DSML - All 25 Assignments (With Graphs)

1. Medical Diagnosis Using SVM

Dataset suggestion: **cancer.csv** — Use image-derived features (diagnosis column M/B)

```
# Medical Diagnosis Using SVM - with graphs
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.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc

df = pd.read_csv("cancer.csv")
y = df['TARGET']  # replace
X = df.drop(['TARGET'], axis=1)

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 = scaler.fit_transform(X_train); X_test = scaler.transform(X_test)

model = SVC(kernel='linear', probability=True); model.fit(X_train, y_train)
y_pred = model.predict(X_test); y_prob = model.predict_proba(X_test)[:,1]

print('Accuracy:', accuracy_score(y_test, y_pred))
print('\nClassification Report:\n', classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d'); plt.title('Confusion Matrix'); plt.show()

fpr, tpr, _ = roc_curve(y_test, y_prob); roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label='AUC=%.3f'%roc_auc); plt.plot([0,1],[0,1],'k--'); plt.legend(); plt.show()
```

2. Loan Default Prediction with Decision Trees

Dataset suggestion: **loan.csv** — Predict loan default (Loan_Status)

```
# Loan Default Prediction with Decision Trees - with graphs
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.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc

df = pd.read_csv("loan.csv")
y = df['TARGET']  # replace
X = df.drop(['TARGET'], axis=1)

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 = scaler.fit_transform(X_train); X_test = scaler.transform(X_test)

model = SVC(kernel='linear', probability=True); model.fit(X_train, y_train)
y_pred = model.predict(X_test); y_prob = model.predict_proba(X_test)[:,1]

print('Accuracy:', accuracy_score(y_test, y_pred))
print('\nClassification Report:\n', classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d'); plt.title('Confusion Matrix'); plt.show()

fpr, tpr, _ = roc_curve(y_test, y_prob); roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label='AUC=%.3f'%roc_auc); plt.plot([0,1],[0,1],'k--'); plt.legend(); plt.show()
```

3. Disease Classification with KNN

Dataset suggestion: **diabetes.csv** — Predict diabetes (Outcome)

```
# Disease Classification with KNN - with graphs
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.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc

df = pd.read_csv("diabetes.csv")
y = df['TARGET']  # replace
X = df.drop(['TARGET'], axis=1)

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 = scaler.fit_transform(X_train); X_test = scaler.transform(X_test)

model = SVC(kernel='linear', probability=True); model.fit(X_train, y_train)
y_pred = model.predict(X_test); y_prob = model.predict_proba(X_test)[:,1]

print('Accuracy:', accuracy_score(y_test, y_pred))
print('\nClassification Report:\n', classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d'); plt.title('Confusion Matrix'); plt.show()

fpr, tpr, _ = roc_curve(y_test, y_prob); roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label='AUC=%.3f'%roc_auc); plt.plot([0,1],[0,1],'k--'); plt.legend(); plt.show()
```

4. Email Spam Detection Using Naive Bayes

Dataset suggestion: **spam.csv** — Email text features or word counts

```
# Email Spam Detection Using Naive Bayes - with graphs
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.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc

df = pd.read_csv("spam.csv")
y = df['TARGET']  # replace
X = df.drop(['TARGET'], axis=1)

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 = scaler.fit_transform(X_train); X_test = scaler.transform(X_test)

model = SVC(kernel='linear', probability=True); model.fit(X_train, y_train)
y_pred = model.predict(X_test); y_prob = model.predict_proba(X_test)[:,1]

print('Accuracy:', accuracy_score(y_test, y_pred))
print('\nClassification Report:\n', classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d'); plt.title('Confusion Matrix'); plt.show()

fpr, tpr, _ = roc_curve(y_test, y_prob); roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label='AUC=%.3f'%roc_auc); plt.plot([0,1],[0,1],'k--'); plt.legend(); plt.show()
```

5. Customer Sentiment Analysis with SVM

Dataset suggestion: **reviews.csv** — Use TF-IDF features from reviews

```
# Customer Sentiment Analysis with SVM - with graphs
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.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc

df = pd.read_csv("reviews.csv")
y = df['TARGET']  # replace
X = df.drop(['TARGET'], axis=1)

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 = scaler.fit_transform(X_train); X_test = scaler.transform(X_test)

model = SVC(kernel='linear', probability=True); model.fit(X_train, y_train)
y_pred = model.predict(X_test); y_prob = model.predict_proba(X_test)[:,1]

print('Accuracy:', accuracy_score(y_test, y_pred))
print('\nClassification Report:\n', classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d'); plt.title('Confusion Matrix'); plt.show()

fpr, tpr, _ = roc_curve(y_test, y_prob); roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label='AUC=%.3f'%roc_auc); plt.plot([0,1],[0,1],'k--'); plt.legend(); plt.show()
```

6. House Price Prediction Using Linear Regression

Dataset suggestion: **house_prices.csv** — Predict price (continuous)

```
# House Price Prediction Using Linear Regression - with graphs (regression)
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_csv("house_prices.csv")
y = df['TARGET']; X = df.drop(['TARGET'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression(); model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print('RMSE:', mean_squared_error(y_test, y_pred, squared=False)); print('R2:', r2_score(y_test, y_pred))

plt.scatter(y_test, y_pred); plt.xlabel('Actual'); plt.ylabel('Predicted'); plt.title('Actual vs Predicted'); p
```

7. Churn Probability with Logistic Regression

Dataset suggestion: **churn.csv** — Predict churn (binary)

```
# Churn Probability with Logistic Regression - with graphs
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.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc

df = pd.read_csv("churn.csv")
y = df['TARGET']  # replace
```

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

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 = scaler.fit_transform(X_train); X_test = scaler.transform(X_test)

model = SVC(kernel='linear', probability=True); model.fit(X_train, y_train)
y_pred = model.predict(X_test); y_prob = model.predict_proba(X_test)[:,1]

print('Accuracy:', accuracy_score(y_test, y_pred))
print('\nClassification Report:\n', classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d'); plt.title('Confusion Matrix'); plt.show()

fpr, tpr, _ = roc_curve(y_test, y_prob); roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label='AUC=%.3f'%roc_auc); plt.plot([0,1],[0,1],'k--'); plt.legend(); plt.show()
```

8. Stock Price Forecasting with Least Squares

Dataset suggestion: **stock.csv** — Use historical prices/time series

```
# Stock Price Forecasting with Least Squares - with graphs (regression)
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_csv("stock.csv")
y = df['TARGET']; X = df.drop(['TARGET'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression(); model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print('RMSE:', mean_squared_error(y_test, y_pred, squared=False)); print('R2:', r2_score(y_test, y_pred))

plt.scatter(y_test, y_pred); plt.xlabel('Actual'); plt.ylabel('Predicted'); plt.title('Actual vs Predicted'); p
```

9. Crop Yield Estimation Using Linear Regression

Dataset suggestion: **crop.csv** — Predict yield from weather/soil

```
# Crop Yield Estimation Using Linear Regression - with graphs (regression)
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_csv("crop.csv")
y = df['TARGET']; X = df.drop(['TARGET'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression(); model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print('RMSE:', mean_squared_error(y_test, y_pred, squared=False)); print('R2:', r2_score(y_test, y_pred))

plt.scatter(y_test, y_pred); plt.xlabel('Actual'); plt.ylabel('Predicted'); plt.title('Actual vs Predicted'); p
```

10. Heart Disease Risk Prediction with Logistic Regression

Dataset suggestion: **heart.csv** — Predict heart disease presence

```
# Heart Disease Risk Prediction with Logistic Regression - with graphs
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.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc

df = pd.read_csv("heart.csv")
y = df['TARGET']  # replace
X = df.drop(['TARGET'], axis=1)

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 = scaler.fit_transform(X_train); X_test = scaler.transform(X_test)

model = SVC(kernel='linear', probability=True); model.fit(X_train, y_train)
y_pred = model.predict(X_test); y_prob = model.predict_proba(X_test)[:,1]

print('Accuracy:', accuracy_score(y_test, y_pred))
print('\nClassification Report:\n', classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d'); plt.title('Confusion Matrix'); plt.show()

fpr, tpr, _ = roc_curve(y_test, y_prob); roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label='AUC=%.3f'%roc_auc); plt.plot([0,1],[0,1],'k--'); plt.legend(); plt.show()
```

11. Fraud Detection in Financial Transactions

Dataset suggestion: **credit_card_fraud.csv** — Detect fraud (binary)

```
# Fraud Detection in Financial Transactions - with graphs
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.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc

df = pd.read_csv("credit_card_fraud.csv")
y = df['TARGET']  # replace
```

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

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 = scaler.fit_transform(X_train); X_test = scaler.transform(X_test)

model = SVC(kernel='linear', probability=True); model.fit(X_train, y_train)
y_pred = model.predict(X_test); y_prob = model.predict_proba(X_test)[:,1]

print('Accuracy:', accuracy_score(y_test, y_pred))
print('\nClassification Report:\n', classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d'); plt.title('Confusion Matrix'); plt.show()

fpr, tpr, _ = roc_curve(y_test, y_prob); roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label='AUC=%.3f'%roc_auc); plt.plot([0,1],[0,1],'k--'); plt.legend(); plt.show()
```

12. Employee Attrition Prediction

Dataset suggestion: **HR_attrition.csv** — Predict attrition (Yes/No)

```
# Employee Attrition Prediction - with graphs
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.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc

df = pd.read_csv("HR_attrition.csv")
y = df['TARGET']  # replace
X = df.drop(['TARGET'], axis=1)

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 = scaler.fit_transform(X_train); X_test = scaler.transform(X_test)

model = SVC(kernel='linear', probability=True); model.fit(X_train, y_train)
y_pred = model.predict(X_test); y_prob = model.predict_proba(X_test)[:,1]

print('Accuracy:', accuracy_score(y_test, y_pred))
print('\nClassification Report:\n', classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d'); plt.title('Confusion Matrix'); plt.show()

fpr, tpr, _ = roc_curve(y_test, y_prob); roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label='AUC=%.3f'%roc_auc); plt.plot([0,1],[0,1],'k--'); plt.legend(); plt.show()
```

13. Air Quality Classification

Dataset suggestion: **air_quality.csv** — Classify AQI levels

```
# Air Quality Classification - with graphs
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.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc

df = pd.read_csv("air_quality.csv")
y = df['TARGET']  # replace
X = df.drop(['TARGET'], axis=1)

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 = scaler.fit_transform(X_train); X_test = scaler.transform(X_test)

model = SVC(kernel='linear', probability=True); model.fit(X_train, y_train)
y_pred = model.predict(X_test); y_prob = model.predict_proba(X_test)[:,1]

print('Accuracy:', accuracy_score(y_test, y_pred))
print('\nClassification Report:\n', classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d'); plt.title('Confusion Matrix'); plt.show()

fpr, tpr, _ = roc_curve(y_test, y_prob); roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label='AUC=%.3f'%roc_auc); plt.plot([0,1],[0,1],'k--'); plt.legend(); plt.show()
```

14. Product Recommendation System

Dataset suggestion: **ratings.csv** — Predict user preference or rating

```
# Graph template not available
```

15. Customer Segmentation for Marketing

Dataset suggestion: **Mall_Customers.csv** — Segment by spending behavior

```
# Customer Segmentation for Marketing - with graphs (KMeans)
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

df = pd.read_csv("Mall_Customers.csv")
X = df.select_dtypes(include=['float','int']).dropna()
scaler = StandardScaler(); X_scaled = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=3, random_state=42); kmeans.fit(X_scaled)
pca = PCA(n_components=2); X2 = pca.fit_transform(X_scaled)
plt.scatter(X2[:,0], X2[:,1], c=kmeans.labels_, cmap='viridis'); plt.title('KMeans clusters (PCA view)'); plt.s
```

16. Document Clustering for News Articles

Dataset suggestion: **news.csv** — Cluster articles by topics

```
# Document Clustering for News Articles - with graphs (KMeans)
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.decomposition import PCA

df = pd.read_csv("news.csv")
X = df.select_dtypes(include=['float','int']).dropna()
scaler = StandardScaler(); X_scaled = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=3, random_state=42); kmeans.fit(X_scaled)
pca = PCA(n_components=2); X2 = pca.fit_transform(X_scaled)
plt.scatter(X2[:,0], X2[:,1], c=kmeans.labels_, cmap='viridis'); plt.title('KMeans clusters (PCA view)'); plt.s
```

17. Image Compression Using K-Means

Dataset suggestion: **image.jpg (use cv2 to read)** — Compress image by color clustering

```
# Image Compression Using K-Means - with graphs (KMeans)
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

df = pd.read_csv("image.jpg (use cv2 to read)")
X = df.select_dtypes(include=['float','int']).dropna()
scaler = StandardScaler(); X_scaled = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=3, random_state=42); kmeans.fit(X_scaled)
pca = PCA(n_components=2); X2 = pca.fit_transform(X_scaled)
plt.scatter(X2[:,0], X2[:,1], c=kmeans.labels_, cmap='viridis'); plt.title('KMeans clusters (PCA view)'); plt.s
```

18. Traffic Pattern Analysis

Dataset suggestion: **gps.csv** — Cluster GPS coordinates/time

```
# Traffic Pattern Analysis - with graphs (KMeans)
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

df = pd.read_csv("gps.csv")
X = df.select_dtypes(include=['float','int']).dropna()
scaler = StandardScaler(); X_scaled = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=3, random_state=42); kmeans.fit(X_scaled)
pca = PCA(n_components=2); X2 = pca.fit_transform(X_scaled)
plt.scatter(X2[:,0], X2[:,1], c=kmeans.labels_, cmap='viridis'); plt.title('KMeans clusters (PCA view)'); plt.s
```

19. Market Basket Analysis for Retail

Dataset suggestion: **groceries.csv (transactions)** — Analyze itemsets

```
# Market Basket Analysis for Retail - with outputs (association rules)
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

df = pd.read_csv('groceries.csv (transactions)')
frequent_itemsets = apriori(df, min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric='confidence', min_threshold=0.5)
print(rules.sort_values('lift', ascending=False).head())
```

20. Web Navigation Pattern Mining

Dataset suggestion: **web_logs.csv** — User click-path transactions

```
# Web Navigation Pattern Mining - with outputs (association rules)
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

df = pd.read_csv('web_logs.csv')
frequent_itemsets = apriori(df, min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric='confidence', min_threshold=0.5)
print(rules.sort_values('lift', ascending=False).head())
```

21. Medical Prescription Pattern Discovery

Dataset suggestion: **prescriptions.csv** — Find co-prescribed drugs

```
# Medical Prescription Pattern Discovery - with outputs (association rules)
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

df = pd.read_csv('prescriptions.csv')
frequent_itemsets = apriori(df, min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric='confidence', min_threshold=0.5)
print(rules.sort_values('lift', ascending=False).head())
```

22. Online Course Recommendation

Dataset suggestion: **enrollments.csv** — Find course associations

```
# Online Course Recommendation - with outputs (association rules)
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

df = pd.read_csv('enrollments.csv')
frequent_itemsets = apriori(df, min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric='confidence', min_threshold=0.5)
print(rules.sort_values('lift', ascending=False).head())
```

23. Retail Loss Prevention

Dataset suggestion: **sales.csv** — Find suspicious combinations

```
# Retail Loss Prevention - with outputs (association rules)
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

df = pd.read_csv('sales.csv')
frequent_itemsets = apriori(df, min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric='confidence', min_threshold=0.5)
print(rules.sort_values('lift', ascending=False).head())
```

24. Predictive Maintenance Using Classification and Regression

Dataset suggestion: **sensor_readings.csv** — Predict failures & time-to-failure

```
# Predictive Maintenance Using Classification and Regression - with graphs
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.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_curve, auc

df = pd.read_csv("sensor_readings.csv")
y = df['TARGET']  # replace
X = df.drop(['TARGET'], axis=1)

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 = scaler.fit_transform(X_train); X_test = scaler.transform(X_test)

model = SVC(kernel='linear', probability=True); model.fit(X_train, y_train)
y_pred = model.predict(X_test); y_prob = model.predict_proba(X_test)[:,1]

print('Accuracy:', accuracy_score(y_test, y_pred))
print('\nClassification Report:\n', classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
```

```
sns.heatmap(cm, annot=True, fmt='d'); plt.title('Confusion Matrix'); plt.show()

fpr, tpr, _ = roc_curve(y_test, y_prob); roc_auc = auc(fpr, tpr)
plt.plot(fpr, tpr, label='AUC=%.3f'%roc_auc); plt.plot([0,1],[0,1],'k--'); plt.legend(); plt.show()
```

25. Healthcare Personalization Using Clustering and Association Rules

Dataset suggestion: **patients.csv** — Cluster patients & mine treatments

```
# Healthcare Personalization Using Clustering and Association Rules - with graphs (KMeans)
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

df = pd.read_csv("patients.csv")
X = df.select_dtypes(include=['float','int']).dropna()
scaler = StandardScaler(); X_scaled = scaler.fit_transform(X)

kmeans = KMeans(n_clusters=3, random_state=42); kmeans.fit(X_scaled)
pca = PCA(n_components=2); X2 = pca.fit_transform(X_scaled)
plt.scatter(X2[:,0], X2[:,1], c=kmeans.labels_, cmap='viridis'); plt.title('KMeans clusters (PCA view)'); plt.s
```