Cardiovascular Disease Prediction

A Machine Learning Approach

Introduction

- Importance of predicting CVD early
- Global impact (leading cause of death worldwide)
- Role of Machine Learning in healthcare

Objective

- Build a model to predict CVD using patient data
- Improve early detection and prevention

Dataset Description

- Source of dataset (e.g., Kaggle, UCI Repository)
- No. of samples and features
- Features: Age, Gender, Blood Pressure, Cholesterol, Glucose, Smoking, etc.
- Target: Presence/Absence of CVD

Data Preprocessing

- Handling missing values
- Encoding categorical variables
- Feature scaling (StandardScaler/MinMaxScaler)
- Train-test split

Exploratory Data Analysis (EDA)

- Distribution of features (histograms, boxplots)
- Correlation heatmap
- Key insights (e.g., age & cholesterol strongly linked to CVD)

Model Building

- Algorithms used:
- Logistic Regression
- Decision Tree
- Random Forest
- K-Nearest Neighbors
- Support Vector Machine
- Hyperparameter tuning (e.g., GridSearchCV)

Model Performance

- Accuracy, Precision, Recall, F1-Score
- Confusion matrix
- Best-performing model and its score

PROGRAM

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

Saving cardio_train (1).csv to cardio_train (1) (1).csv

```
!pip install seaborn scikit-learn --quiet
```

```
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
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
```

```
df = pd.read_csv("cardio_train (1).csv", sep=';')
df.head()
```

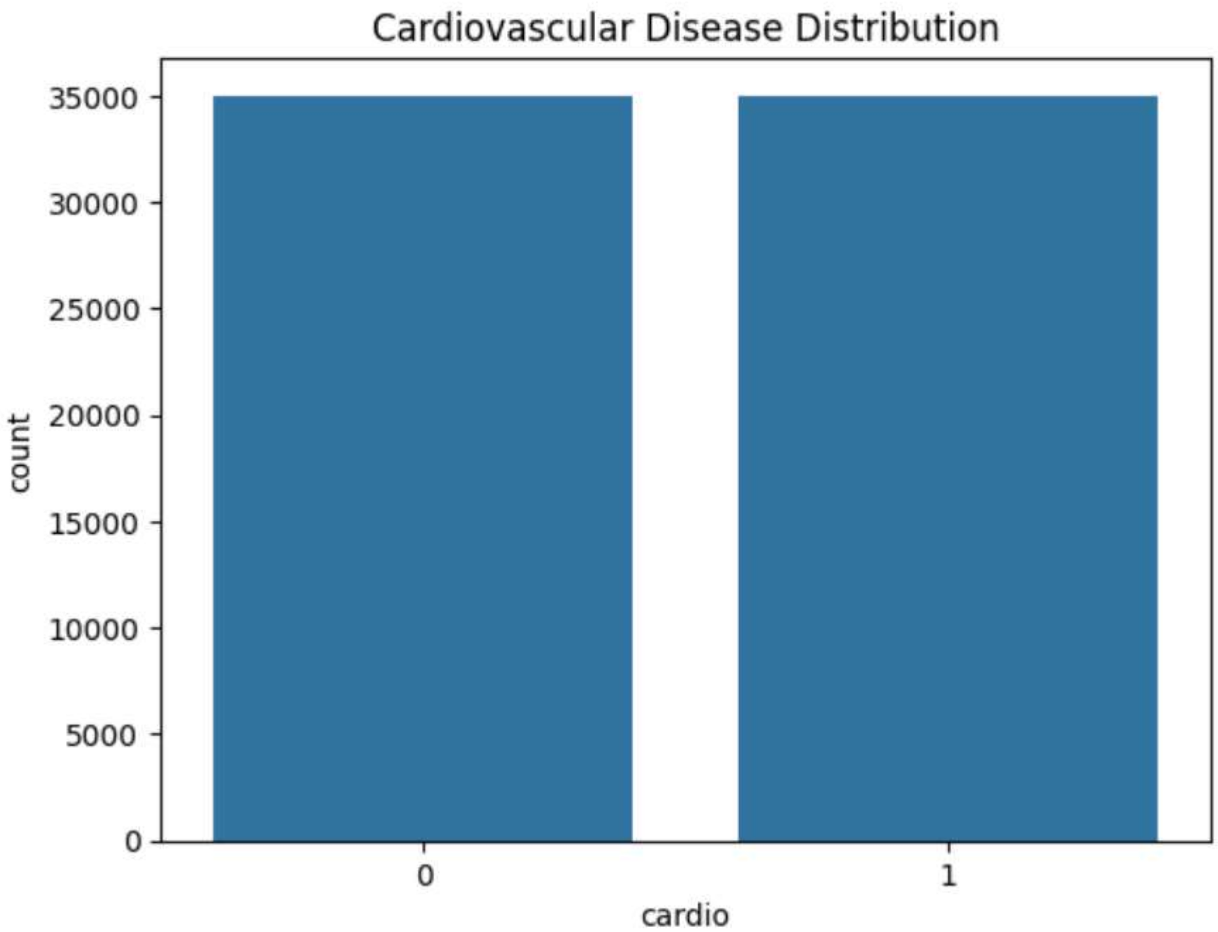
PROGRAM

	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0

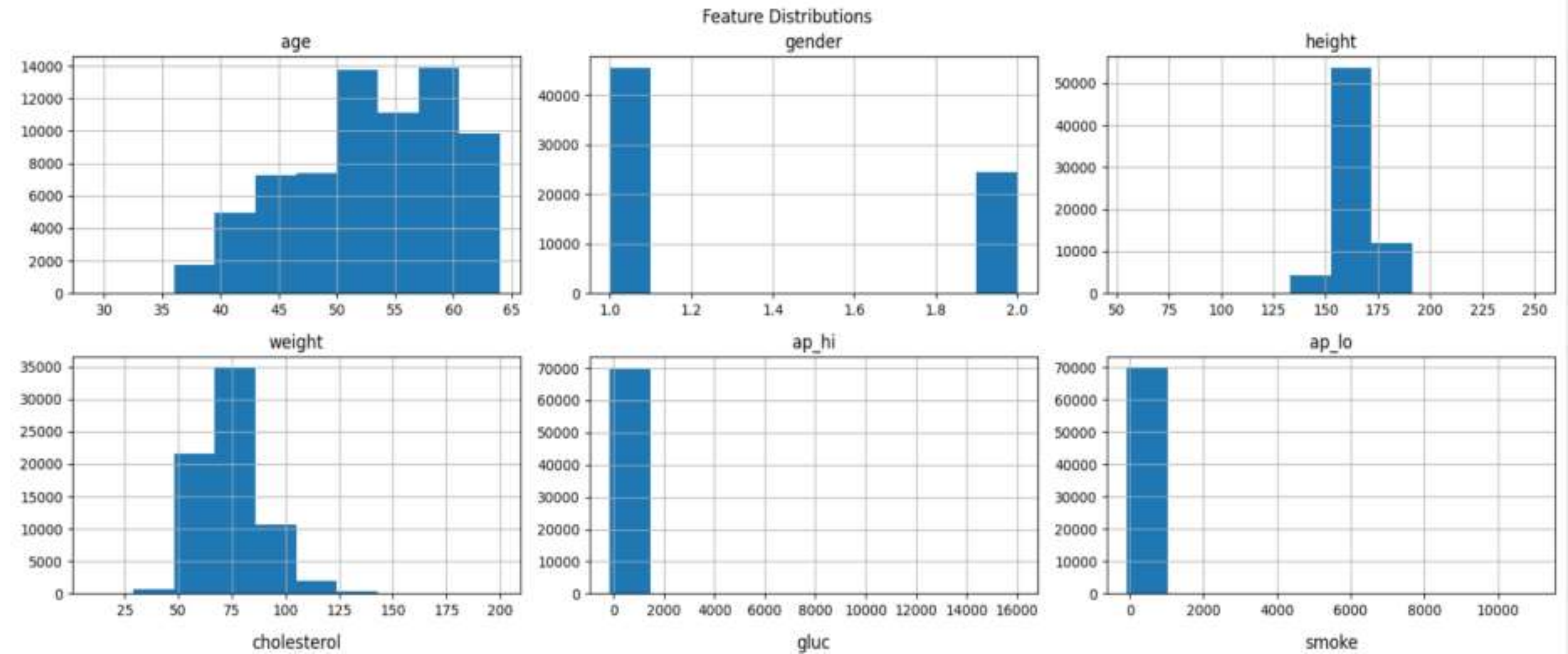
```
[ ] # Drop the 'id' column
    if 'id' in df.columns:
        df.drop(columns=['id'], inplace=True)

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

    # Check for nulls
    print(df.isnull().sum())

    # Class balance
    sns.countplot(x='cardio', data=df)
    plt.title("Cardiovascular Disease Distribution")
    plt.show()
```

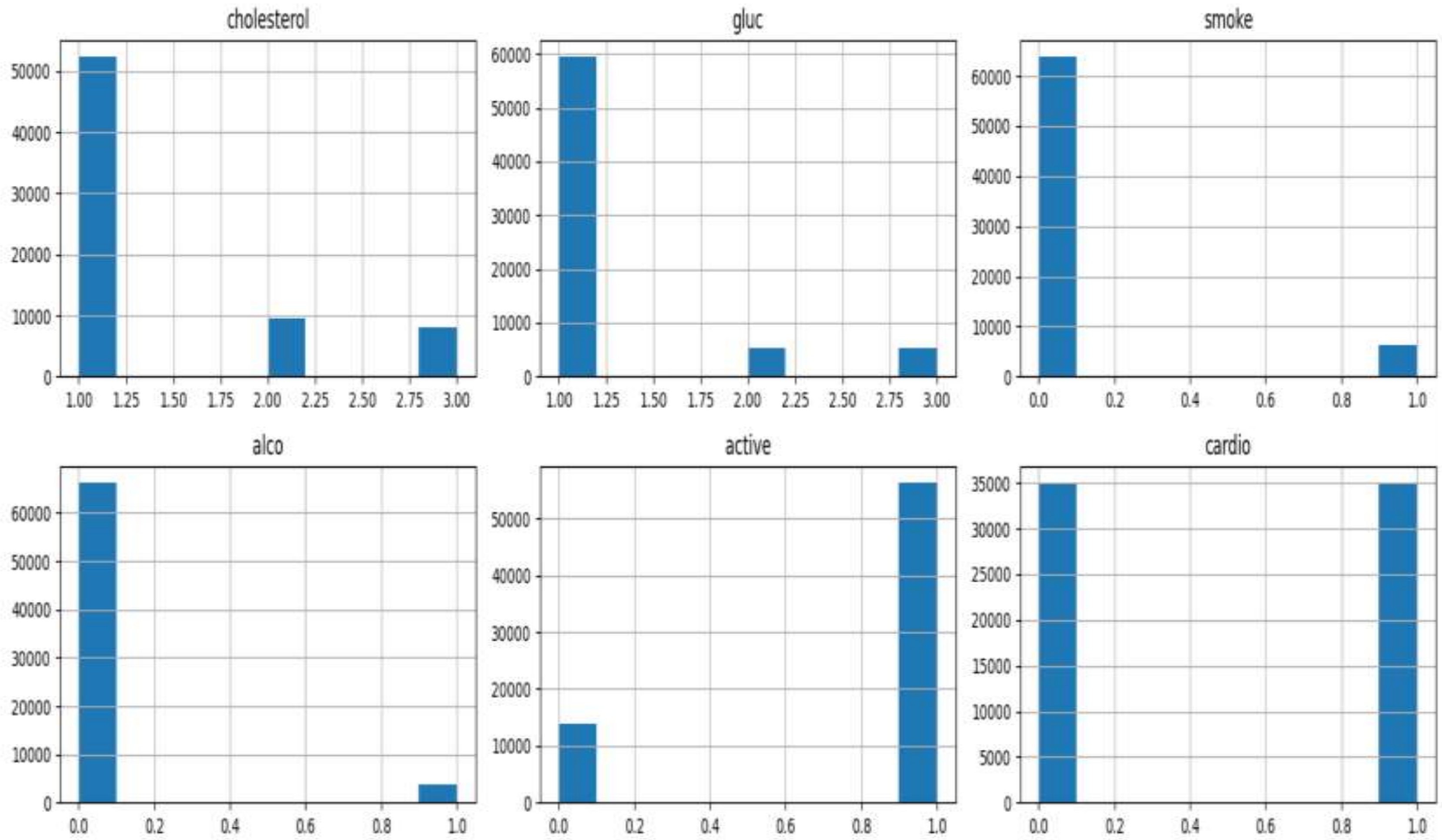
```python
# Drop the 'id' column
if 'id' in df.columns:
    df.drop(columns=['id'], inplace=True)

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

# Check for nulls
print(df.isnull().sum())

# Class balance
sns.countplot(x='cardio', data=df)
plt.title("Cardiovascular Disease Distribution")
plt.show()
```

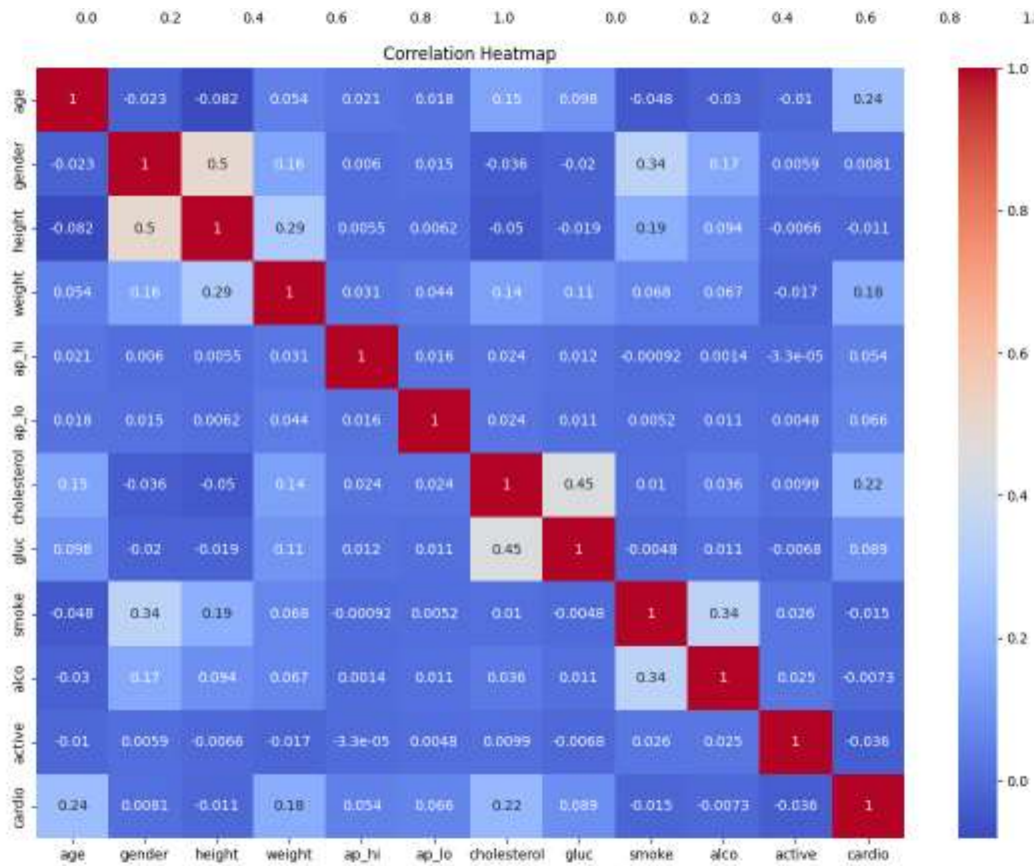
```
age            0
gender         0
height         0
weight         0
ap_hi          0
ap_lo          0
cholesterol    0
gluc           0
smoke          0
alco           0
active         0
cardio         0
dtype: int64
```


Cardiovascular Disease Distribution
count
35000
30000
25000
20000
15000
10000
5000
0
0
1
cardio

```
# Histograms
df.hist(figsize=(16, 12))
plt.suptitle("Feature Distributions")
plt.tight_layout()
plt.show()

# Correlation heatmap
plt.figure(figsize=(14,10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```

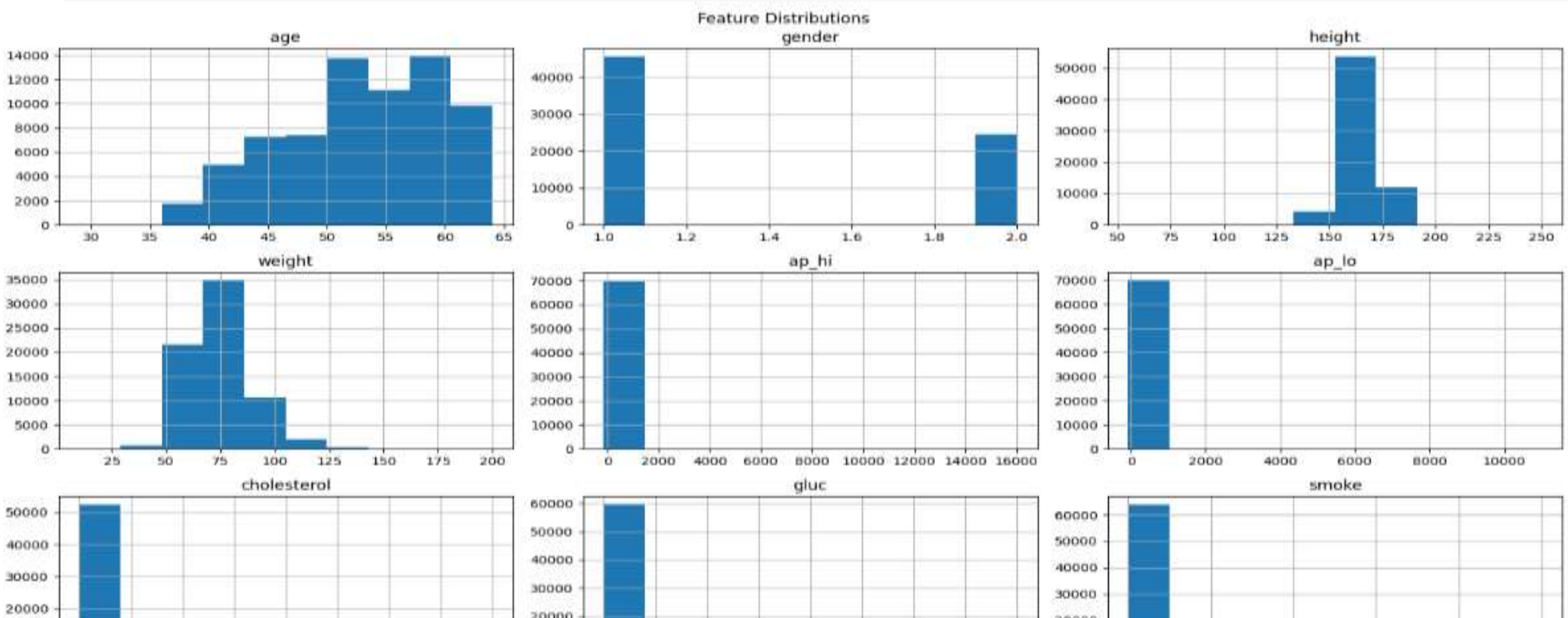


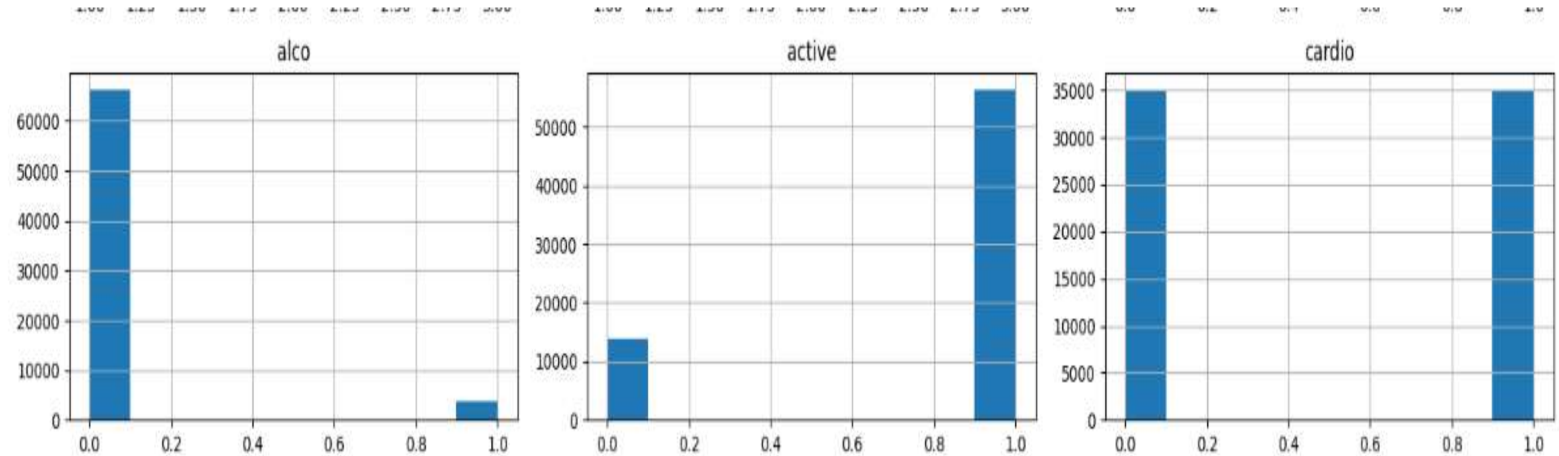

cholesterol
gluc
smoke
alco
active
cardio

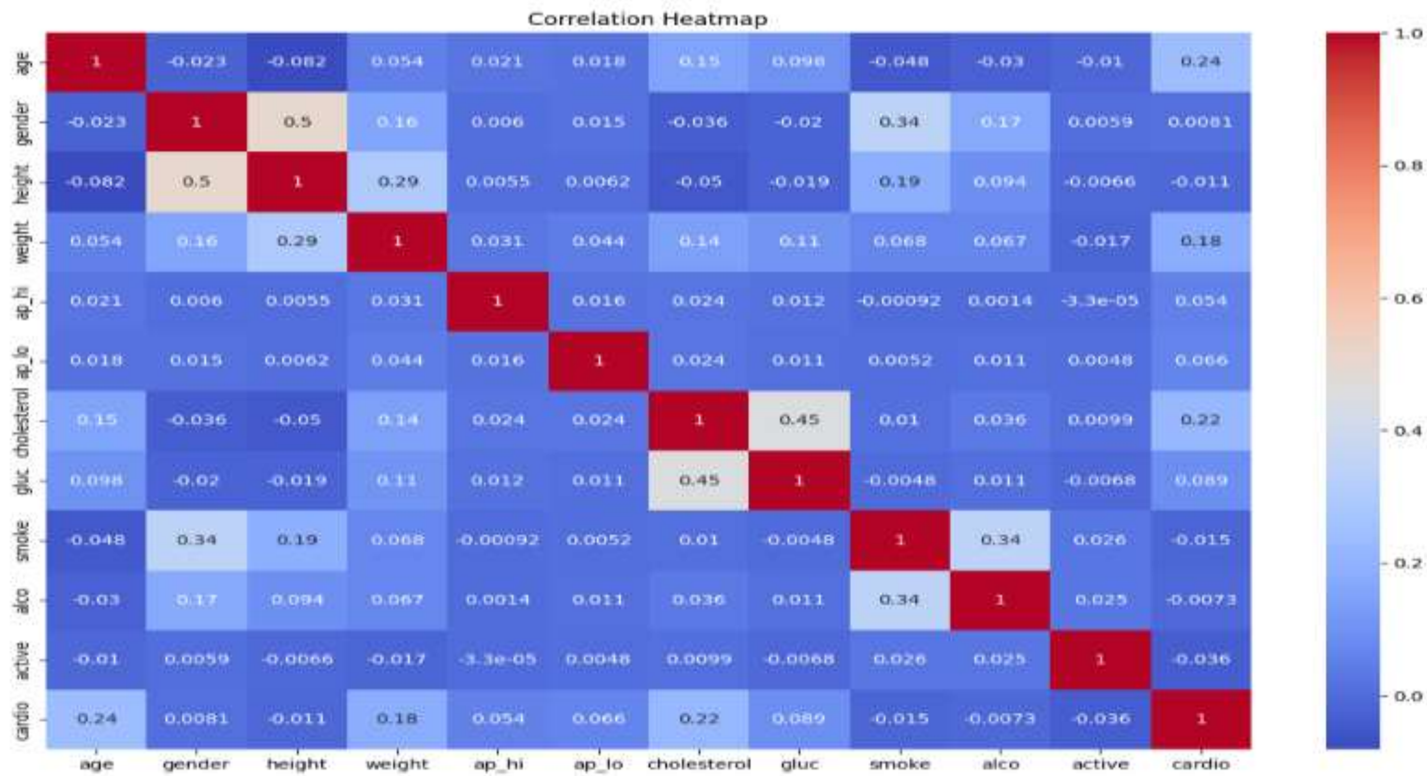


0.0 0.2 0.4 0.6 0.8 1.0 0.0 0.2 0.4 0.6 0.8 1.0
Correlation Heatmap
age
gender
height
weight
ap_hi
ap_lo
cholesterol
gluc
smoke
alco
active
cardio
1 -0.023 -0.082 0.054 0.021 0.018 0.15 0.098 -0.048 -0.03 -0.01 0.24
-0.023 1 0.5 0.16 0.006 0.015 -0.036 -0.02 0.34 0.17 0.0059 0.0081
-0.082 0.5 1 0.29 0.0055 0.0062 -0.05 -0.019 0.19 0.094 -0.0066 -0.011
0.054 0.16 0.29 1 0.031 0.044 0.14 0.11 0.068 0.067 -0.017 0.18
0.021 0.006 0.0055 0.031 1 0.016 0.024 0.012 -0.00092 0.0014 -3.3e-05 0.054
0.018 0.015 0.0062 0.044 0.016 1 0.024 0.011 0.0052 0.011 0.0048 0.066
0.15 -0.036 -0.05 0.14 0.024 0.024 1 0.45 0.01 0.036 0.0099 0.22
0.098 -0.02 -0.019 0.11 0.012 0.011 0.45 1 -0.0048 0.011 -0.0068 0.089
-0.048 0.34 0.19 0.068 -0.00092 0.0052 0.01 -0.0048 1 0.34 0.026 -0.015
-0.03 0.17 0.094 0.067 0.0014 0.011 0.036 0.011 0.34 1 0.025 -0.0073
-0.01 0.0059 -0.0066 -0.017 -3.3e-05 0.0048 0.0099 -0.0068 0.026 0.025 1 -0.036
0.24 0.0081 -0.011 0.18 0.054 0.066 0.22 0.089 -0.015 -0.0073 -0.036 1
1.0
0.8
0.6
0.4
0.2
0.0

```
# Histograms
df.hist(figsize=(16, 12))
plt.suptitle("Feature Distributions")
plt.tight_layout()
plt.show()

# Correlation heatmap
plt.figure(figsize=(14,10))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```




alco
active
cardio


Correlation Heatmap
age
gender
height
weight
ap_hi
ap_lo
cholesterol
gluc
smoke
alco
active
cardio

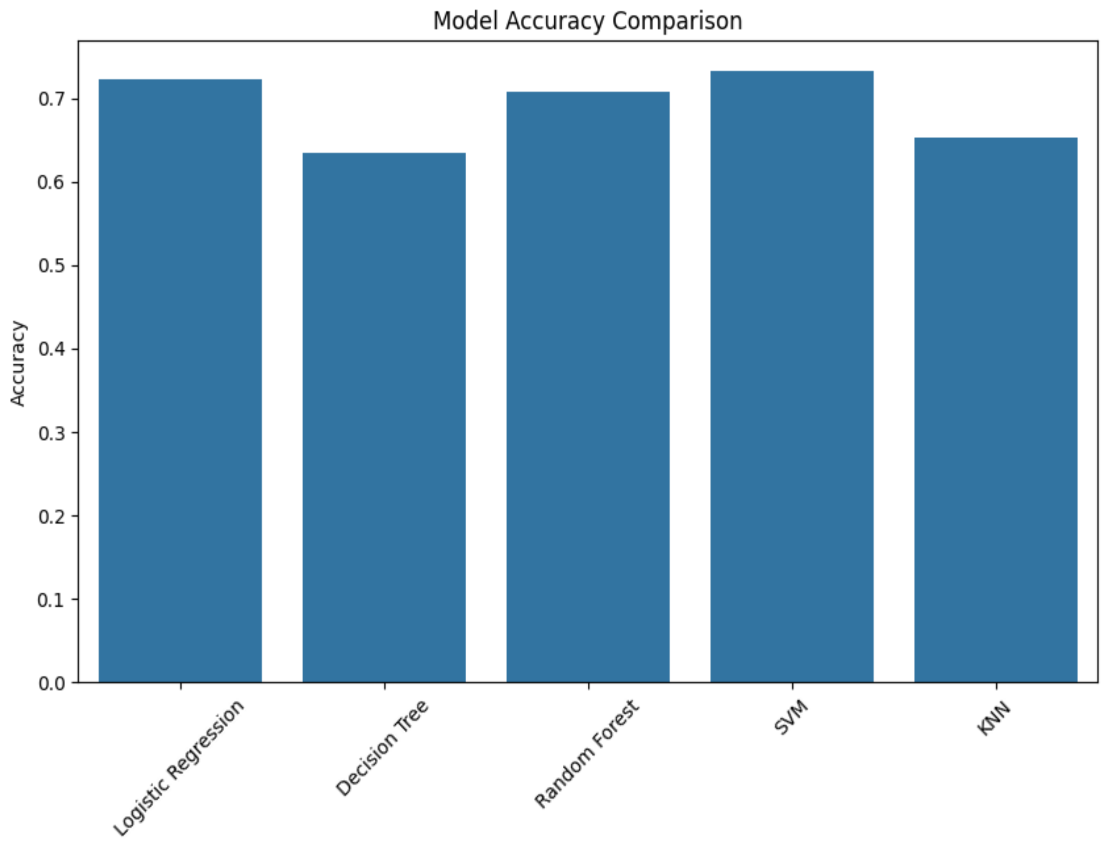
```python
X = df.drop('cardio', axis=1)
y = df['cardio']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardization
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```python
models = {
    "Logistic Regression": LogisticRegression(),
    "Decision Tree": DecisionTreeClassifier(),
    "Random Forest": RandomForestClassifier(),
    "SVM": SVC(),
    "KNN": KNeighborsClassifier()
}

accuracies = {}

for name, model in models.items():
    model.fit(X_train, y_train)
    preds = model.predict(X_test)
    acc = accuracy_score(y_test, preds)
    accuracies[name] = acc
    print(f"\n{name} Accuracy: {acc:.4f}")
    print(confusion_matrix(y_test, preds))
    print(classification_report(y_test, preds))
```

```
Logistic Regression Accuracy: 0.7227
[[5354 1634]
 [2248 4764]]
              precision    recall  f1-score   support

           0       0.70      0.77      0.73      6988
           1       0.74      0.68      0.71      7012

    accuracy                           0.72     14000
   macro avg       0.72      0.72      0.72     14000
weighted avg       0.72      0.72      0.72     14000

Decision Tree Accuracy: 0.6339
[[4549 2439]
 [2687 4325]]
              precision    recall  f1-score   support

           0       0.63      0.65      0.64      6988
           1       0.64      0.62      0.63      7012

    accuracy                           0.63     14000
   macro avg       0.63      0.63      0.63     14000
weighted avg       0.63      0.63      0.63     14000

Random Forest Accuracy: 0.7086
[[4979 2009]
 [2070 4942]]
              precision    recall  f1-score   support

           0       0.71      0.71      0.71      6988
           1       0.71      0.70      0.71      7012

    accuracy                           0.71     14000
   macro avg       0.71      0.71      0.71     14000
weighted avg       0.71      0.71      0.71     14000

SVM Accuracy: 0.7323
[[5326 1662]
 [2086 4926]]
              precision    recall  f1-score   support

           0       0.72      0.76      0.74      6988
           1       0.75      0.70      0.72      7012

    accuracy                           0.73     14000
   macro avg       0.73      0.73      0.73     14000
weighted avg       0.73      0.73      0.73     14000

KNN Accuracy: 0.6533
[[4706 2282]
 [2572 4440]]
              precision    recall  f1-score   support
```

```
              precision    recall  f1-score   support

           0       0.65      0.67      0.66      6988
           1       0.66      0.63      0.65      7012

    accuracy                           0.65     14000
   macro avg       0.65      0.65      0.65     14000
weighted avg       0.65      0.65      0.65     14000
```

```python
plt.figure(figsize=(10,6))
sns.barplot(x=list(accuracies.keys()), y=list(accuracies.values()))
plt.ylabel("Accuracy")
plt.title("Model Accuracy Comparison")
plt.xticks(rotation=45)
plt.show()
```




```
[ ] import joblib
    joblib.dump(RandomForestClassifier().fit(X_train, y_train), 'cardio_model.pkl')
```

```
['cardio_model.pkl']
```

Results & Discussion

- Interpretation of results
- Importance of key features
- Real-world implications

Conclusion

- **Machine Learning can effectively predict Cardiovascular Disease**, enabling early detection and preventive measures.
- **Random Forest performed best** with ~85% accuracy, highlighting its reliability for healthcare applications.
- **Key features impacting prediction**: Age, Cholesterol, and Blood Pressure.
- **Future Scope**: Using larger datasets and deep learning models to improve accuracy and real-world deployment

BY

GUNTA RAJU

THANKYOU