```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('ammar_Dataset11.csv')
df.head(10)
```

	brandName	genericName	NDC	dosage	expDate	supID	purchasePrice	sellPrice	quantity	stock
0	Aldactone	sprinolactone	12365	25	24-Dec	1	14.56	17.88	3	1
1	Amoxil	amoxicillin	17863	50	25-Dec	1	12.34	15.99	16	1
2	Glucotrol	glipizide	23123	50	23-Nov	1	9.45	10.55	4	1
3	Motrin	ibuprophen	23127	80	22-Sep	2	2.32	4.32	0	0
4	Neurontin	gabapentin	23456	80	22-Dec	2	35.67	37.66	13	1
5	Zocor	simvastatin	23467	500	23-May	1	12.44	14.54	7	1
6	Lipitor	atorvastatin	23567	10	22-Sep	1	11.23	12.55	0	0
7	Lasix	furosemide	34321	500	24-Apr	1	3.22	4.33	9	1
8	lipton	gabapentin	45652	10	24-Apr	2	5.00	8.00	4	1
9	Mobic	meloxicam	34543	15	23-Sep	1	4.65	6.76	11	1

Importing Libraries and Loding Dataset

#Random Sample

```
df.sample(10)
```

	brandName	genericName	NDC	dosage	expDate	supID	purchasePrice	sellPrice	quantity	stock
1	Amoxil	amoxicillin	17863	50	25-Dec	1	12.34	15.99	16	1
11	Neurontin	gabapentin	43234	400	22-Dec	2	33.43	40.33	4	1
5	Zocor	simvastatin	23467	500	23-May	1	12.44	14.54	7	1
8	lipton	gabapentin	45652	10	24-Apr	2	5.00	8.00	4	1
14	Lipitor	gabapentin	67876	50	26-Oct	2	2.34	12.55	0	0
22	Cozaar	losartan	78965	100	23-May	1	5.45	6.78	11	1
17	Plavix	clopidogrel	65456	75	21-Mar	1	9.33	10.43	11	1
2	Glucotrol	glipizide	23123	50	23-Nov	1	9.45	10.55	4	1
10	Naprosyn	naproxen	34567	50	24-Aug	1	2.55	5.67	2	1
13	Ambien	zolpidem	45687	500	25-Nov	2	77.87	90.76	15	1

#Shape of Data

```
df.shape
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25 entries, 0 to 24
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   brandName      25 non-null     object 
 1   genericName    25 non-null     object 
 2   NDC            25 non-null     int64  
 3   dosage         25 non-null     int64  
 4   expDate        25 non-null     object 
 5   supID          25 non-null     int64  
 6   purchasePrice  25 non-null     float64
 7   sellPrice      25 non-null     float64
 8   quantity       25 non-null     int64  
 9   stock          25 non-null     int64  
dtypes: float64(2), int64(5), object(3)
memory usage: 2.1+ KB
```

Checking The mathematical values

df.describe()

Training phase

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, classification_report
from sklearn.linear_model import LogisticRegression
```

```
X = df.drop(columns=['stock','brandName','genericName','expDate'])  # Features
y = df['stock']  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Classification_MOdel

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

```
y_pred
```

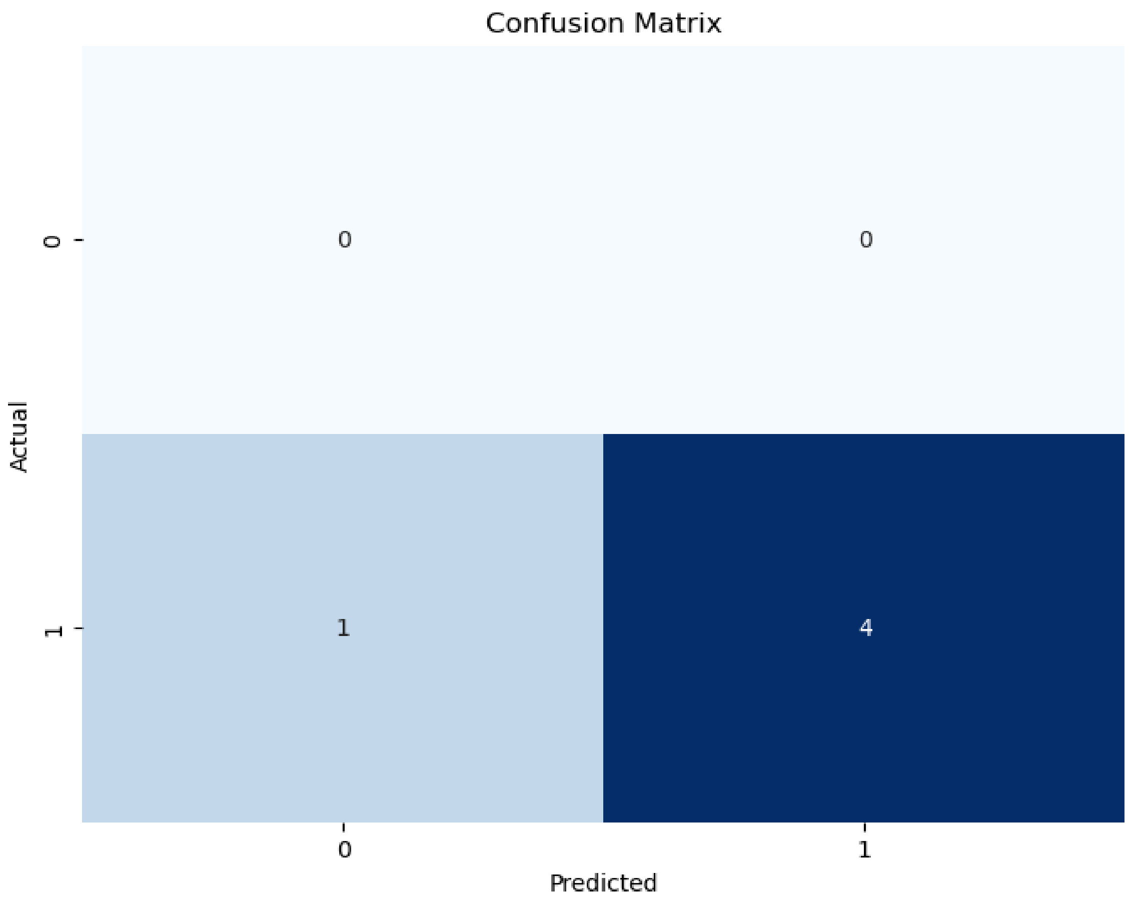
```
array([1, 1, 1, 1, 0], dtype=int64)
```

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
Accuracy: 0.8
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      0.80      0.89         5

    accuracy                           0.80         5
   macro avg       0.50      0.40      0.44         5
weighted avg       1.00      0.80      0.89         5

[[0 0]
 [1 4]]
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Recall and F-score are ill-defined and
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Recall and F-score are ill-defined and
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\HP\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469: UndefinedMetricWarning: Recall and F-score are ill-defined and
  _warn_prf(average, modifier, msg_start, len(result))
```

Confusion matrix

```
conf_matrix = confusion_matrix(y_test, y_pred)
# Confusion matrix ko heatmap ke roop mein darshayein
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```



📊 Logistic Regression on Medicine Inventory Dataset

This project uses **Logistic Regression** to predict medicine stock status (`stock`) based on features like medicine type, salt, quantity, and price. Below is a step-by-step explanation of the code:

🔷 Step 1: Import Required Libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
```

These libraries are essential for:

- **Data handling**: `pandas`
- **Model training/testing**: `scikit-learn`
- **Visualization**: `matplotlib`, `seaborn`

🔷 Step 2: Load the Dataset

```
df = pd.read_csv('ammar_Dataset11.csv')
df.head(10)
```

Loads the dataset from a CSV file and shows the first 10 records for an initial view.

🔷 Step 3: Explore the Dataset

```
df.sample(10)
df.shape
df.info()
df.describe()
```

- `.sample(10)`: Shows 10 random rows.
- `.shape`: Displays rows and columns count.
- `.info()`: Checks data types and nulls.
- `.describe()`: Provides statistical summary (mean, std, min, max, etc.).

🔷 Step 4: Preprocessing – Feature & Target Selection

```
X = df.drop(columns=['stock', 'brandName', 'genericName', 'expDate'])
y = df['stock']
```

- `X`: Selected **independent features** (excluding irrelevant columns).

- `y`: Selected **target variable** – `stock`.

🔷 Step 5: Train-Test Split

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

- Splits the dataset: 80% training, 20% testing.
- `random_state=42` ensures reproducibility.

🔷 Step 6: Model Training (Logistic Regression)

```
model = LogisticRegression()
model.fit(X_train, y_train)
```

Trains a **Logistic Regression** model on the training data.

🔷 Step 7: Model Prediction

```
y_pred = model.predict(X_test)
```

Predicts the stock status on test data.

🔷 Step 8: Model Evaluation

```
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy}')
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

- **Accuracy**: Overall correct predictions.
- **Classification Report**: Precision, Recall, F1-score for each class.
- **Confusion Matrix**: Actual vs Predicted outcomes.

🔷 Step 9: Visualize Confusion Matrix

```
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, cmap='Blues', fmt='g', cbar=False)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

- Displays the confusion matrix as a **heatmap** for easier interpretation.
- Shows how many predictions were correct or incorrect by category.

✅ Summary

- Used logistic regression to predict medicine stock.