

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import SelectKBest, chi2, RFE
from sklearn.metrics import accuracy_score, precision_score, recall_s
f1_score, classification_report
```

```
import kagglehub
path = kagglehub.dataset_download("sullivanova/cardiovascular-disease-d
print("Path to dataset files:", path)
```

Using Colab cache for faster access to the 'cardiovascular-disease-dataset' d
Path to dataset files: /kaggle/input/cardiovascular-disease-dataset

```
import os
print(os.listdir(path))

df=pd.read_csv(path+"/cardio_train.csv", sep=';')
df.head()
df.shape
```

```
['cardio_train.csv']
(70000, 13)
```

```
print(df.isnull().sum())
print(df.dtypes)
print(df.describe())
```

```
id          0
age         0
gender      0
height      0
weight      0
ap_hi       0
ap_lo       0
cholesterol 0
gluc        0
smoke       0
alco        0
active      0
cardio      0
dtype: int64
id          int64
age         int64
gender      int64
height      int64
weight      float64
```

```
ap_hi      int64
ap_lo      int64
cholesterol int64
gluc       int64
smoke      int64
alco       int64
active     int64
cardio     int64
dtype: object
```

	id	age	gender	height	weight
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000
mean	49972.419900	19468.865814	1.349571	164.359229	74.205690
std	28851.302323	2467.251667	0.476838	8.210126	14.395757
min	0.000000	10798.000000	1.000000	55.000000	10.000000
25%	25006.750000	17664.000000	1.000000	159.000000	65.000000
50%	50001.500000	19703.000000	1.000000	165.000000	72.000000
75%	74889.250000	21327.000000	2.000000	170.000000	82.000000
max	99999.000000	23713.000000	2.000000	250.000000	200.000000

	ap_hi	ap_lo	cholesterol	gluc	smoke
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000
mean	128.817286	96.630414	1.366871	1.226457	0.088129
std	154.011419	188.472530	0.680250	0.572270	0.283484
min	-150.000000	-70.000000	1.000000	1.000000	0.000000
25%	120.000000	80.000000	1.000000	1.000000	0.000000
50%	120.000000	80.000000	1.000000	1.000000	0.000000
75%	140.000000	90.000000	2.000000	1.000000	0.000000
max	16020.000000	11000.000000	3.000000	3.000000	1.000000

	alco	active	cardio
count	70000.000000	70000.000000	70000.000000
mean	0.053771	0.803729	0.499700
std	0.225568	0.397179	0.500003
min	0.000000	0.000000	0.000000
25%	0.000000	1.000000	0.000000
50%	0.000000	1.000000	0.000000
75%	0.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000

```

if 'id' in df.columns:
    df = df.drop('id', axis=1)

# Convert age from days to years (easier to interpret)
df['age'] = (df['age'] / 365).round().astype(int)

# The blood pressure columns contain data-entry errors (negative values,
# ap_hi < ap_lo, and physiologically impossible extreme values).
# These are cleaned here as part of preparing the dataset.
df = df[(df['ap_hi'] > 0) & (df['ap_lo'] > 0)]
df = df[df['ap_hi'] >= df['ap_lo']]
df = df[(df['ap_hi'] <= 250) & (df['ap_lo'] <= 200)]
print("Missing values after cleaning:")
print(df.isnull().sum())
print("Shape after cleaning:", df.shape)

X = df.drop('cardio', axis=1)
y = df['cardio']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

```

```

Missing values after cleaning:
age          0
gender       0
height       0
weight       0
ap_hi        0
ap_lo        0
cholesterol  0
gluc         0
smoke        0
alco         0
active       0
cardio       0
dtype: int64
Shape after cleaning: (68711, 12)

```

```

param_grids = {
    "Logistic Regression": (
        LogisticRegression(max_iter=1000),
        {'C': [0.01, 0.1, 1, 10]}
    ),
    "Decision Tree": (
        DecisionTreeClassifier(random_state=42),
        {'max_depth': [4, 6, 8, 10]}
    ),
    "Random Forest": (
        RandomForestClassifier(random_state=42),
        {'n_estimators': [100, 200], 'max_depth': [6, 10]}
    ),
}

```

```
def run_grid_search(X_tr, X_te, y_tr, y_te, label):
    results = []
    for name, (model, params) in param_grids.items():
        grid = GridSearchCV(model, params, cv=3, scoring='accuracy', n_jobs=
        grid.fit(X_tr, y_tr)
        y_pred = grid.predict(X_te)
        results.append({
            "Feature Set": label,
            "Model": name,
            "Best Params": grid.best_params_,
            "Accuracy": accuracy_score(y_te, y_pred),
            "Precision": precision_score(y_te, y_pred),
            "Recall": recall_score(y_te, y_pred),
            "F1": f1_score(y_te, y_pred)
        })
    return pd.DataFrame(results)

baseline_df = run_grid_search(X_train_scaled, X_test_scaled, y_train, y_test
print("\nBaseline Model Performance (All Features):")
print(baseline_df)
```

Baseline Model Performance (All Features):

	Feature Set	Model	Best Params
0	All Features	Logistic Regression	{'C': 1}
1	All Features	Decision Tree	{'max_depth': 4}
2	All Features	Random Forest	{'max_depth': 10, 'n_estimators': 100}

	Accuracy	Precision	Recall	F1
0	0.722113	0.757696	0.644359	0.696447
1	0.725751	0.750579	0.667451	0.706578
2	0.727861	0.765492	0.648625	0.702229

```
selector_chi2 = SelectKBest(score_func=chi2, k='all')
selector_chi2.fit(X_train, y_train) # chi2 needs non-negative, unscaled da

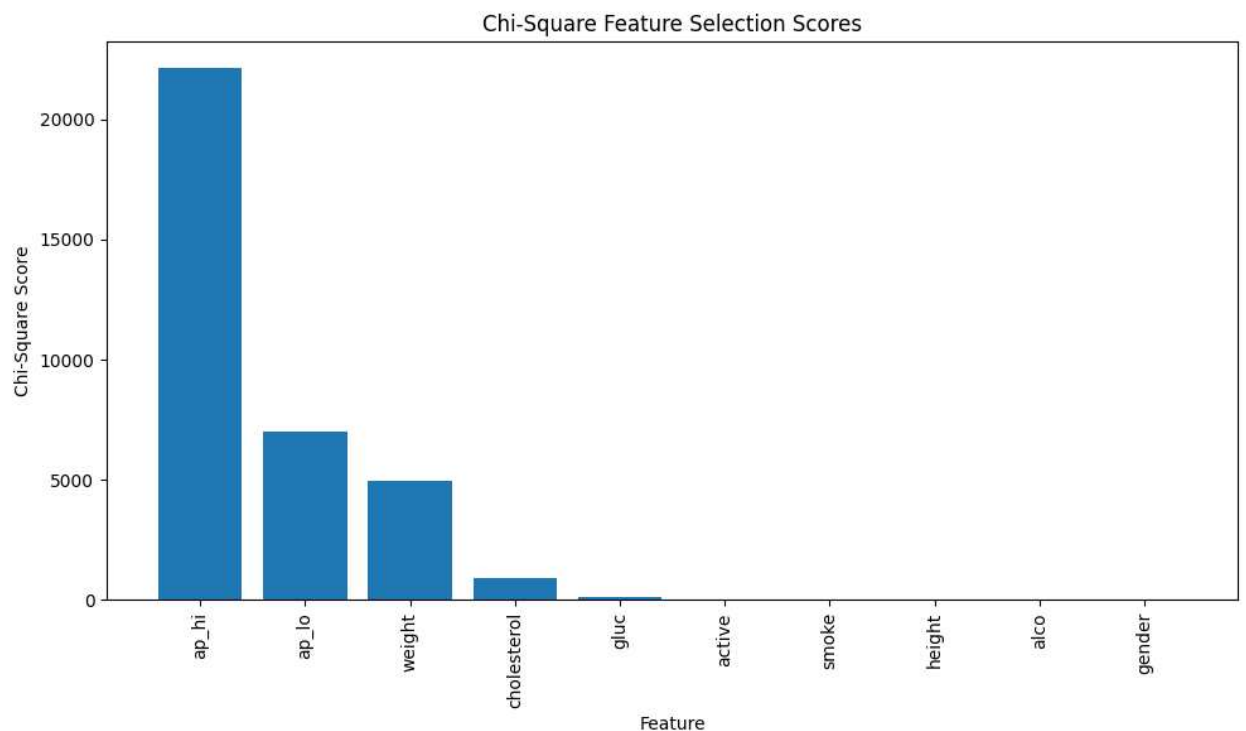
chi2_scores = pd.DataFrame({
    'Feature': X.columns,
    'Score': selector_chi2.scores_
}).sort_values(by='Score', ascending=False)
print("\nChi-Square Scores:")
print(chi2_scores)

plt.figure(figsize=(10, 6))
plt.bar(chi2_scores['Feature'], chi2_scores['Score'])
plt.xlabel("Feature")
plt.ylabel("Chi-Square Score")
plt.title("Chi-Square Feature Selection Scores")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

k = 8
top_chi2_features = chi2_scores['Feature'].head(k).tolist()
print("Selected Features (Chi-Square):", top_chi2_features)
```

Chi-Square Scores:

	Feature	Score
4	ap_hi	22121.745439
5	ap_lo	6991.411271
3	weight	4982.855299
6	cholesterol	925.500719
7	gluc	120.904362
10	active	16.942425
8	smoke	9.353186
2	height	2.638872
9	alco	2.417181
1	gender	0.798138
0	age	NaN



Selected Features (Chi-Square): ['ap_hi', 'ap_lo', 'weight', 'cholesterol', 'gluc', 'active', 'smoke', 'height', 'alco', 'gender']

```
rf_selector = RandomForestClassifier(n_estimators=200, random_state=42)
rfe = RFE(estimator=rf_selector, n_features_to_select=k)
rfe.fit(X_train_scaled, y_train)
```

```
rfe_ranking = pd.DataFrame({
    'Feature': X.columns,
    'Ranking': rfe.ranking_
}).sort_values(by='Ranking')
print("\nRFE Ranking:")
print(rfe_ranking)
```

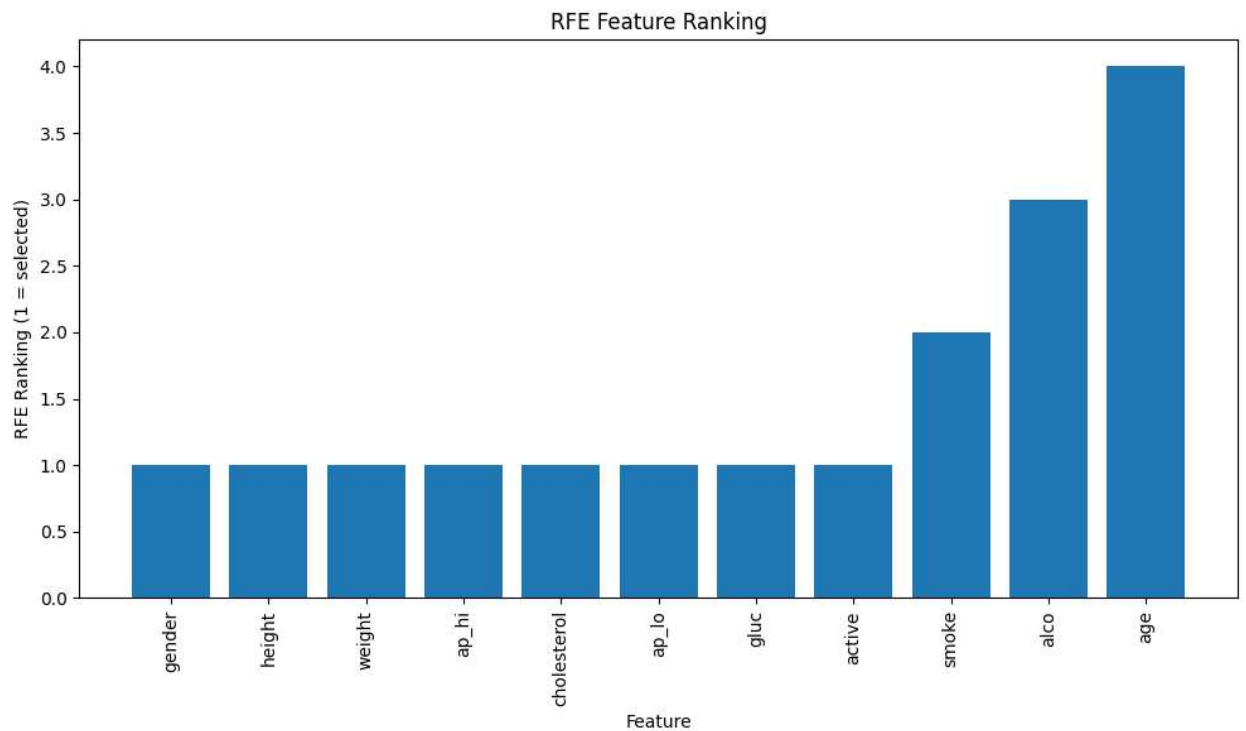
```
plt.figure(figsize=(10, 6))
plt.bar(rfe_ranking['Feature'], rfe_ranking['Ranking'])
```

```
plt.xlabel("Feature")
plt.ylabel("RFE Ranking (1 = selected)")
plt.title("RFE Feature Ranking")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()

rfe_selected_features = X.columns[rfe.get_support()].tolist()
print("Selected Features (RFE):", rfe_selected_features)
```

RFE Ranking:

	Feature	Ranking
1	gender	1
2	height	1
3	weight	1
4	ap_hi	1
6	cholesterol	1
5	ap_lo	1
7	gluc	1
10	active	1
8	smoke	2
9	alco	3
0	age	4



Selected Features (RFE): ['gender', 'height', 'weight', 'ap_hi', 'ap_lo', 'cholesterol', 'gluc', 'active', 'smoke', 'alco', 'age']

```
def evaluate_on_features(feature_list, label):
    Xtr = X_train[feature_list]
    Xte = X_test[feature_list]
    sc = StandardScaler()
    Xtr_s = sc.fit_transform(Xtr)
    Xte_s = sc.transform(Xte)
    return run_grid_search(Xtr_s, Xte_s, y_train, y_test, label)

chi2_results = evaluate_on_features(top_chi2_features, "Chi-Square Selected")
rfe_results = evaluate_on_features(rfe_selected_features, "RFE Selected")

print("\nModel Performance (Chi-Square Selected Features):")
print(chi2_results)
print("\nModel Performance (RFE Selected Features):")
print(rfe_results)
```

Model Performance (Chi-Square Selected Features):

	Feature Set	Model \	Best Params	Accuracy	Precision	Recall \
0	Chi-Square Selected	Logistic Regression				
1	Chi-Square Selected	Decision Tree				
2	Chi-Square Selected	Random Forest				
0			{'C': 0.01}	0.722695	0.759375	0.643330
1			{'max_depth': 4}	0.725751	0.750579	0.667451
2			{'max_depth': 10, 'n_estimators': 100}	0.727425	0.764604	0.648772
	F1					
0	0.696552					
1	0.706578					
2	0.701941					

Model Performance (RFE Selected Features):

	Feature Set	Model	Best Params	Accuracy	Precision	Recall	F1
0	RFE Selected	Logistic Regression	{'C': 10}				
1	RFE Selected	Decision Tree	{'max_depth': 4}				
2	RFE Selected	Random Forest	{'max_depth': 6, 'n_estimators': 200}				
0	0.722186	0.757738	0.644507	0.696551			
1	0.725751	0.750579	0.667451	0.706578			
2	0.728153	0.764645	0.650831	0.703162			

```

final_comparison = pd.concat([baseline_df, chi2_results, rfe_results], ignore
print("\nFinal Comparison - All Features vs Chi-Square vs RFE:")
print(final_comparison[["Feature Set", "Model", "Accuracy", "Precision", "Re

final_comparison.to_csv("final_comparison_results.csv", index=False)

```

Final Comparison - All Features vs Chi-Square vs RFE:

	Feature Set	Model	Accuracy	Precision	Recall	\
0	All Features	Logistic Regression	0.722113	0.757696	0.644359	
1	All Features	Decision Tree	0.725751	0.750579	0.667451	
2	All Features	Random Forest	0.727861	0.765492	0.648625	
3	Chi-Square Selected	Logistic Regression	0.722695	0.759375	0.643330	
4	Chi-Square Selected	Decision Tree	0.725751	0.750579	0.667451	
5	Chi-Square Selected	Random Forest	0.727425	0.764604	0.648772	
6	RFE Selected	Logistic Regression	0.722186	0.757738	0.644507	
7	RFE Selected	Decision Tree	0.725751	0.750579	0.667451	
8	RFE Selected	Random Forest	0.728153	0.764645	0.650831	

F1

0	0.696447
1	0.706578
2	0.702229
3	0.696552
4	0.706578
5	0.701941
6	0.696551
7	0.706578
8	0.703162