

Step 0: Import Libraries

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.feature_selection import SelectKBest, chi2, RFE
```

Step 1: Load Dataset

```
data = pd.read_csv("heart.csv") # Replace with your dataset path
```

```
# Quick overview
data.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
#   Column      Non-Null Count  Dtype
---  -
0   age         1025 non-null   int64
1   sex         1025 non-null   int64
2   cp          1025 non-null   int64
3   trestbps    1025 non-null   int64
4   chol        1025 non-null   int64
5   fbs         1025 non-null   int64
6   restecg     1025 non-null   int64
7   thalach     1025 non-null   int64
8   exang       1025 non-null   int64
9   oldpeak     1025 non-null   float64
10  slope       1025 non-null   int64
11  ca          1025 non-null   int64
12  thal        1025 non-null   int64
13  target      1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

```
data.isnull().sum()
```

```
age      0
sex      0
cp       0
trestbps 0
chol     0
fbs      0
restecg  0
thalach  0
exang    0
oldpeak  0
slope    0
ca       0
thal     0
target   0
dtype: int64
```

Step 2: Handle Missing Values

```
# Using median for numeric columns
numeric_cols = data.select_dtypes(include=np.number).columns
imputer = SimpleImputer(strategy='median')
data[numeric_cols] = imputer.fit_transform(data[numeric_cols])
```

Step 3: Split Dataset & Scale

```
X = data.drop("target", axis=1)
y = data["target"]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

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

Step 4: Apply ML Models with Grid Search

```
# 1. Logistic Regression

lr = LogisticRegression(max_iter=1000)
lr_params = {'C':[0.01,0.1,1,10], 'solver':['liblinear']}
lr_grid = GridSearchCV(lr, lr_params, cv=5)
lr_grid.fit(X_train, y_train)
y_pred_lr = lr_grid.predict(X_test)
print("Logistic Regression Accuracy:", accuracy_score(y_test, y_pred_lr))
```

Logistic Regression Accuracy: 0.7951219512195122

```
# 2. Random Forest

rf = RandomForestClassifier(random_state=42)
rf_params = {'n_estimators':[50,100,200], 'max_depth':[None,5,10]}
rf_grid = GridSearchCV(rf, rf_params, cv=5)
rf_grid.fit(X_train, y_train)
y_pred_rf = rf_grid.predict(X_test)
print("Random Forest Accuracy:", accuracy_score(y_test, y_pred_rf))
```

Random Forest Accuracy: 0.9853658536585366

```
# 3. Support Vector Classifier

svc = SVC()
svc_params = {'C':[0.1,1,10], 'kernel':['linear','rbf']}
svc_grid = GridSearchCV(svc, svc_params, cv=5)
svc_grid.fit(X_train, y_train)
y_pred_svc = svc_grid.predict(X_test)
print("SVC Accuracy:", accuracy_score(y_test, y_pred_svc))
```

SVC Accuracy: 0.975609756097561

Step 5: Feature Selection

```
from sklearn.feature_selection import SelectKBest, chi2

# Ensure features are non-negative (no scaling)
X_nonneg = data.drop("target", axis=1)
X_nonneg = X_nonneg.apply(lambda x: x - x.min() if x.min() < 0 else x)

kbest = SelectKBest(score_func=chi2, k=5)
X_kbest = kbest.fit_transform(X_nonneg, y)
print("Top features by SelectKBest:", X_kbest.columns[kbest.get_support()])
```

```
Top features by SelectKBest: Index(['cp', 'thalach', 'exang', 'oldpeak', 'ca'], dtype='object')
```

```
from sklearn.feature_selection import RFE
from sklearn.ensemble import RandomForestClassifier

# RFE with Random Forest
rfe = RFE(estimator=RandomForestClassifier(random_state=42), n_features_to_select=5)
rfe.fit(X_scaled, y) # X_scaled from earlier scaling

# Get top features selected by RFE
selected_features_rfe = X.columns[rfe.support_]
print("Top features by RFE:", selected_features_rfe)

# Create dataset with selected features
X_selected_rfe = data[selected_features_rfe]
```

```
Top features by RFE: Index(['age', 'cp', 'thalach', 'oldpeak', 'thal'], dtype='object')
```

Step 6: Train Models on Selected Features

```
# Method 1: SelectKBest

# Get the top features selected by SelectKBest
selected_features = X_nonneg.columns[kbest.get_support()]
X_selected = data[selected_features]

# Split into training and test sets
X_train_sel, X_test_sel, y_train_sel, y_test_sel = train_test_split(
    X_selected, y, test_size=0.2, random_state=42
)

# Scale features based on training data
scaler_sel = StandardScaler()
X_train_sel_scaled = scaler_sel.fit_transform(X_train_sel)
X_test_sel_scaled = scaler_sel.transform(X_test_sel)

# Train Random Forest on selected features
rf_grid.fit(X_train_sel_scaled, y_train_sel)
y_pred_sel = rf_grid.predict(X_test_sel_scaled)
print("Random Forest Accuracy on Selected Features:", accuracy_score(y_test_sel, y_pred_sel))

# Optional: Train Logistic Regression and SVC on selected features too
lr_grid.fit(X_train_sel_scaled, y_train_sel)
y_pred_lr_sel = lr_grid.predict(X_test_sel_scaled)
print("Logistic Regression Accuracy on Selected Features:", accuracy_score(y_test_sel, y_pred_lr_sel))

svc_grid.fit(X_train_sel_scaled, y_train_sel)
y_pred_svc_sel = svc_grid.predict(X_test_sel_scaled)
print("SVC Accuracy on Selected Features:", accuracy_score(y_test_sel, y_pred_svc_sel))
```

```
Random Forest Accuracy on Selected Features: 0.9658536585365853
Logistic Regression Accuracy on Selected Features: 0.7853658536585366
SVC Accuracy on Selected Features: 0.8536585365853658
```

```
# Method 2: RFE

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

# Split into train and test sets
X_train_rfe, X_test_rfe, y_train_rfe, y_test_rfe = train_test_split(
    X_selected_rfe, y, test_size=0.2, random_state=42
)

# Scale features
scaler_rfe = StandardScaler()
X_train_rfe_scaled = scaler_rfe.fit_transform(X_train_rfe)
X_test_rfe_scaled = scaler_rfe.transform(X_test_rfe)

# Train Random Forest
rf_grid.fit(X_train_rfe_scaled, y_train_rfe)
y_pred_rf_rfe = rf_grid.predict(X_test_rfe_scaled)
print("Random Forest Accuracy on RFE Features:", accuracy_score(y_test_rfe, y_pred_rf_rfe))

# Train Logistic Regression
```

```
lr_grid.fit(X_train_rfe_scaled, y_train_rfe)
y_pred_lr_rfe = lr_grid.predict(X_test_rfe_scaled)
print("Logistic Regression Accuracy on RFE Features:", accuracy_score(y_test_rfe, y_pred_lr_rfe))

# Train SVC
svc_grid.fit(X_train_rfe_scaled, y_train_rfe)
y_pred_svc_rfe = svc_grid.predict(X_test_rfe_scaled)
print("SVC Accuracy on RFE Features:", accuracy_score(y_test_rfe, y_pred_svc_rfe))
```

```
Random Forest Accuracy on RFE Features: 0.9853658536585366
Logistic Regression Accuracy on RFE Features: 0.7853658536585366
SVC Accuracy on RFE Features: 0.8390243902439024
```