

```
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/c

```
import pandas as pd
```

```
df = pd.read_csv('heart (1).csv')
print("Shape:", df.shape)
print("Missing values:\n", df.isnull().sum())
print("\nFirst 5 rows:\n", df.head())
```

Shape: (1025, 14)

Missing values:

```
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
```

First 5 rows:

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

	ca	thal	target
0	2	3	0
1	0	3	0
2	0	3	0
3	1	3	0
4	3	2	0

```
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.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report

from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import SelectKBest, f_classif
```

```

Load dataset
f = pd.read_csv('heart (1).csv')
df = df.drop(columns=['target'])
y = df['target']

Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y)

Handle Missing Values
Median imputation for outliers
imputer = SimpleImputer(strategy='median')
X_train_imp = imputer.fit_transform(X_train)
X_test_imp = imputer.transform(X_test)

Feature Scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_imp)
X_test_scaled = scaler.transform(X_test_imp)

GridSearchCV
1 Random Forest
rf_params = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5]

f_grid = GridSearchCV(RandomForestClassifier(random_state=42), rf_params, cv=5, scoring='f1')
f_grid.fit(X_train_scaled, y_train)

2 Logistic Regression
lr_params = {
    'C': [0.01, 0.1, 1.0, 10.0],
    'penalty': ['l2'],
    'solver': ['lbfgs']

lr_grid = GridSearchCV(LogisticRegression(random_state=42), lr_params, cv=5, scoring='f1')
lr_grid.fit(X_train_scaled, y_train)

rint("=== FULL FEATURE SET EVALUATION ===")
evaluate_model(lr_grid.best_estimator_, X_test_scaled, y_test, "Full RF")
evaluate_model(lr_grid.best_estimator_, X_test_scaled, y_test, "Full Logistic Regression")

function to evaluate models
def evaluate_model(model, X_te, y_te, name="Model"):
    preds = model.predict(X_te)
    acc = accuracy_score(y_te, preds)
    prec = precision_score(y_te, preds)
    rec = recall_score(y_te, preds)
    f1 = f1_score(y_te, preds)
    print(f"--- {name} Results ---")
    print(f"Accuracy: {acc:.4f} | Precision: {prec:.4f} | Recall: {rec:.4f} | F1-Score: {f1:.4f}")
    return {'Accuracy': acc, 'Precision': prec, 'Recall': rec, 'F1': f1}

```

Feature Selection Methods

```
_FEATURES = 5
```

```
Method A: Tree-Based Feature Importance (Random Forest)
```

```
f_selector = RandomForestClassifier(random_state=42)
f_selector.fit(X_train_scaled, y_train)
importances = rf_selector.feature_importances_
f_selected_indices = np.argsort(importances)[::-1][:K_FEATURES]
f_selected_features = X.columns[rf_selected_indices].tolist()
```

```
Method B: SelectKBest (ANOVA F-value)
```

```
kb_selector = SelectKBest(score_func=f_classif, k=K_FEATURES)
kb_selector.fit(X_train_scaled, y_train)
kb_selected_indices = skb_selector.get_support(indices=True)
kb_selected_features = X.columns[skb_selected_indices].tolist()
```

```
rint(f"Top {K_FEATURES} Features (RF Importance): {rf_selected_features}")
rint(f"Top {K_FEATURES} Features (SelectKBest ANOVA): {skb_selected_features}\n")
```

Train & Evaluate on Selected Features

```
_tr_rf_sub = X_train_scaled[:, rf_selected_indices]
_te_rf_sub = X_test_scaled[:, rf_selected_indices]

_tr_skb_sub = X_train_scaled[:, skb_selected_indices]
_te_skb_sub = X_test_scaled[:, skb_selected_indices]
```

```
Retrain RF on RF Selected Features
```

```
f_sub_grid = GridSearchCV(RandomForestClassifier(random_state=42), rf_params, cv=5, scoring='f1')
f_sub_grid.fit(X_tr_rf_sub, y_train)
```

```
Retrain Logistic Regression on SelectKBest Features
```

```
r_sub_grid = GridSearchCV(LogisticRegression(random_state=42), lr_params, cv=5, scoring='f1')
r_sub_grid.fit(X_tr_skb_sub, y_train)
```

```
rint("=== SELECTED FEATURES EVALUATION ===")
valuate_model(rf_sub_grid.best_estimator_, X_te_rf_sub, y_test, "RF (Tree-Selected Features)")
valuate_model(lr_sub_grid.best_estimator_, X_te_skb_sub, y_test, "Logistic Regression (SelectKBest Features)")
```

```
=== FULL FEATURE SET EVALUATION ===
```

```
--- Full RF Results ---
```

```
Accuracy: 1.0000 | Precision: 1.0000 | Recall: 1.0000 | F1-Score: 1.0000
```

```
--- Full Logistic Regression Results ---
```

```
Accuracy: 0.8098 | Precision: 0.7619 | Recall: 0.9143 | F1-Score: 0.8312
```

```
Top 5 Features (RF Importance): ['cp', 'thalach', 'ca', 'oldpeak', 'thal']
```

```
Top 5 Features (SelectKBest ANOVA): ['cp', 'thalach', 'exang', 'oldpeak', 'ca']
```

```
=== SELECTED FEATURES EVALUATION ===
```

```
--- RF (Tree-Selected Features) Results ---
```

```
Accuracy: 0.9902 | Precision: 0.9813 | Recall: 1.0000 | F1-Score: 0.9906
```

```
--- Logistic Regression (SelectKBest Features) Results ---
```

```
Accuracy: 0.8049 | Precision: 0.7778 | Recall: 0.8667 | F1-Score: 0.8198
```

```
{'Accuracy': 0.8048780487804879,  
 'Precision': 0.7777777777777778,  
 'Recall': 0.8666666666666667,  
 'F1': 0.8198198198198198}
```