ML Classification: Cirrhosis Prediction

IBM Machine Learning Professional Certificate

Contents

- Dataset Description
- Main Objectives
- Applying Classification Models
- Machine Learning Analysis and Findings

Dataset Description

ID	N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides
1	400	D	D-penicillamine	21464	F	Y	Y	Y	Y	14.5	261.0	2.60	156.0	1718.0	137.95	172.0
2	4500	C	D-penicillamine	20617	F	N	Y	Y	N	1.1	302.0	4.14	54.0	7394.8	113.52	88.0
3	1012	D	D-penicillamine	25594	M	N	N	N	S	1.4	176.0	3.48	210.0	516.0	96.10	55.0
4	1925	D	D-penicillamine	19994	F	N	Y	Y	S	1.8	244.0	2.54	64.0	6121.8	60.63	92.0
5	1504	CL	Placebo	13918	F	N	Y	Y	N	3.4	279.0	3.53	143.0	671.0	113.15	72.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
414	681	D	NaN	24472	F	NaN	NaN	NaN	N	1.2	NaN	2.96	NaN	NaN	NaN	NaN
415	1103	C	NaN	14245	F	NaN	NaN	NaN	N	0.9	NaN	3.83	NaN	NaN	NaN	NaN
416	1055	C	NaN	20819	F	NaN	NaN	NaN	N	1.6	NaN	3.42	NaN	NaN	NaN	NaN
417	691	C	NaN	21185	F	NaN	NaN	NaN	N	0.8	NaN	3.75	NaN	NaN	NaN	NaN
418	976	C	NaN	19358	F	NaN	NaN	NaN	N	0.7	NaN	3.29	NaN	NaN	NaN	NaN

Dataset Description

Variable Name	Role	Type	Demographic	Description	Units	Missing Values
ID	ID	Integer		unique identifier		no
N_Days	Other	Integer		number of days between registration and the earlier of death, transplantation, or study analysis time in July 1986		no
Status	Target	Categorical		status of the patient C (censored), CL (censored due to liver tx), or D (death)		no
Drug	Feature	Categorical		type of drug D-penicillamine or placebo		yes
Age	Feature	Integer	Age	age	days	no
Sex	Feature	Categorical	Sex	M (male) or F (female)		no
Ascites	Feature	Categorical		presence of ascites N (No) or Y (Yes)		yes

Dataset Description

Hepatomegaly	Feature	Categorical	presence of hepatomegaly N (No) or Y (Yes)		yes
Spiders	Feature	Categorical	presence of spiders N (No) or Y (Yes)		yes
Edema	Feature	Categorical	presence of edema N (no edema and no diuretic therapy for edema), S (edema present without diuretics, or edema resolved by diuretics), or Y (edema despite diuretic therapy)		no
Bilirubin	Feature	Continuous	serum bilirubin	mg/dl	no
Cholesterol	Feature	Integer	serum cholesterol	mg/dl	yes
Albumin	Feature	Continuous	albumin	gm/dl	no
Copper	Feature	Integer	urine copper	ug/day	yes

Dataset Description

Alk_Phos	Feature	Continuous	alkaline phosphatase	U/liter	yes
SGOT	Feature	Continuous	SGOT	U/ml	yes
Tryglicerides	Feature	Integer	tryglicerides		yes
Platelets	Feature	Integer	platelets per cubic	ml/1000	yes
Prothrombin	Feature	Continuous	prothrombin time	s	yes
Stage	Feature	Categorical	histologic stage of disease (1, 2, 3, or 4)		yes

Dataset Description

	ID	N_Days	Age	Bilirubin	Cholesterol	Albumin	Copper	Alk_Phos	SGOT	Tryglicerides	Platelets	Prothrombin
count	418.000000	418.000000	418.000000	418.000000	284.000000	418.000000	310.000000	312.000000	312.000000	282.000000	407.000000	416.000000
mean	209.500000	1917.782297	18533.351675	3.220813	369.510563	3.497440	97.648387	1982.655769	122.556346	124.702128	257.024570	10.731731
std	120.810458	1104.672992	3815.845055	4.407506	231.944545	0.424972	85.613920	2140.388824	56.699525	65.148639	98.325585	1.022000
min	1.000000	41.000000	9598.000000	0.300000	120.000000	1.960000	4.000000	289.000000	26.350000	33.000000	62.000000	9.000000
25%	105.250000	1092.750000	15644.500000	0.800000	249.500000	3.242500	41.250000	871.500000	80.600000	84.250000	188.500000	10.000000
50%	209.500000	1730.000000	18628.000000	1.400000	309.500000	3.530000	73.000000	1259.000000	114.700000	108.000000	251.000000	10.600000
75%	313.750000	2613.500000	21272.500000	3.400000	400.000000	3.770000	123.000000	1980.000000	151.900000	151.000000	318.000000	11.100000
max	418.000000	4795.000000	28650.000000	28.000000	1775.000000	4.640000	588.000000	13862.400000	457.250000	598.000000	721.000000	18.000000

Dataset Analysis

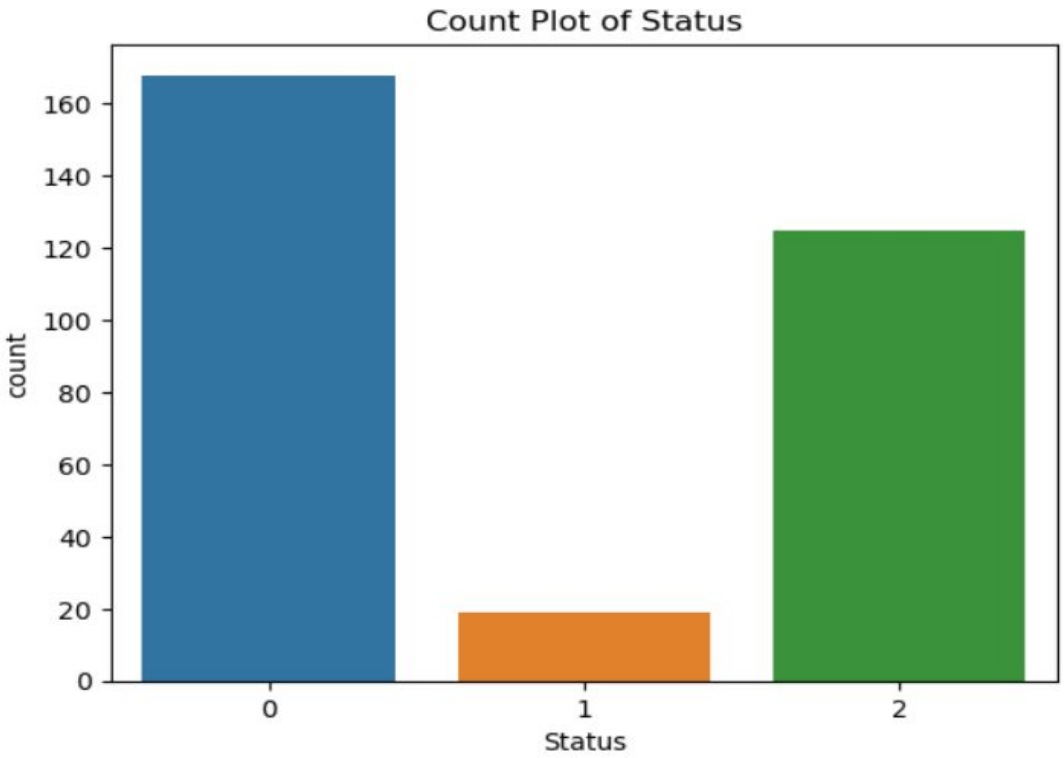
```
continuous_features
```

```
Index(['N_Days', 'Age', 'Bilirubin', 'Cholesterol', 'Albumin', 'Copper',
       'Alk_Phos', 'SGOT', 'Trygliceride s', 'Platelets', 'Prothrombin'],
      dtype='object')
```

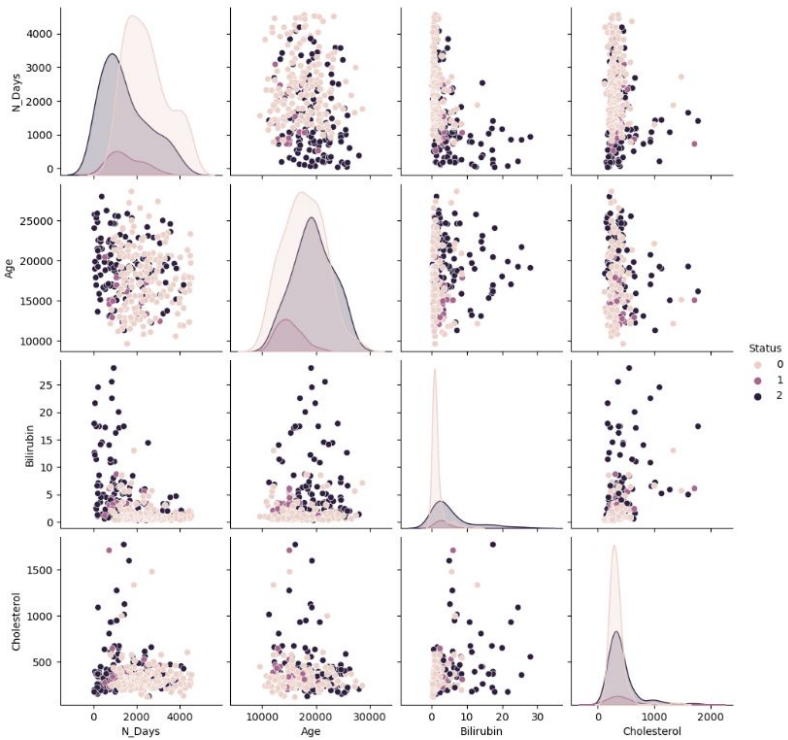
```
categorical_features
```

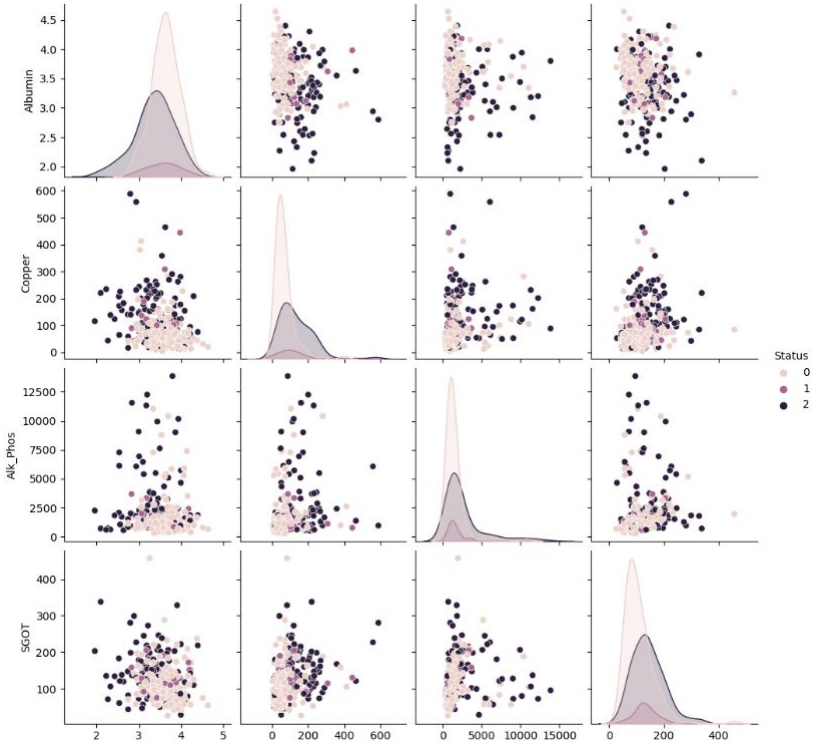
```
Index(['Status', 'Drug', 'Sex', 'Ascites', 'Hepatomegaly', 'Spiders', 'Edema',
       'Stage'],
      dtype='object')
```

Dataset Analysis

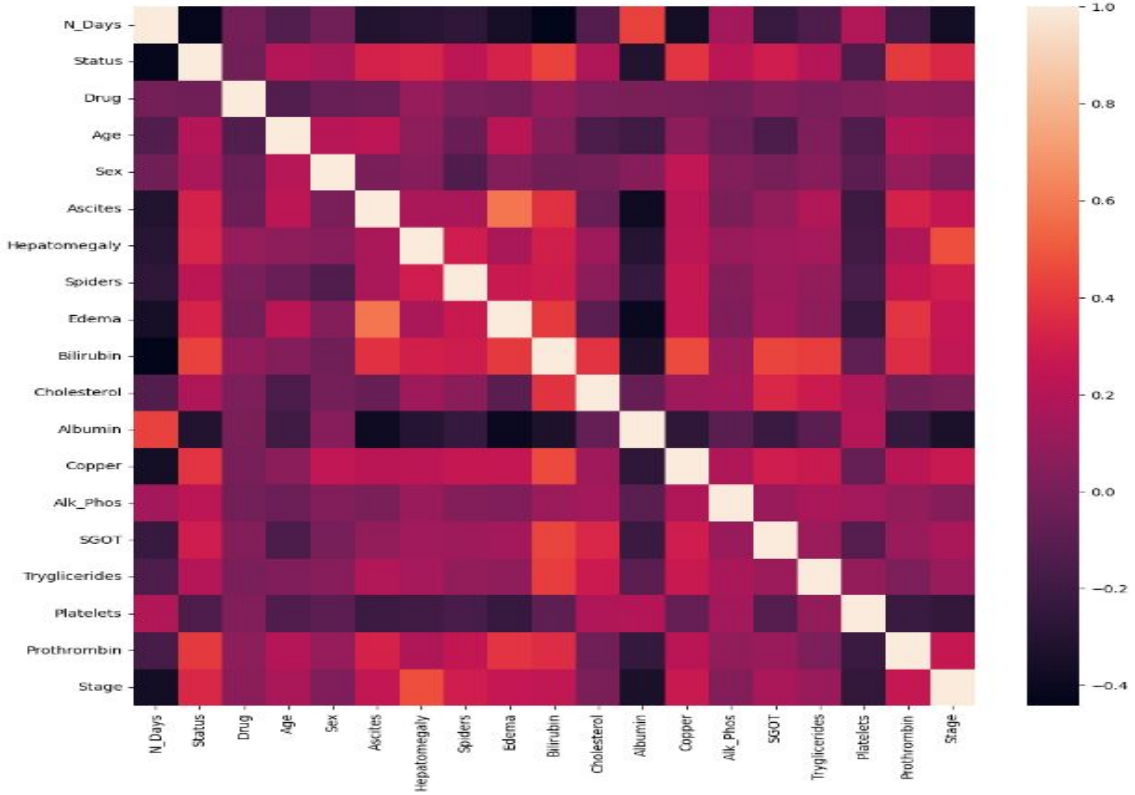


Dataset Analysis





Dataset Analysis



Feature Engineering

N_Days	Status	Drug	Age	Sex	Ascites	Hepatomegaly	Spiders	Edema	Albumin	...	Stage	Skewed_Bilirubin	Skewed_Prothrombin
400	2	0	59	0	1	1	1	2	2.60	...	4.0	1.075729	0.222375
4500	0	0	56	0	0	1	1	0	4.14	...	3.0	0.552967	0.222373
1012	2	0	70	1	0	0	0	1	3.48	...	4.0	0.620947	0.222375
1925	2	0	55	0	0	1	1	1	2.54	...	4.0	0.690551	0.222373
1504	1	1	38	0	0	1	1	0	3.53	...	3.0	0.849818	0.222374
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1153	0	0	61	0	0	1	0	0	3.58	...	2.0	0.293317	0.222373
994	0	1	58	0	0	0	0	0	2.75	...	2.0	0.293317	0.222374
939	0	0	62	0	0	0	0	0	3.35	...	2.0	0.674929	0.222373
839	0	0	38	0	0	0	0	0	3.16	...	2.0	0.718915	0.222373
788	0	1	33	0	0	0	1	0	3.79	...	2.0	0.972117	0.222374

Data Splitting

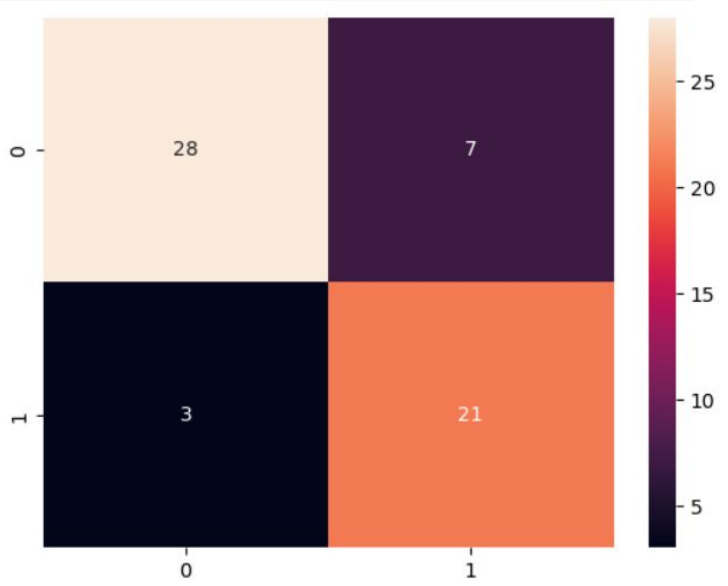
```
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.metrics import classification_report, confusion_matrix, ConfusionMatrixDisplay
feature_cols = [col_name for col_name in dataset.columns if col_name != 'target']

# Get the split indexes
strat_shuf_split = StratifiedShuffleSplit(n_splits=1,
                                          test_size=0.3,
                                          random_state=42)

train_idx, test_idx = next(strat_shuf_split.split(dataset[feature_cols], dataset.target))
# Create the dataframes
X_train = dataset.loc[train_idx, feature_cols]
y_train = dataset.loc[train_idx, 'target']

X_test  = dataset.loc[test_idx, feature_cols]
y_test  = dataset.loc[test_idx, 'target']
```

Machine Learning Analysis

Decision Tree Classifier:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
tree_model = DecisionTreeClassifier(criterion='gini', max_depth=15, min_samples_leaf=5, random_state=123)
tree_model.fit(X_train, y_train)
y_pred_tree = tree_model.predict(X_test)

accuracy_tree = accuracy_score(y_test, y_pred_tree)
print(f'Accuracy (Decision Tree): {accuracy_tree:.2f}')

print(classification_report(y_test, y_pred_tree))
```

```
Accuracy (Decision Tree): 0.71
              precision    recall  f1-score   support

           0       0.85      0.63      0.72        35
           2       0.61      0.83      0.70        24

    accuracy                           0.71        59
   macro avg       0.73      0.73      0.71        59
weighted avg       0.75      0.71      0.71        59
```

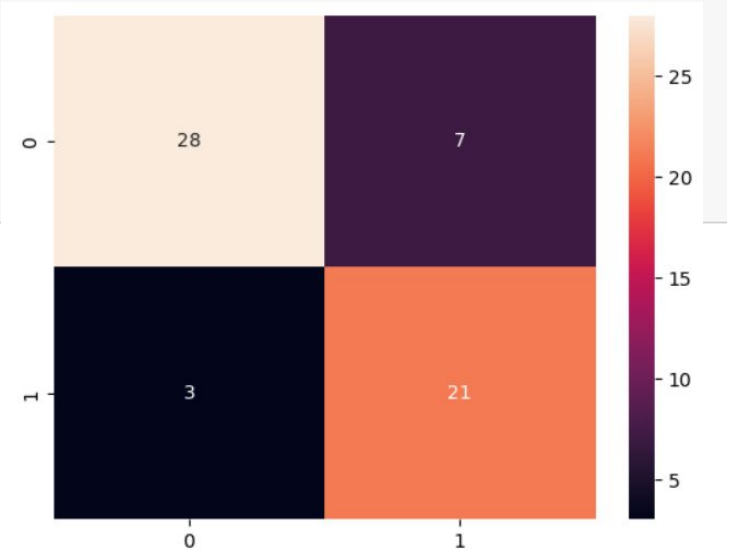


Machine Learning Analysis

Random Forest Classifier:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
max_features=round(np.sqrt(M_features))-1
rf_model = RandomForestClassifier(max_features=max_features,n_estimators=100,criterion='gini',random_state=123)
rf_model.fit(X_train, y_train)

y_pred_rf = rf_model.predict(X_test)

accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f'Accuracy (Random Forest): {accuracy_rf:.2f}')

print(classification_report(y_test, y_pred_rf))
```

```
Accuracy (Random Forest): 0.83
              precision    recall  f1-score   support

           0       0.93      0.77      0.84        35
           2       0.73      0.92      0.81        24

    accuracy                           0.83        59
   macro avg       0.83      0.84      0.83        59
weighted avg       0.85      0.83      0.83        59
```

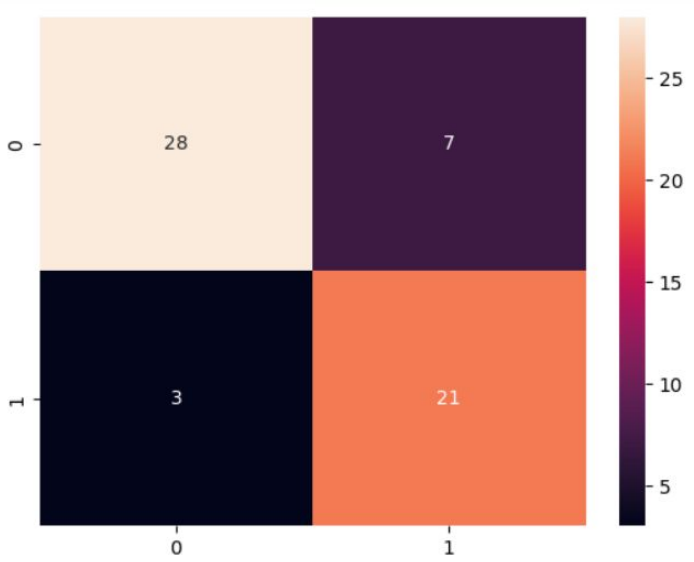


Machine Learning Analysis

SVM:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
svm_model = SVC(kernel='rbf',C=100)
svm_model.fit(X_train, y_train)

y_pred_svm = svm_model.predict(X_test)

accuracy_svm = accuracy_score(y_test, y_pred_svm)
print(f'Accuracy (SVM): {accuracy_svm:.2f}')

print(classification_report(y_test, y_pred_svm))
```

```
Accuracy (SVM): 0.78
              precision    recall  f1-score   support

           0       0.73      1.00      0.84        35
           2       1.00      0.46      0.63        24

    accuracy                           0.78        59
   macro avg       0.86      0.73      0.74        59
weighted avg       0.84      0.78      0.76        59
```

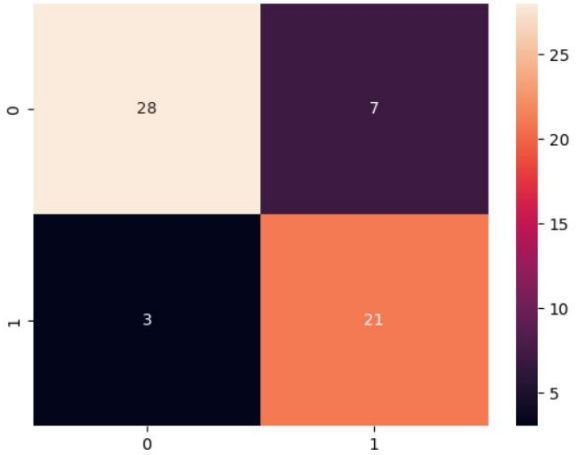


Machine Learning Analysis

KNN:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
knn_model = KNeighborsClassifier(n_neighbors=11, weights='distance')
knn_model.fit(X_train, y_train)

y_pred_knn = knn_model.predict(X_test)

accuracy_knn = accuracy_score(y_test, y_pred_knn)
print(f'Accuracy (KNN): {accuracy_knn:.2f}')

print('Classification Report (KNN):\n', classification_report(y_test, y_pred_knn))
```

```
Accuracy (KNN): 0.75
Classification Report (KNN):
              precision    recall  f1-score   support

           0       0.75      0.86      0.80        35
           2       0.74      0.58      0.65        24

    accuracy                           0.75        59
   macro avg       0.74      0.72      0.73        59
weighted avg       0.74      0.75      0.74        59
```

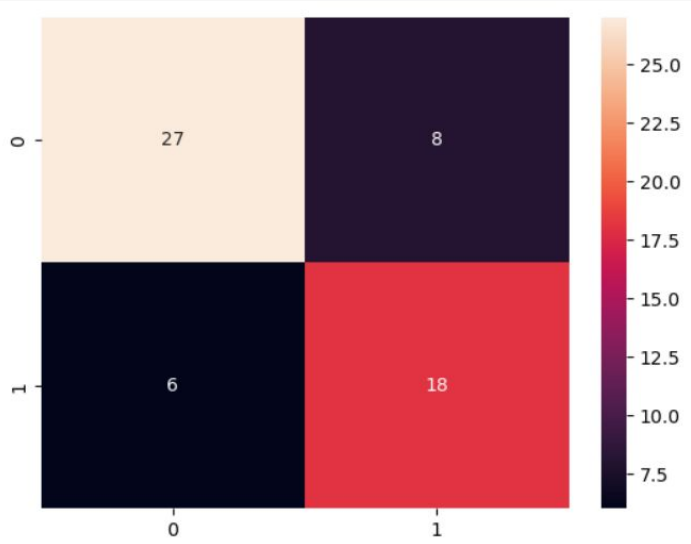


Machine Learning Analysis

Logistic Regression:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
logistic_model = LogisticRegression(random_state=12, penalty='l2', multi_class='multinomial', solver='lbfgs', max_iter=1000)
logistic_model.fit(X_train, y_train)

y_pred = logistic_model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')

print('Classification Report:\n', classification_report(y_test, y_pred))
```

```
Accuracy: 0.76
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.77      0.79        35
           2       0.69      0.75      0.72        24

    accuracy                           0.76        59
   macro avg       0.76      0.76      0.76        59
weighted avg       0.77      0.76      0.76        59
```

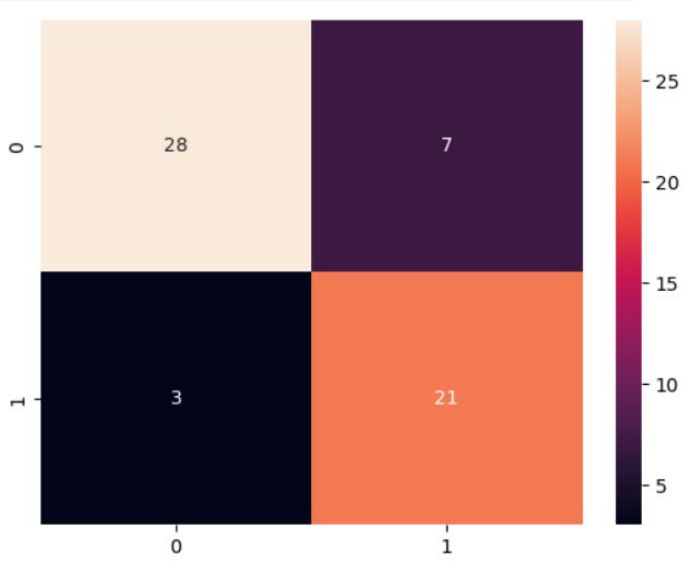


Machine Learning Analysis

Adaboost Classifier:

```
from sklearn.ensemble import AdaBoostClassifier
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = AdaBoostClassifier(n_estimators=200,random_state=0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.90	0.80	0.85	35
2	0.75	0.88	0.81	24
accuracy			0.83	59
macro avg	0.83	0.84	0.83	59
weighted avg	0.84	0.83	0.83	59

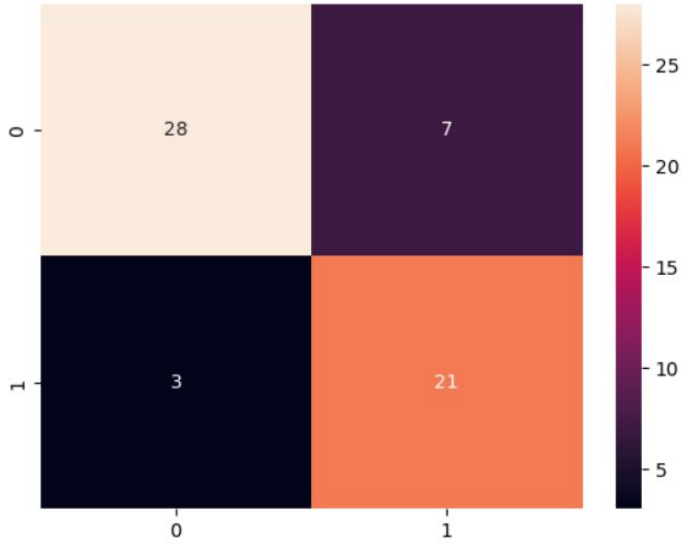


Machine Learning Analysis

Model Comparison:

1. Decision Tree Classifier
2. Random Forest Classifier
3. SVM
4. KNN
5. Logistic Regression
6. AdaBoost Classifier

	precision	recall	f1-score	support
0	0.90	0.80	0.85	35
2	0.75	0.88	0.81	24
accuracy			0.83	59
macro avg	0.83	0.84	0.83	59
weighted avg	0.84	0.83	0.83	59



THANK YOU