STATS 3DA3

Homework Assignment 6

Zhujunsheng 400332040, jiming yang (400324487)

2025-03-21

(1)

Classification problem: Predict whether a patient has heart disease based on features such as age, sex, and chest pain type.

(2)

We will transform the 'num' variable into a binary outcome.

(3)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

```
df = pd.read_csv("https://raw.githubusercontent.com/PratheepaJ/datasets/refs/heads/master/ass6-
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
```

```
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        299 non-null    float64
 12  thal      301 non-null    float64
 13  num       303 non-null    int64
dtypes: float64(3), int64(11)
memory usage: 33.3 KB
```

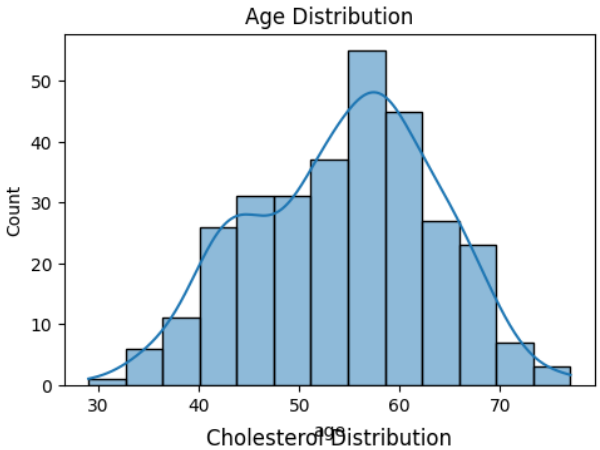
```
df.describe()
```

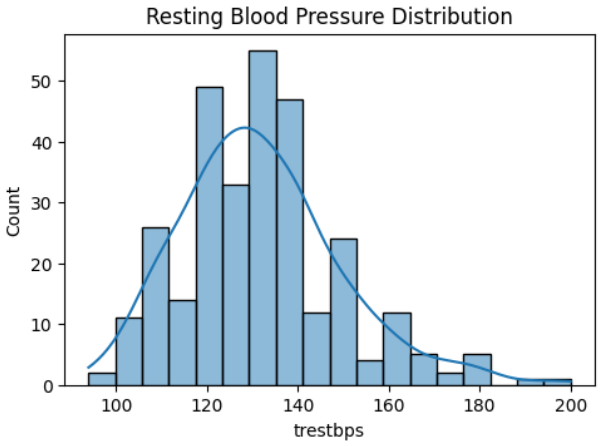
	age	sex	cp	trestbps	chol	fbs	restecg	thalach
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.438944	0.679868	3.158416	131.689769	246.693069	0.148515	0.990099	149.607261
std	9.038662	0.467299	0.960126	17.599748	51.776918	0.356198	0.994971	22.875003
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.000000
25%	48.000000	0.000000	3.000000	120.000000	211.000000	0.000000	0.000000	133.500000
50%	56.000000	1.000000	3.000000	130.000000	241.000000	0.000000	1.000000	153.000000
75%	61.000000	1.000000	4.000000	140.000000	275.000000	0.000000	2.000000	166.000000
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000

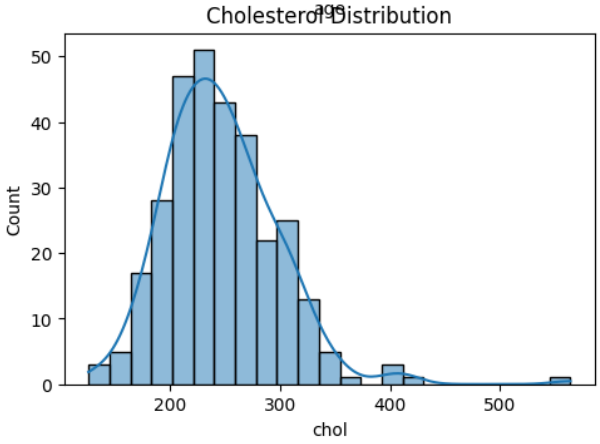
```
df.shape
```

```
(303, 14)
```

```
# plot the distributions of some variables
plt.figure(figsize=(12, 8))
plt.subplot(2, 2, 1)
sns.histplot(df['age'], kde=True)
plt.title('Age Distribution')

plt.subplot(2, 2, 2)
sns.histplot(df['trestbps'], kde=True)
plt.title('Resting Blood Pressure Distribution')

plt.subplot(2, 2, 3)
sns.histplot(df['chol'], kde=True)
plt.title('Cholesterol Distribution')

plt.show()
```







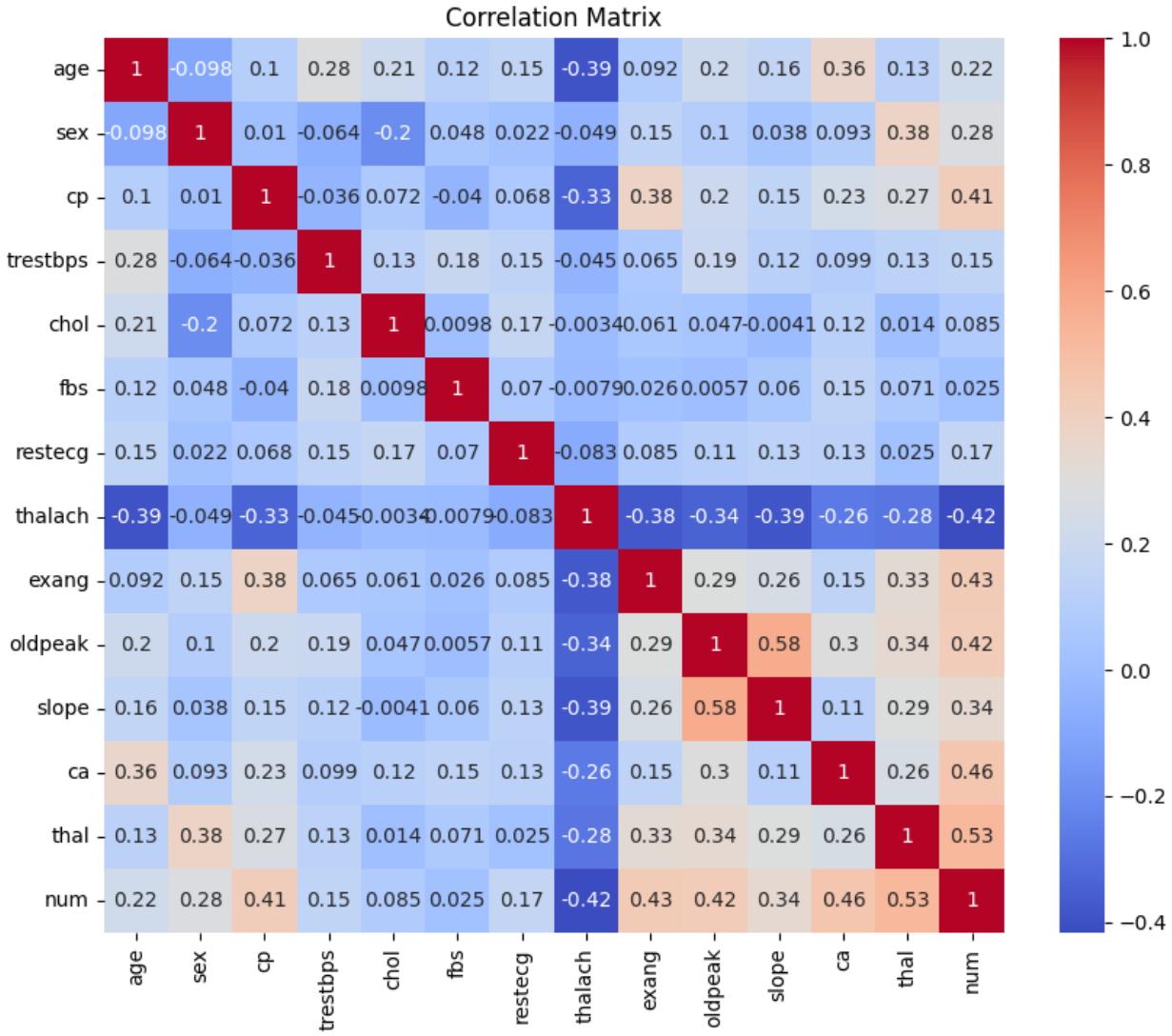
The dataset contains 303 rows and 14 variables. All the variables are stored as numeric. The average age is 54, the average chol is 247, the average trestbps is 132. The distribution of age is approximate normal, the distribution of trestbps and chol are right skewed.

(4)

```
df['num'] = df['num'].apply(lambda x: 1 if x in [1, 2, 3, 4] else 0)
```

(5)

```
correlation_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



The correlation coefficient matrix tells us that the target variable num is moderate positively associated with ca and thal. There is a moderate negative relationship between num and thalach.

(6)

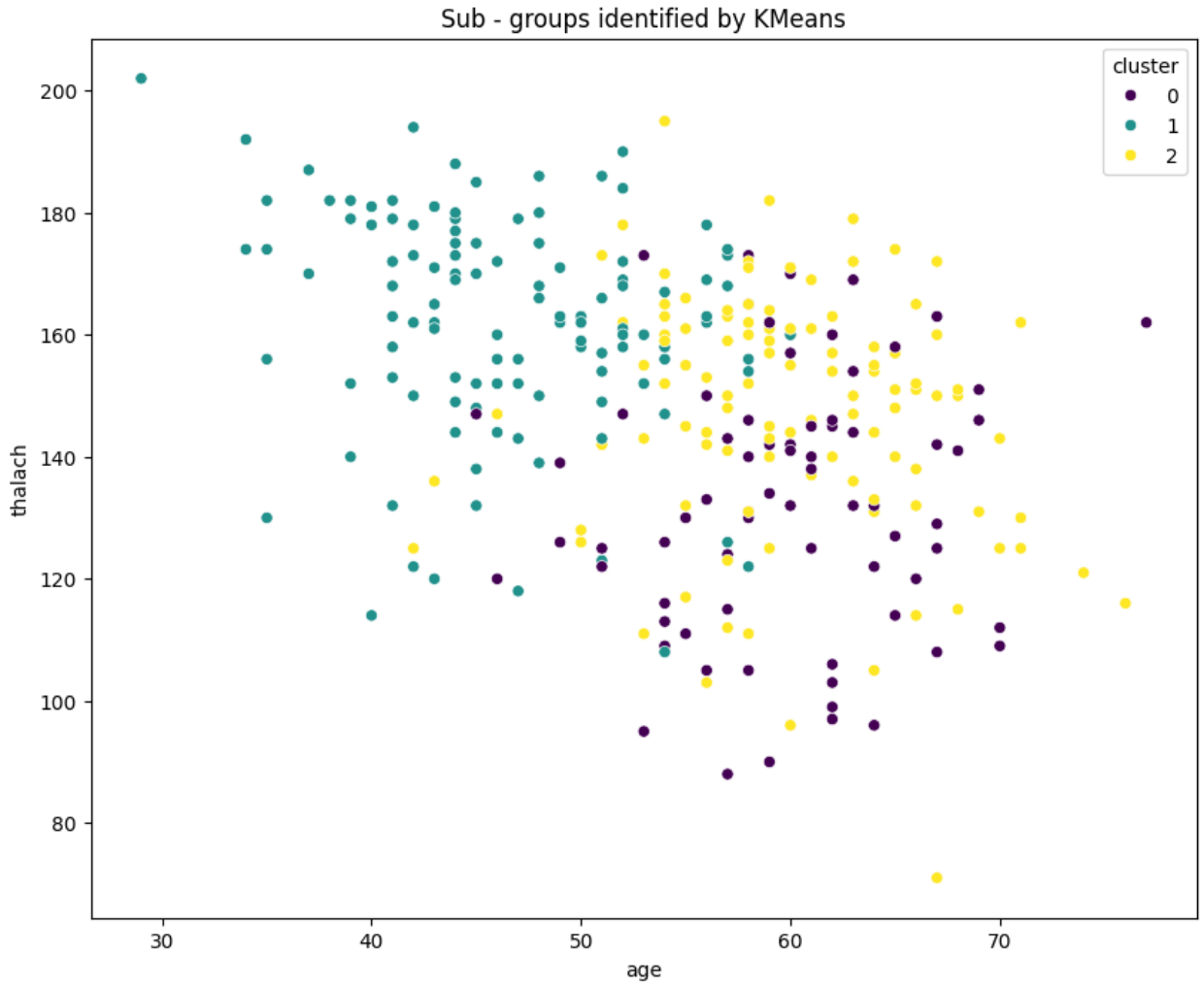
```
df = df.dropna()
df.shape
```

```
(297, 14)
```

Number of observations after dropping missing values: 279.

(7)

```
numerical_columns = ["age", "trestbps", "chol", "thalach", "oldpeak", "ca"]
numerical_df = df[numerical_columns]
scaler = StandardScaler()
scaled_numerical_df = scaler.fit_transform(numerical_df)
kmeans = KMeans(n_clusters=3, random_state=1)
df['cluster'] = kmeans.fit_predict(scaled_numerical_df)
# visualize the clustering results
plt.figure(figsize=(10, 8))
sns.scatterplot(x='age', y='thalach', hue='cluster', data=df, palette='viridis')
plt.title('Sub - groups identified by KMeans')
plt.show()
```



(8)

```
X = df.drop(['num', 'cluster'], axis=1)
y = df['num']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1)
```

(9)

We selected Logistic Regression and Decision Tree Classifier here.

Logistic Regression is selected because it is a linear model, and the coefficients can directly explain the impact of each feature on the classification result.

Decision Tree Classifier is selected because it can generate an intuitive decision tree, which is convenient for understanding the decision-making process.

(10)

We will use the following two metrics:

- Accuracy: The proportion of correctly classified samples to the total number of samples. Formula: Accuracy = (TP + TN) / (TP + TN + FP + FN).
- Precision: The proportion of actual positive samples among the samples predicted as positive. Formula: Precision = TP / (TP + FP).

where TP = True Positives, TN = True Negatives, FP = False Positives, FN = False Negatives

(11)

```
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.model_selection import GridSearchCV

logistic_regression = LogisticRegression(max_iter=3000)
logistic_regression.fit(X_train, y_train)

decision_tree = DecisionTreeClassifier()

param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [3, 4, 5, 6, 7, 8],
    'min_samples_split': [2, 3, 4, 5],
    'min_samples_leaf': [1, 2, 3]
}
```

```
# use GridSearchCV to find the best parameters
grid_search = GridSearchCV(decision_tree, param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)

best_decision_tree = grid_search.best_estimator_
grid_search.best_params_
```

```
{'criterion': 'entropy',
 'max_depth': 7,
 'min_samples_leaf': 2,
 'min_samples_split': 2}
```

(12)

```
selector = SelectKBest(score_func=f_classif, k=5)
X_train_selected = selector.fit_transform(X_train, y_train)
X_test_selected = selector.transform(X_test)

decision_tree_selected = DecisionTreeClassifier()
decision_tree_selected.fit(X_train_selected, y_train)
```

```
DecisionTreeClassifier()
```

(13)

```
y_pred_logistic = logistic_regression.predict(X_test)
y_pred_decision_tree = best_decision_tree.predict(X_test)
y_pred_decision_tree_selected = decision_tree_selected.predict(X_test_selected)

accuracy_logistic = accuracy_score(y_test, y_pred_logistic)
```

```
precision_logistic = precision_score(y_test, y_pred_logistic)


accuracy_decision_tree = accuracy_score(y_test, y_pred_decision_tree)
precision_decision_tree = precision_score(y_test, y_pred_decision_tree)


accuracy_decision_tree_selected = accuracy_score(y_test, y_pred_decision_tree_selected)
precision_decision_tree_selected = precision_score(y_test, y_pred_decision_tree_selected)


print("Logistic Regression - Accuracy: {:.4f} (Formula: (TP + TN) / (TP + TN + FP + FN))".forma
print("Logistic Regression - Precision: {:.4f} (Formula: TP / (TP + FP))".format(precision_logi
print("Decision Tree - Accuracy: {:.4f} (Formula: (TP + TN) / (TP + TN + FP + FN))".format(accu
print("Decision Tree - Precision: {:.4f} (Formula: TP / (TP + FP))".format(precision_decision_t
print("Decision Tree with Feature Selection - Accuracy: {:.4f} (Formula: (TP + TN) / (TP + TN +
print("Decision Tree with Feature Selection - Precision: {:.4f} (Formula: TP / (TP + FP))".form
```

```
Logistic Regression - Accuracy: 0.8111 (Formula: (TP + TN) / (TP + TN + FP + FN))
Logistic Regression - Precision: 0.7500 (Formula: TP / (TP + FP))
Decision Tree - Accuracy: 0.7111 (Formula: (TP + TN) / (TP + TN + FP + FN))
Decision Tree - Precision: 0.6275 (Formula: TP / (TP + FP))
Decision Tree with Feature Selection - Accuracy: 0.8000 (Formula: (TP + TN) / (TP + TN + FP + F
Decision Tree with Feature Selection - Precision: 0.7442 (Formula: TP / (TP + FP))
```

The logistic regression model has the best performance, because it has the highest accuracy and precision. The Decision tree with feature selection performs better than the decision tree without feature selection. it indicates that reducing irrelevant or redundant features helps the model focus on the most important attributes, improving its classification quality.

(14)

```
coefficients = pd.DataFrame({'feature': X.columns, 'coefficient': logistic_regression.coef_[0]}
coefficients = coefficients.sort_values(by='coefficient', ascending=False)
coefficients
```

	feature	coefficient
11	ca	1.222785
1	sex	1.046301
9	oldpeak	0.627025
8	exang	0.615781
2	cp	0.331671
12	thal	0.315468
6	restecg	0.306580
3	trestbps	0.022108
4	chol	0.003197
7	thalach	-0.040283
0	age	-0.042133
10	slope	-0.109174
5	fbs	-0.656134

The coefficient of ca is positive and relatively large, which indicates that a higher number of major vessels is associated with a greater likelihood of having heart disease. On the other hand, the coefficient of fbs is negative and has a relatively large absolute value, suggesting that high fasting blood sugar may reduce the probability of having heart disease.

(15)

```
subgroup_models = {}
for cluster in df['cluster'].unique():
    subgroup_df = df[df['cluster'] == cluster]
    X_sub = subgroup_df.drop(['num', 'cluster'], axis=1)
    y_sub = subgroup_df['num']
    X_train_sub, X_test_sub, y_train_sub, y_test_sub = train_test_split(X_sub, y_sub, test_siz
    model = LogisticRegression(max_iter=3000)
    model.fit(X_train_sub, y_train_sub)
```

```
    subgroup_models[cluster] = model


# evaluate the subgroup models
for cluster, model in subgroup_models.items():
    subgroup_test_df = df[df['cluster'] == cluster]
    X_test_sub = subgroup_test_df.drop(['num', 'cluster'], axis=1)
    y_test_sub = subgroup_test_df['num']
    y_pred_sub = model.predict(X_test_sub)
    accuracy_sub = accuracy_score(y_test_sub, y_pred_sub)
    precision_sub = precision_score(y_test_sub, y_pred_sub)
    print(f"Cluster {cluster} - Accuracy: {accuracy_sub}, Precision: {precision_sub}")
```

```
Cluster 2 - Accuracy: 0.7592592592592593, Precision: 0.7560975609756098
Cluster 0 - Accuracy: 0.8888888888888888, Precision: 0.9253731343283582
Cluster 1 - Accuracy: 0.905982905982906, Precision: 0.85
```

In comparison with the results from Q13, the subgroup models show varying performance. For Cluster 0 and Cluster 1, both accuracy and precision are higher than the basic decision tree model, and comparable to or better than the logistic regression model. However, the performance of Cluster 2 is similar to the decision-tree model with feature selection and lower than the logistic regression model.

(16)

- Jiming yang (400324487): Question to Question 10.
- Zhujunsheng 400332040: Question 11 to Question 17

(17)

https://github.com/JIMINGyang123/assign6