```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder, OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA
from sklearn.metrics import accuracy_score
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.datasets import load_iris, load_diabetes
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
import matplotlib.pyplot as plt


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

Choose Files heart.csv

- **heart.csv**(text/csv) - 36840 bytes, last modified: 5/5/2025 - 100% done

Saving heart.csv to heart.csv

```
df = pd.read_csv('heart.csv')
df
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0
...	...	...	...	...	...	...	...	...	...	...	...	...
913	45	M	TA	110	264	0	Normal	132	N	1.2	Flat	1
914	68	M	ASY	144	193	1	Normal	141	N	3.4	Flat	1
915	57	M	ASY	130	131	0	Normal	115	Y	1.2	Flat	1
916	57	F	ATA	130	236	0	LVH	174	N	0.0	Flat	1
917	38	M	NAP	138	175	0	Normal	173	N	0.0	Up	0

918 rows × 12 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
label_encoder = LabelEncoder()
df['Sex'] = label_encoder.fit_transform(df['Sex'])
df['FastingBS'] = label_encoder.fit_transform(df['FastingBS'])
df['ExerciseAngina'] = label_encoder.fit_transform(df['ExerciseAngina'])
df['HeartDisease'] = label_encoder.fit_transform(df['HeartDisease'])
print(df.head())
```

```
   Age  Sex ChestPainType  RestingBP  Cholesterol  FastingBS RestingECG  \
0   40    1           ATA        140          289          0     Normal
1   49    0           NAP        160          180          0     Normal
```

```
2   37    1           ATA        130          283          0         ST
3   48    0           ASY        138          214          0     Normal
4   54    1           NAP        150          195          0     Normal

   MaxHR  ExerciseAngina  Oldpeak ST_Slope  HeartDisease
0    172               0      0.0       Up             0
1    156               0      1.0     Flat             1
2     98               0      0.0       Up             0
3    108               1      1.5     Flat             1
4    122               0      0.0       Up             0
```

```
df = pd.get_dummies(df, columns=['ChestPainType', 'RestingECG', 'ST_Slope'], drop_first=True)
print(df.head())
```

```
   Age  Sex  RestingBP  Cholesterol  FastingBS  MaxHR  ExerciseAngina  \
0   40    1        140          289          0    172               0
1   49    0        160          180          0    156               0
2   37    1        130          283          0     98               0
3   48    0        138          214          0    108               1
4   54    1        150          195          0    122               0

   Oldpeak  HeartDisease  ChestPainType_ATA  ChestPainType_NAP  \
0      0.0             0               True              False
1      1.0             1              False               True
2      0.0             0               True              False
3      1.5             1              False              False
4      0.0             0              False               True

   ChestPainType_TA  RestingECG_Normal  RestingECG_ST  ST_Slope_Flat  \
0             False               True          False          False
1             False               True          False           True
2             False              False           True          False
3             False               True          False           True
4             False               True          False          False

   ST_Slope_Up
0         True
1        False
2         True
3        False
4         True
```

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

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
print(X_scaled[:5])
```

```
[[-1.4331398   0.51595242  0.41090889  0.82507026 -0.55134134  1.38292822
  -0.8235563  -0.83243239  2.07517671 -0.53283777 -0.22967867  0.81427482
  -0.49044933 -1.00218103  1.15067399]
 [-0.47848359 -1.93816322  1.49175234 -0.17196105 -0.55134134  0.75415714
  -0.8235563   0.10566353 -0.48188667  1.87674385 -0.22967867  0.81427482
  -0.49044933  0.99782372 -0.86905588]
 [-1.75135854  0.51595242 -0.12951283  0.7701878  -0.55134134 -1.52513802
  -0.8235563  -0.83243239  2.07517671 -0.53283777 -0.22967867 -1.22808661
   2.03894663 -1.00218103  1.15067399]
 [-0.5845565  -1.93816322  0.30282455  0.13903954 -0.55134134 -1.13215609
   1.21424608  0.57471149 -0.48188667 -0.53283777 -0.22967867  0.81427482
  -0.49044933  0.99782372 -0.86905588]
 [ 0.05188098  0.51595242  0.95133062 -0.0347549  -0.55134134 -0.5819814
  -0.8235563  -0.83243239 -0.48188667  1.87674385 -0.22967867  0.81427482
  -0.49044933 -1.00218103  1.15067399]]
```

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

```
svm_model = SVC()
svm_model.fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print(f"SVM Accuracy: {accuracy_svm}")
```

```
SVM Accuracy: 0.875
```

```
log_reg_model = LogisticRegression()
log_reg_model.fit(X_train, y_train)
y_pred_log_reg = log_reg_model.predict(X_test)
```

```
accuracy_log_reg = accuracy_score(y_test, y_pred_log_reg)
print(f"Logistic Regression Accuracy: {accuracy_log_reg}")
```

```
Logistic Regression Accuracy: 0.8532608695652174
```

```
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f"Random Forest Accuracy: {accuracy_rf}")
```

```
Random Forest Accuracy: 0.8695652173913043
```

```
pca = PCA(n_components=5)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
print(f"Explained variance ratio by PCA: {pca.explained_variance_ratio_}")
```

```
Explained variance ratio by PCA: [0.23613165 0.10817164 0.09475436 0.08335247 0.07379415]
```

```
svm_model_pca = SVC()
svm_model_pca.fit(X_train_pca, y_train)
y_pred_svm_pca = svm_model_pca.predict(X_test_pca)
accuracy_svm_pca = accuracy_score(y_test, y_pred_svm_pca)
print(f"SVM Accuracy with PCA: {accuracy_svm_pca}")
```

```
SVM Accuracy with PCA: 0.8369565217391305
```

```
log_reg_model_pca = LogisticRegression()
log_reg_model_pca.fit(X_train_pca, y_train)
y_pred_log_reg_pca = log_reg_model_pca.predict(X_test_pca)
accuracy_log_reg_pca = accuracy_score(y_test, y_pred_log_reg_pca)
print(f"Logistic Regression Accuracy with PCA: {accuracy_log_reg_pca}")
```

```
Logistic Regression Accuracy with PCA: 0.8315217391304348
```

```
rf_model_pca = RandomForestClassifier()
rf_model_pca.fit(X_train_pca, y_train)
y_pred_rf_pca = rf_model_pca.predict(X_test_pca)
accuracy_rf_pca = accuracy_score(y_test, y_pred_rf_pca)
print(f"Random Forest Accuracy with PCA: {accuracy_rf_pca}")
```

```
Random Forest Accuracy with PCA: 0.8532608695652174
```

```
print("\nAccuracy Comparison:")
print(f"SVM Accuracy: {accuracy_svm}")
print(f"Logistic Regression Accuracy: {accuracy_log_reg}")
print(f"Random Forest Accuracy: {accuracy_rf}")
print(f"SVM Accuracy with PCA: {accuracy_svm_pca}")
print(f"Logistic Regression Accuracy with PCA: {accuracy_log_reg_pca}")
print(f"Random Forest Accuracy with PCA: {accuracy_rf_pca}")
```

```
Accuracy Comparison:
SVM Accuracy: 0.875
Logistic Regression Accuracy: 0.8532608695652174
Random Forest Accuracy: 0.8695652173913043
SVM Accuracy with PCA: 0.8369565217391305
Logistic Regression Accuracy with PCA: 0.8315217391304348
Random Forest Accuracy with PCA: 0.8532608695652174
```

```
Start coding or generate with AI.
```