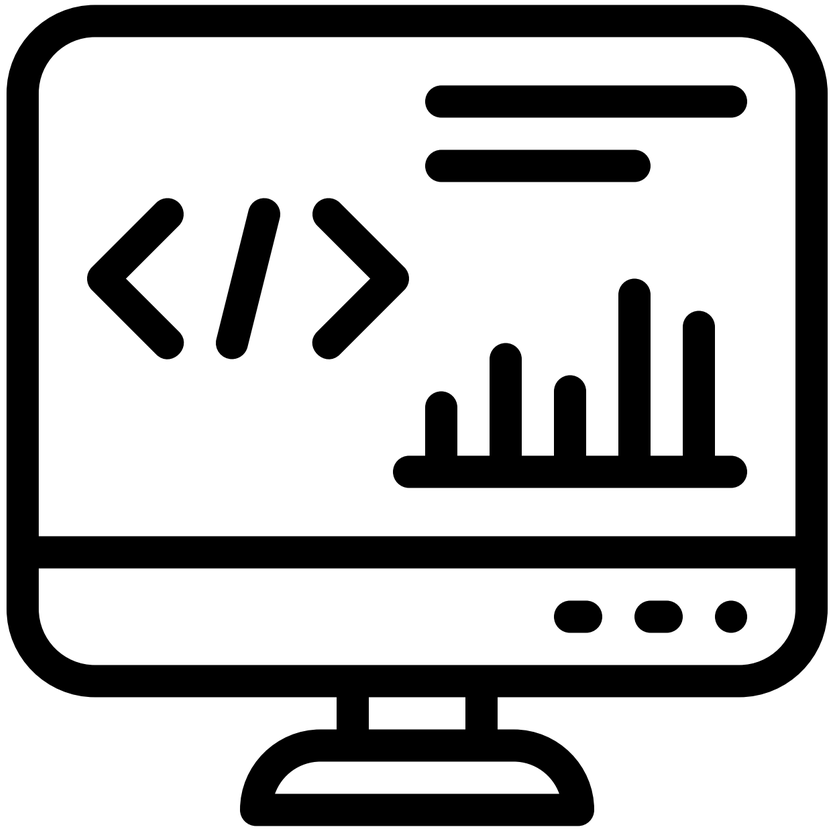
The final model takes just Price and Rating as input and predicts the purchase decision with over 94% accuracy. The entire workflow – from data scraping to model deployment – is showcased in a deployed Streamlit app.

LIBRARIES USED FOR SCRAPPING, EDA, AND VISUALIZATION

- REQUESTS
- BEAUTIFULSOUP
- PANDAS
- NUMPY
- MATPLOTLIB
- SEABORN

LIBRARIES USED FOR MACHINE LEARNING, APP INTERFACE

- SCIKIT-LEARN
- JOBLIB
- STREAMLIT



Scrapping, EDA, & Visualization

This Is the website I have scrapped the data from

Books to Scrape We love being scraped!

Home / All products

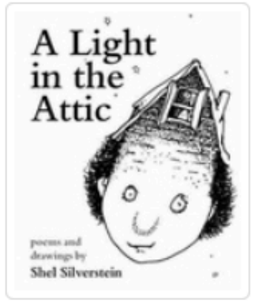
Books

- Travel
- Mystery
- Historical Fiction
- Sequential Art
- Classics
- Philosophy
- Romance
- Womens Fiction
- Fiction
- Childrens
- Religion
- Nonfiction
- Music
- Default
- Science Fiction
- Sports and Games
- Add a comment
- Fantasy
- New Adult
- Young Adult
- Science
- Poetry
- Paranormal

All products

1000 results - showing **1** to **20**.

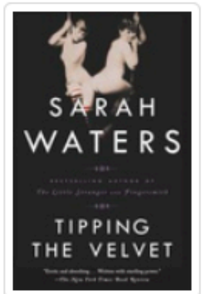
Warning! This is a demo website for web scraping purposes. Prices and ratings here were randomly assigned and have no real meaning.



A Light in the ...

£51.77

✔ In stock



Tipping the Velvet

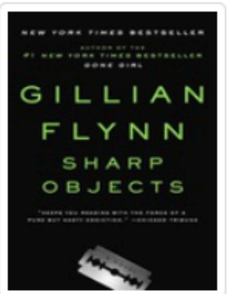
£53.74

✔ In stock

Soumission

£50.10

✔ In stock



Sharp Objects

£47.82

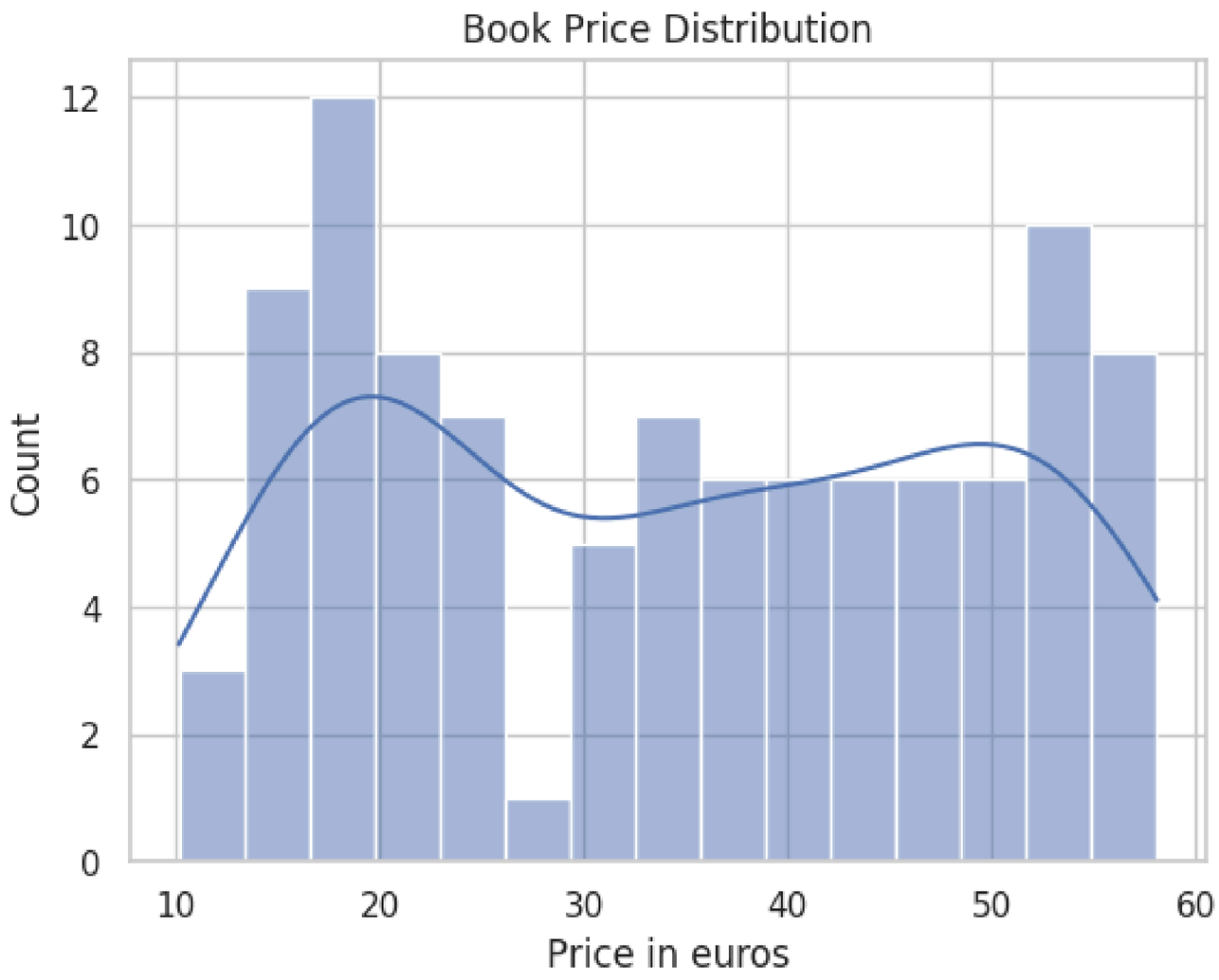
✔ In stock

Visual on the scarped data

Title	Rating	Price	Avalability
A Light in the Attic	Three	51.77	In stock
Tipping the Velvet	One	53.74	In stock
Soumission	One	50.10	In stock
Sharp Objects	Four	47.82	In stock
Sapiens: A Brief History of Humankind	Five	54.23	In stock

Modified Scrapped Data

Title	Rating	Price	Avalability	expensive	purchased	quantity_purchased	Gender	added_to_cart
A Light in the Attic	3	51.77	In stock	Expensive	0	0	Male	5
Tipping the Velvet	1	53.74	In stock	Expensive	0	0	Male	4
Soumission	1	50.1	In stock	Expensive	0	0	Male	1
Sharp Objects	4	47.82	In stock	Expensive	0	0	Female	3
Sapiens: A Brief History of Humankind	5	54.23	In stock	Expensive	0	0	Male	5
The Requiem Red	1	22.65	In stock	Not Expensive	0	0	Female	5
The Black Maria	1	52.15	In stock	Expensive	0	0	Male	0
Starving Hearts (Triangular Trade Trilogy, #1)	2	13.99	In stock	Not Expensive	0	0	Male	1
Scott Pilgrim's Precious Little Life (Scott Pilgrim #1	5	52.29	In stock	Expensive	0	0	Male	0
Our Band Could Be Your Life: Scenes from the Am	3	57.25	In stock	Expensive	0	0	Female	3
Olio	1	23.88	In stock	Not Expensive	0	0	Male	5
Mesaerion: The Best Science Fiction Stories 1800-	1	37.59	In stock	Expensive	0	0	Female	5
Libertarianism for Beginners	2	51.33	In stock	Expensive	0	0	Female	4
It's Only the Himalayas	2	45.17	In stock	Expensive	0	0	Female	4
In Her Wake	1	12.84	In stock	Not Expensive	0	0	Male	5
How Music Works	2	37.32	In stock	Expensive	0	0	Male	5
Birdsong: A Story in Pictures	3	54.64	In stock	Expensive	0	0	Male	5

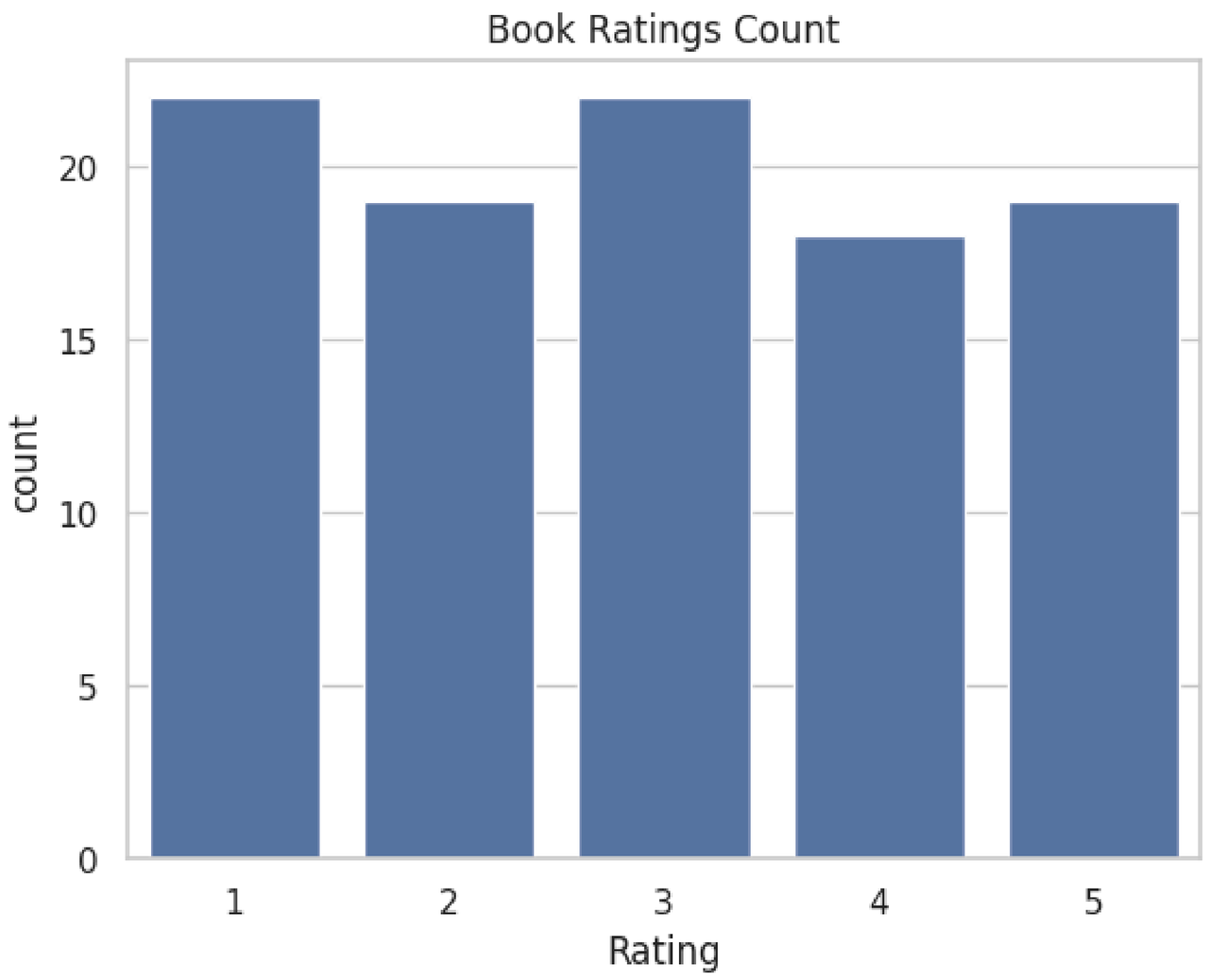


BOOK PRICE DISTRIBUTION:

THIS VISUAL IS USED TO CHECK HOW MANY BOOKS WE HAVE AT A CERTAIN PRICE RANGE.
A SHARP INCEASE IS NOTICED IN THE 10-20 RANGE FOLLOWED BY A SIMILAR DECEND IN 20-30 RANGE. IMPLYING THERE IS A HIGH VARIENCE IN THE PRICE TO QUANTITY RATIO.

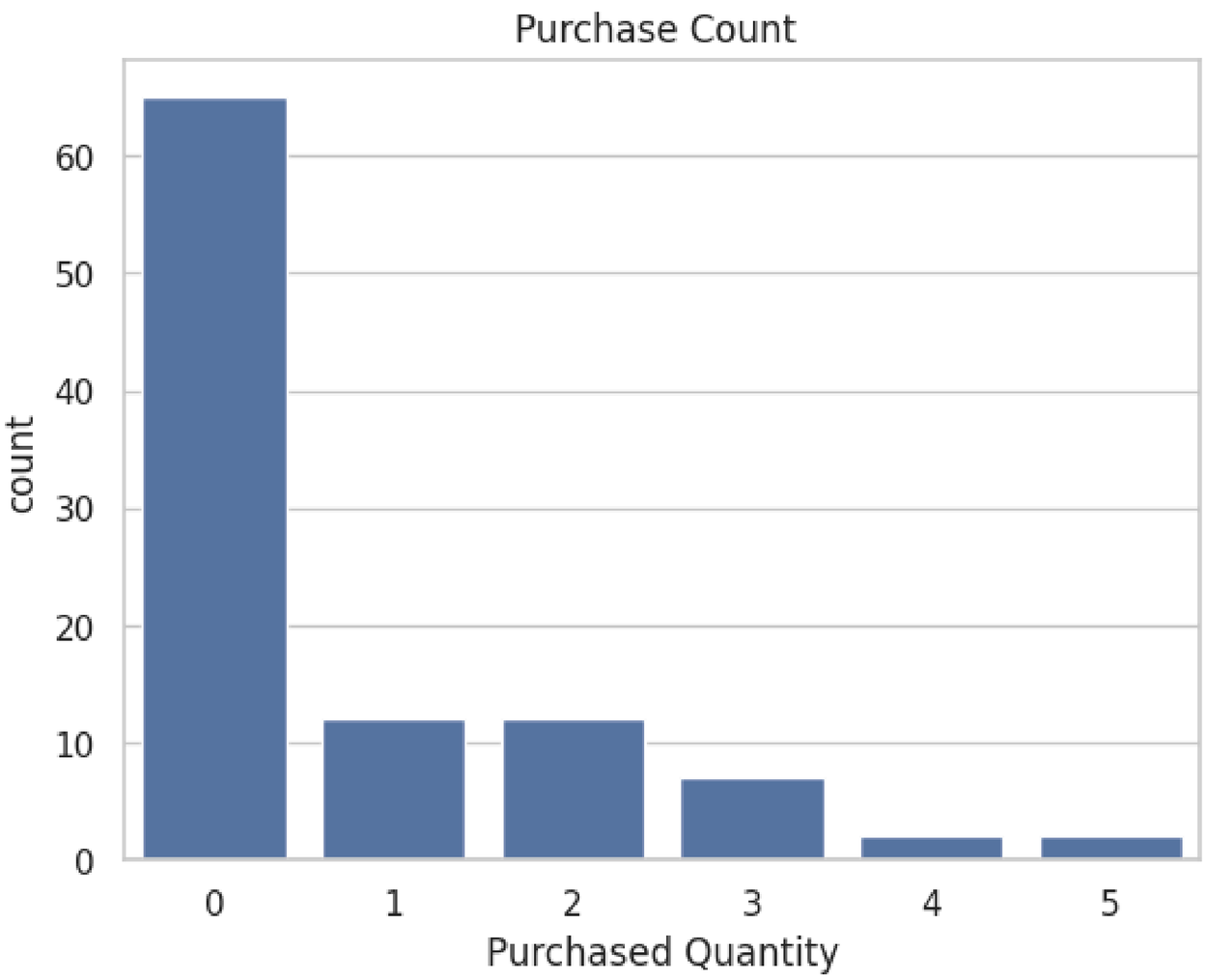
CODE USED:

```
sns.histplot(df['Price'], kde=True, bins=15)
plt.title('Book Price Distribution')
plt.xlabel('Price in euros')
```



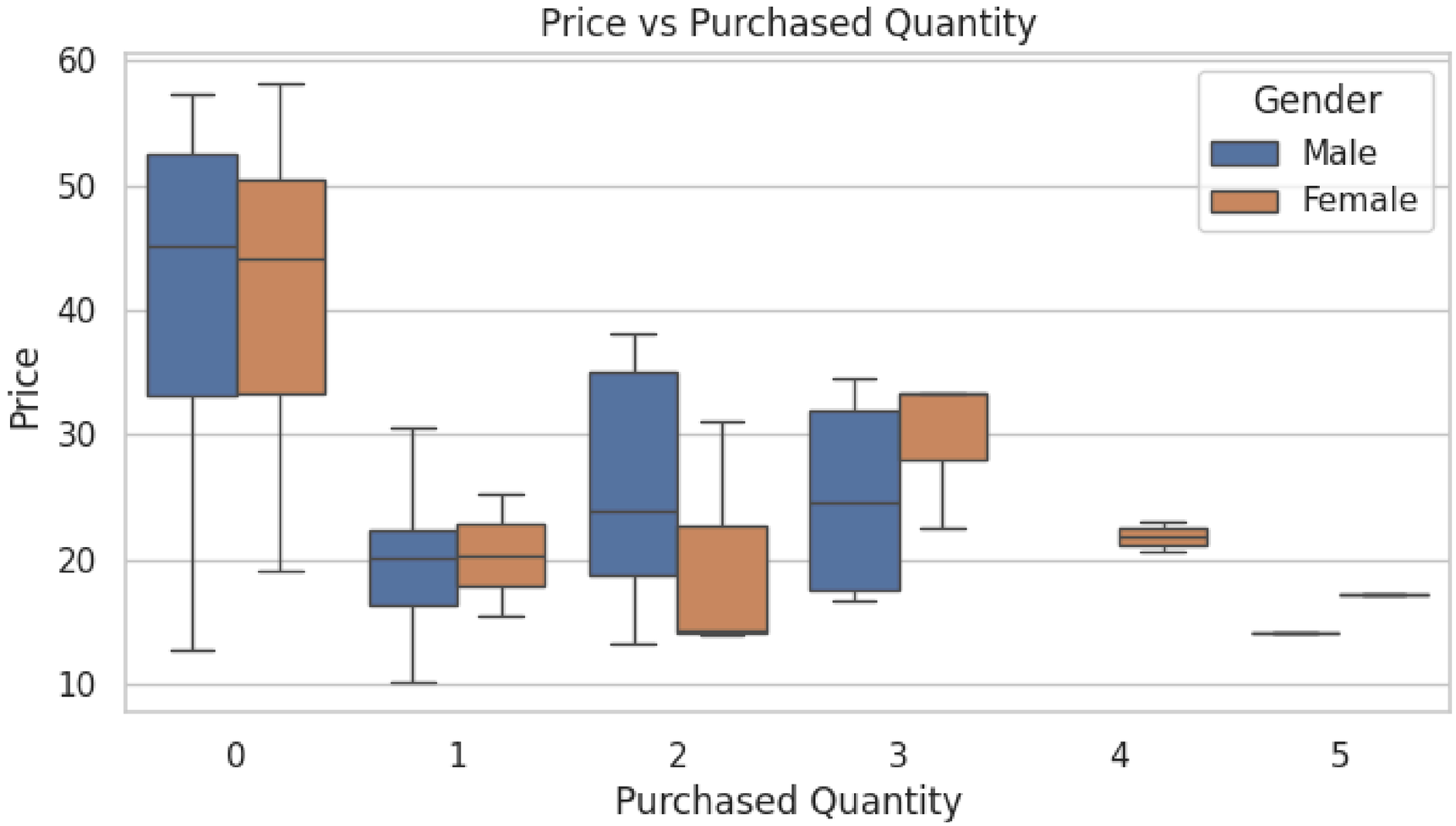
CODE USED:

```
sns.countplot(df,x='Rating')
plt.title("Book Ratings Count")
```



CODE USED:

```
sns.countplot(x='quantity_purchased',data=df)
plt.title("Purchase Count ")
plt.xlabel("Purchased Quantity")
```

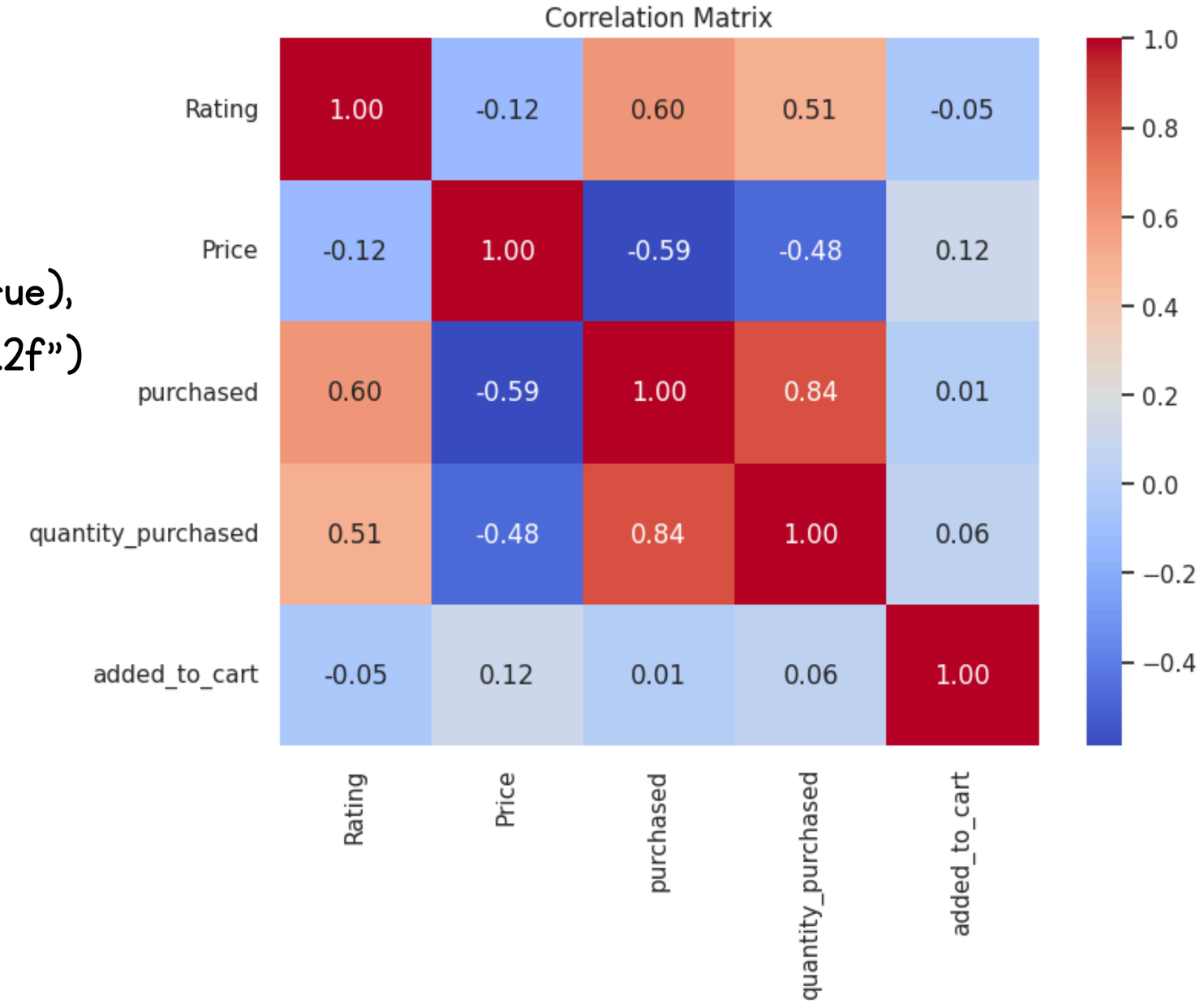


CODE USED:

```
sns.countplot(x='quantity_purchased',data=df)
plt.title("Purchase Count ")
plt.xlabel("Purchased Quantity")
```

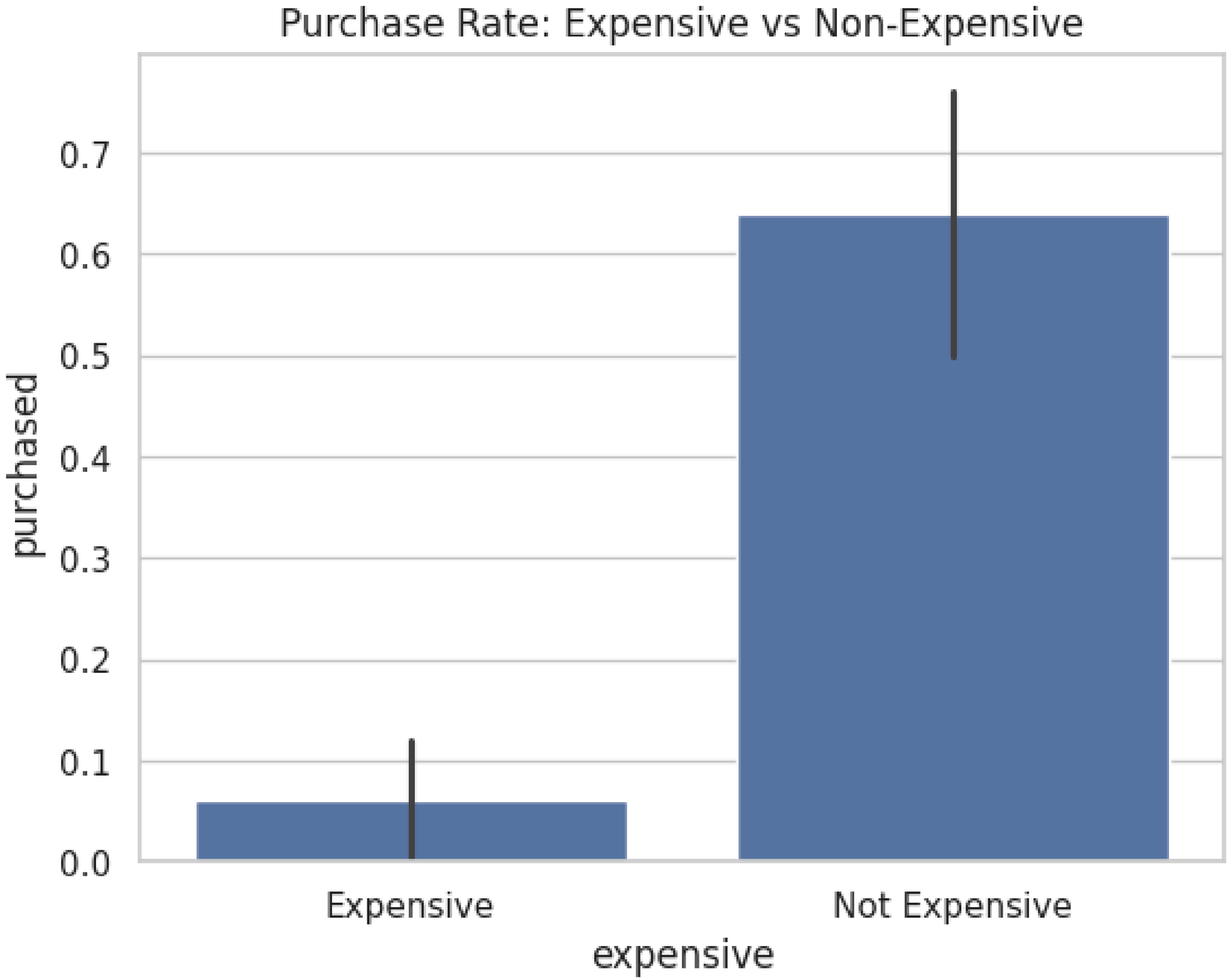
CODE USED:

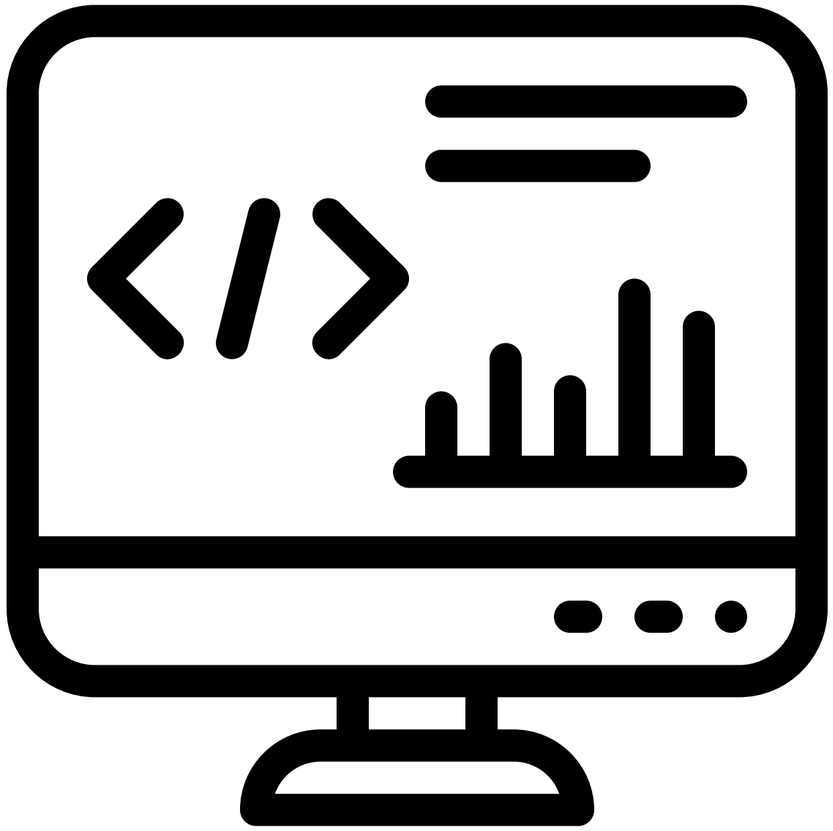
```
plt.figure(figsize=(8, 6))
sns.heatmap(df.corr(numeric_only=True),
annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Correlation Matrix")
plt.show()
```



CODE USED:

```
sns.barplot(x='expensive',
y='purchased', data=df)
plt.title("Purchase Rate: Expensive
vs Non-Expensive")
```





Machine Learning & App Interface

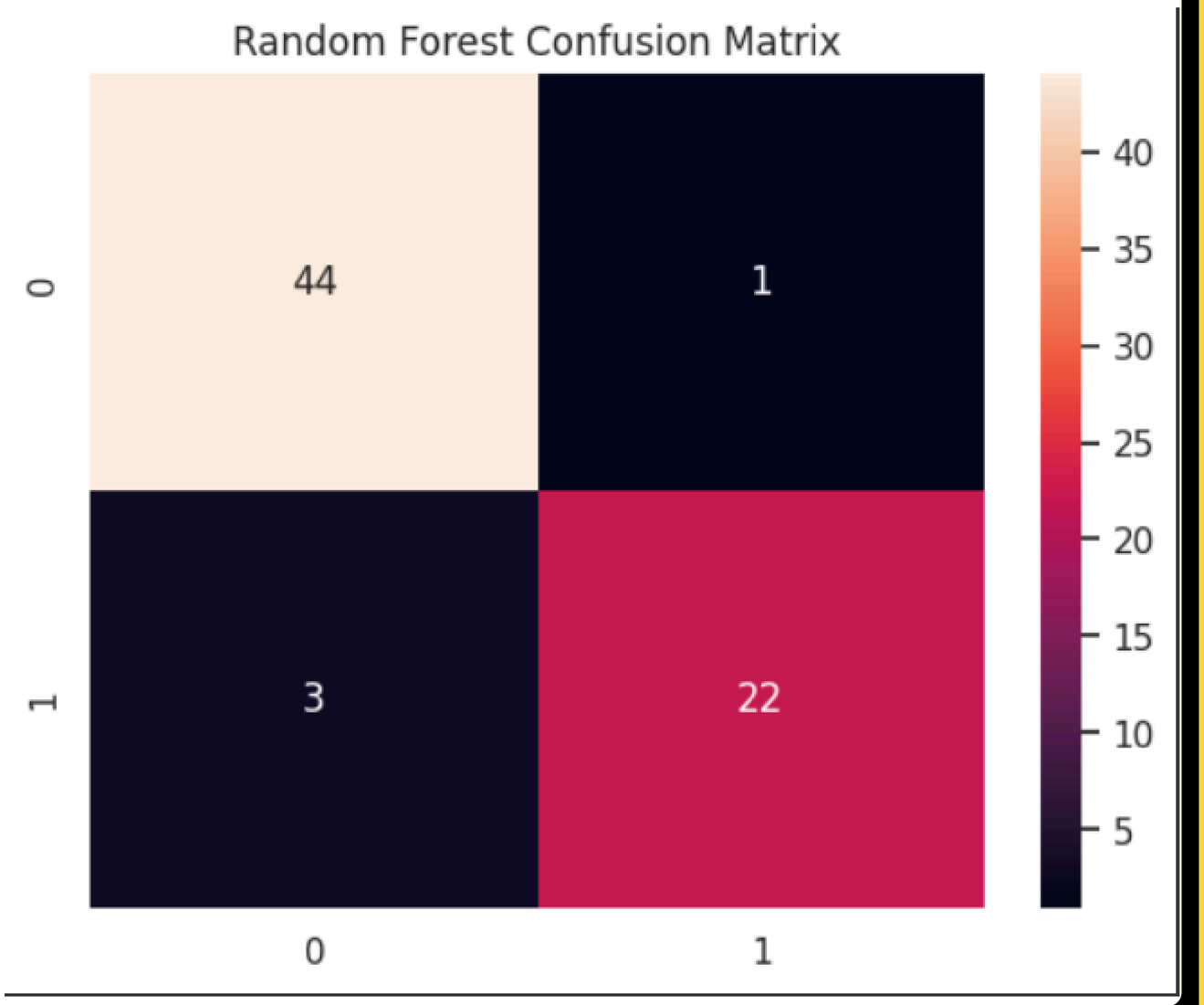
CODE USED:

```
X=dfm[['Price','Rating']]
Y=dfm['purchased']
X_test,X_train,Y_test,Y_train=train_test_split(X,Y,test_size=0.3,random_state=50)
rf=RandomForestClassifier()
rf.fit(X_train,Y_train)
rf_preds=rf.predict(X_test)
print("\n---------- RandomForestClassifier-------------")
print(classification_report(Y_test, rf_preds))
print("\n Confustion Matrix:\n", confusion_matrix(Y_test, rf_preds))
print("\n Mean Squared Error: \n", mean_squared_error(Y_test, rf_preds))
```

OUTPUT:

```
---------- RandomForestClassifier-------------
              precision    recall  f1-score   support

           0       0.94      0.98      0.96        45
           1       0.96      0.88      0.92        25

    accuracy                           0.94        70
   macro avg       0.95      0.93      0.94        70
weighted avg       0.94      0.94      0.94        70


 Confustion Matrix:
[[44  1]
[ 3 22]]

 Mean Squared Error:
0.05714285714285714
```

OUTPUT:



CODE USED:

```
sns.heatmap(confusion_matrix(Y_test, rf_preds),
                    annot=True,fmt='d')
plt.title("Random Forest Confusion Matrix")
plt.show()
```

CODE USED: (FOR VISUALS & INPUT FIELDS)

```
st.set_page_config(page_title="Book Purchase Predictor", page_icon="📚")
st.title("📚 Book Purchase Predictor")
st.markdown("Predict whether a customer will purchase a book based on product and user behavior.")

st.sidebar.header("🔧 Input Features")

# Input fields
price = st.sidebar.number_input("Book Price (£)", min_value=0.0, max_value=200.0, value=25.0)
rating = st.sidebar.selectbox("Rating", [1, 2, 3, 4, 5], index=2)

# Prepare the feature array
input_data = np.array([[price, rating]])
```

CODE USED: (FOR PREDICTION OUTPUT)

```
if st.button("🔍 Predict Purchase"):
  prediction = model.predict(input_data)[0]
  probability = model.predict_proba(input_data)[0][prediction]
  if prediction == 1:
    st.success(f"✅ This customer is likely to purchase the book! (Confidence: {probability:.2f})")
  else:
      st.warning(f"❌ This customer is unlikely to make a purchase. (Confidence: {probability:.2f})")
  # Showing the prediction probability for both classes
  st.subheader("📊 Prediction Probabilities:")
  st.write({
    "Not Purchased (0)": round(model.predict_proba(input_data)[0][0], 2),
    "Purchased (1)": round(model.predict_proba(input_data)[0][1], 2)
  })
```

🔧 Input Features

Book Price (£)

32.00 − +

Rating

4

📚 Book Purchase Predictor

Predict whether a customer will purchase a book based on product and user behavior.

🔍 Predict Purchase

✅ This customer is likely to purchase the book! (Confidence: 0.87)

📊 Prediction Probabilities:


```
{
  "Not Purchased (0)" : 0.13
  "Purchased (1)" : 0.87
}
```


RESULT

DISCUSSION

This project demonstrates how simple product attributes – price and rating – can drive purchase decisions in an e-commerce setting. By combining web scraping, simulated behavior modeling, and machine learning, I built a system that predicts purchase likelihood with over 94% accuracy. The model was deployed via a live Streamlit app, turning data insights into an interactive experience. This reflects a full data science pipeline – from raw data to real-world usability.