```
# Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load your dataset
# Note: Replace 'your_dataset.csv' with your actual dataset file in Google Colab
from google.colab import files
uploaded = files.upload()

# Assuming the uploaded file is a CSV
data = pd.read_csv(next(iter(uploaded)))

# Display the first few rows of the dataset
print("Data Head:")
print(data.head())

# Check for existing columns
print("\nColumns in the dataset:")
print(data.columns)

# 1. Feature Engineering: Create new features
# Example: Creating a new feature by multiplying two existing features

# Ensure 'feature1' and 'feature2' exist in the dataset
if 'feature1' in data.columns and 'feature2' in data.columns:
    # Convert columns to numeric, if necessary
    data['feature1'] = pd.to_numeric(data['feature1'], errors='coerce')
    data['feature2'] = pd.to_numeric(data['feature2'], errors='coerce')

    # Handle missing values by filling NaNs with 0
    data[['feature1', 'feature2']].fillna(0, inplace=True)

    # Create the new feature
    data['new_feature'] = data['feature1'] * data['feature2']
    print("\nNew feature created successfully.")
else:
    print("\nOne or both columns 'feature1' and 'feature2' do not exist in the dataset.")

# 2. Split the data into training and testing sets
# Replace 'target' with your actual target column
X = data.drop('target', axis=1)  # Features
y = data['target']  # Target

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

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 3. Feature Selection using PCA (Principal Component Analysis)
pca = PCA(n_components=0.95)  # Retain 95% of the variance
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

print(f'\nOriginal number of features: {X_train.shape[1]}')
print(f'Reduced number of features: {X_train_pca.shape[1]}')

# 4. Train a model using RandomForestClassifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train_pca, y_train)

# 5. Evaluate the model
y_pred = model.predict(X_test_pca)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)

print(f'\nModel Accuracy: {accuracy}')
print(f'\nClassification Report:\n{report}')

# 6. Feature Importance (Optional)
importances = model.feature_importances_
indices = np.argsort(importances)[::-1]

# Print the feature ranking
print("\nFeature ranking (from PCA components):")
```

```
for f in range(X_train_pca.shape[1]):
    print(f"{f + 1}. feature {indices[f]} ({importances[indices[f]]})")

# 7. Optimization: Optionally, you can retrain the model with only the top features
```

Choose files heart.csv

- **heart.csv**(text/csv) - 38114 bytes, last modified: 22/08/2024 - 100% done

```
Saving heart.csv to heart (1).csv
Data Head:
   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

Columns in the dataset:
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')

One or both columns 'feature1' and 'feature2' do not exist in the dataset.

Original number of features: 13
Reduced number of features: 12

Model Accuracy: 0.9853658536585366

Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       102
           1       1.00      0.97      0.99       103

    accuracy                           0.99       205
   macro avg       0.99      0.99      0.99       205
weighted avg       0.99      0.99      0.99       205


Feature ranking (from PCA components):
1. feature 0 (0.3922925556122199)
2. feature 2 (0.0835853197473908)
3. feature 6 (0.07208844131461817)
4. feature 1 (0.07034158694419751)
5. feature 5 (0.0589117119519183)
6. feature 9 (0.05262984064580328)
7. feature 11 (0.04942974672373123)
8. feature 3 (0.04906002395874745)
9. feature 8 (0.048173290196371905)
10. feature 10 (0.04325387039479891)
11. feature 4 (0.042561956778836715)
```