```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import TruncatedSVD

from mpl_toolkits.mplot3d import Axes3D

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

data=pd.read_csv("C:\\Users\\SHARIB\\Downloads\\heart.csv")

data.head()


data.info()

features = data.drop(columns=['target'])

target = data['target']


scaler = StandardScaler()

scaled_features = scaler.fit_transform(features)


svd = TruncatedSVD(n_components=5, random_state=42)

svd_features = svd.fit_transform(scaled_features)

explained_variance = svd.explained_variance_ratio_

svd_features_df = pd.DataFrame(svd_features, columns=[f'SVD_Component_{i+1}' for i in
range(svd_features.shape[1])])

explained_variance, svd_features_df.head()


svd = TruncatedSVD(n_components=5)


svd.fit(features)
```

```
singular_values = svd.singular_values_

plt.figure(figsize=(12, 5))

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
sns.barplot(x=np.arange(1, len(singular_values) + 1), y=singular_values)
plt.xlabel('Component')
plt.ylabel('Singular Value')
plt.title('Singular Values of Components')

plt.subplot(1, 2, 2)
sns.barplot(x=np.arange(1, len(explained_variance) + 1), y=explained_variance)
plt.xlabel('Component')
plt.ylabel('Explained Variance Ratio')
plt.title('Explained Variance Ratio of Components')

cumulative_explained_ratios = np.cumsum(explained_variance)
plt.figure(figsize=(6, 4))
sns.lineplot(x=np.arange(1, len(cumulative_explained_variance) + 1),
        y=cumulative_explained_variance, marker='o')
plt.xlabel('Number of Components')
plt.ylabel('Cumulative Explained Variance')
plt.title('Cumulative Explained Variance by Components')
plt.grid(True)

plt.tight_layout()
plt.show()

correlations = pd.DataFrame(np.dot(scaled_features.T, svd_features) / (len(scaled_features) - 1),
                index=features.columns,
                    columns=[f'SVD_Component_{i+1}' for i in range(svd_features.shape[1])])

correlations

plt.figure(figsize=(10, 8))
sns.scatterplot(x=svd_features[:, 0], y=svd_features[:, 1], hue=target, palette="viridis", edgecolor='k')
plt.xlabel('SVD Component 1')
plt.ylabel('SVD Component 2')
plt.title('Data Visualization in SVD Component Space (Colored by Heart Disease Target)')
plt.legend(title="Heart Disease Target", labels=['No Disease', 'Disease'])
plt.show()

explained_variance = svd.explained_variance_ratio_
plt.figure(figsize=(8,6))
sns.barplot(x=[f'Component {i+1}' for i in range(len(explained_variance))],
        y=explained_variance * 100)
plt.ylabel('Explained Variance (%)')
plt.xlabel('SVD Components')
plt.title('Scree Plot')
plt.show()

svd_plot_df = svd_features_df.copy()
svd_plot_df['target'] = target
plt.figure(figsize=(10,8))
sns.scatterplot(data=svd_plot_df,
            x='SVD_Component_1',
            y='SVD_Component_2',
            hue='target',
            palette=['blue', 'red'],
            alpha=0.6)
plt.title('SVD Components 1 vs 2 Colored by Heart Disease Diagnosis')
plt.xlabel('SVD Component 1')

plt.ylabel('SVD Component 2')

plt.legend(title='Heart Disease', labels=['No Disease', 'Disease'])

plt.show()


fig = plt.figure(figsize=(12,10))

ax = fig.add_subplot(111, projection='3d')


colors = svd_plot_df['target'].map({0: 'blue', 1: 'red'})


ax.scatter(svd_plot_df['SVD_Component_1'],

        svd_plot_df['SVD_Component_2'],

        svd_plot_df['SVD_Component_3'],

        c=colors, alpha=0.6)


ax.set_title('3D Scatter Plot of SVD Components Colored by Heart Disease Diagnosis')

ax.set_xlabel('SVD Component 1')

ax.set_ylabel('SVD Component 2')

ax.set_zlabel('SVD Component 3')


from matplotlib.lines import Line2D

legend_elements = [Line2D([0], [0], marker='o', color='w', label='No Disease',

                markerfacecolor='blue', markersize=10),

            Line2D([0], [0], marker='o', color='w', label='Disease',

                markerfacecolor='red', markersize=10)]

ax.legend(handles=legend_elements, title='Heart Disease')


plt.show()


melted_df = svd_plot_df.melt(id_vars='target',
                        value_vars=[f'SVD_Component_{i+1}' for i in range(svd.n_components)], # Use
svd.n_components
                        var_name='SVD_Component',
                        value_name='Value')
plt.figure(figsize=(14,10))
sns.boxplot(x='SVD_Component', y='Value', hue='target', data=melted_df, palette='Set2')
plt.title('Distribution of SVD Components by Heart Disease Diagnosis')
plt.xlabel('SVD Components')
plt.ylabel('Component Values')
plt.legend(title='Heart Disease', labels=['No Disease', 'Disease'])
plt.show()

X_train, X_test, y_train, y_test = train_test_split(svd_features_df, target,
                                            test_size=0.2,
                                            random_state=42,
                                            stratify=target)

# Initialize and train the model
model = LogisticRegression(max_iter=1000, random_state=42)
model.fit(X_train, y_train)

# Predictions
y_pred = model.predict(X_test)

# Confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix as heatmap
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False,
          xticklabels=["No Disease", "Disease"],
          yticklabels=["No Disease", "Disease"])
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.title("Confusion Matrix of Heart Disease Classification")
plt.show()


report = classification_report(y_test, y_pred, target_names=["No Disease", "Disease"],
output_dict=True)


report_df = pd.DataFrame(report).transpose()


report_df = report_df.round(2)


report_df = report_df.rename(columns={
   "precision": "Precision",
   "recall": "Recall",
   "f1-score": "F1-Score",
   "support": "Support"
})
print("Classification Report:")
print(report_df)
print(f"\nOverall Accuracy:{accuracy_score(y_test, y_pred):.2f}")
```