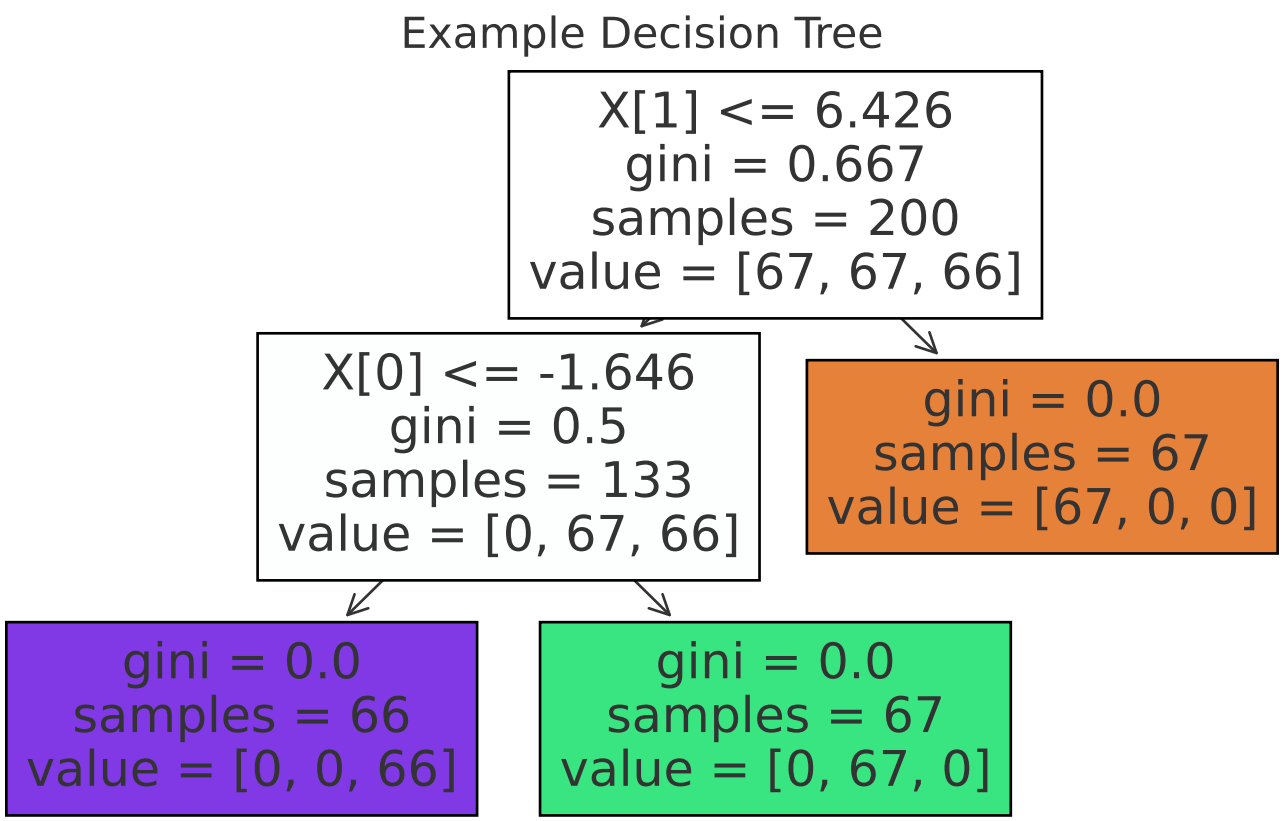
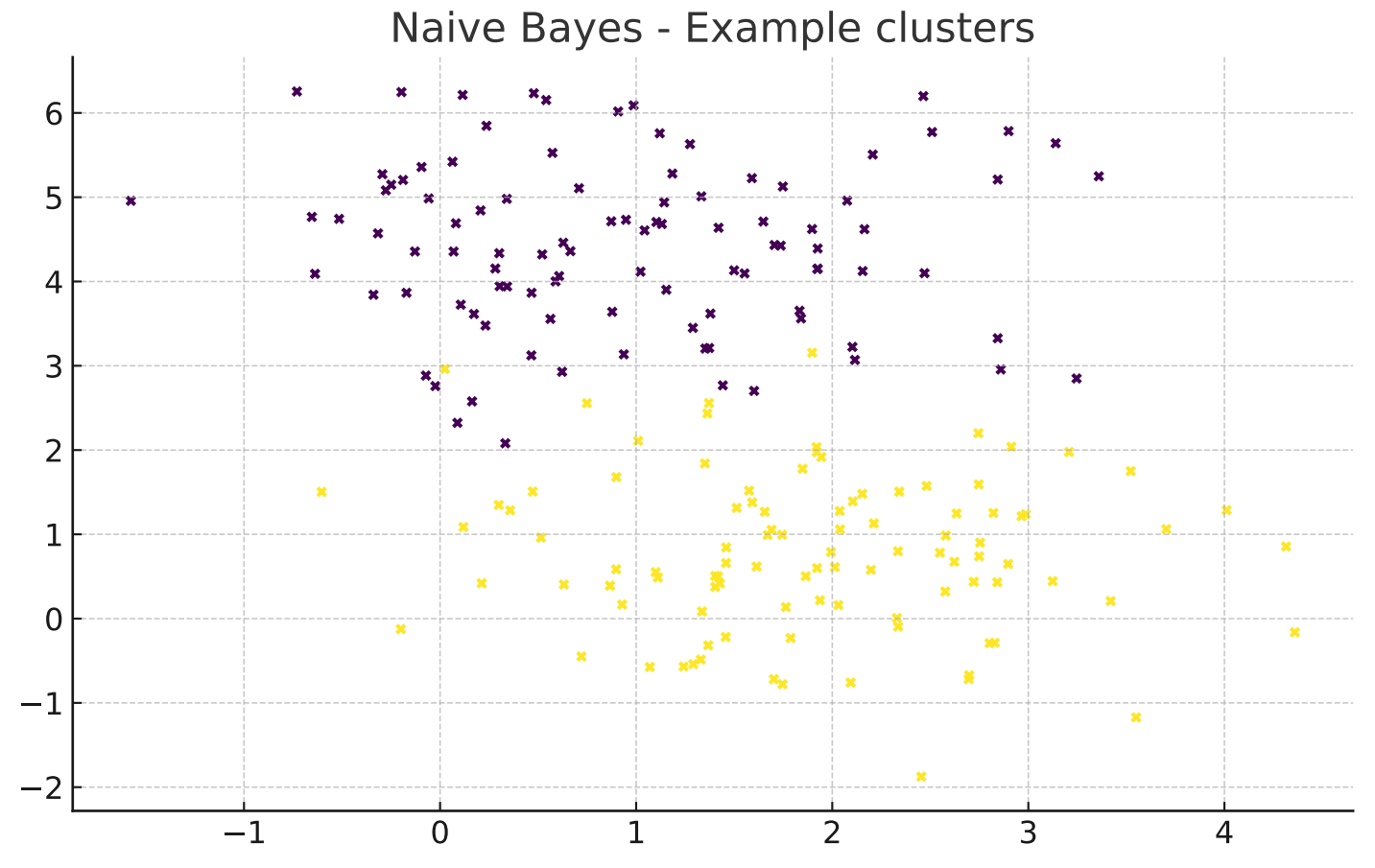
Problem 1 - Decision Tree — Dataset: madfhantr.csv

```
# Decision Tree - madfhantr.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt

df = pd.read_csv('madfhantr.csv')
X = df.drop(columns=['Loan_Status'])
y = df['Loan_Status']
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

# Plot tree (example)
plt.figure(figsize=(6,4))
plot_tree(clf, max_depth=3, filled=True)
plt.show()
```



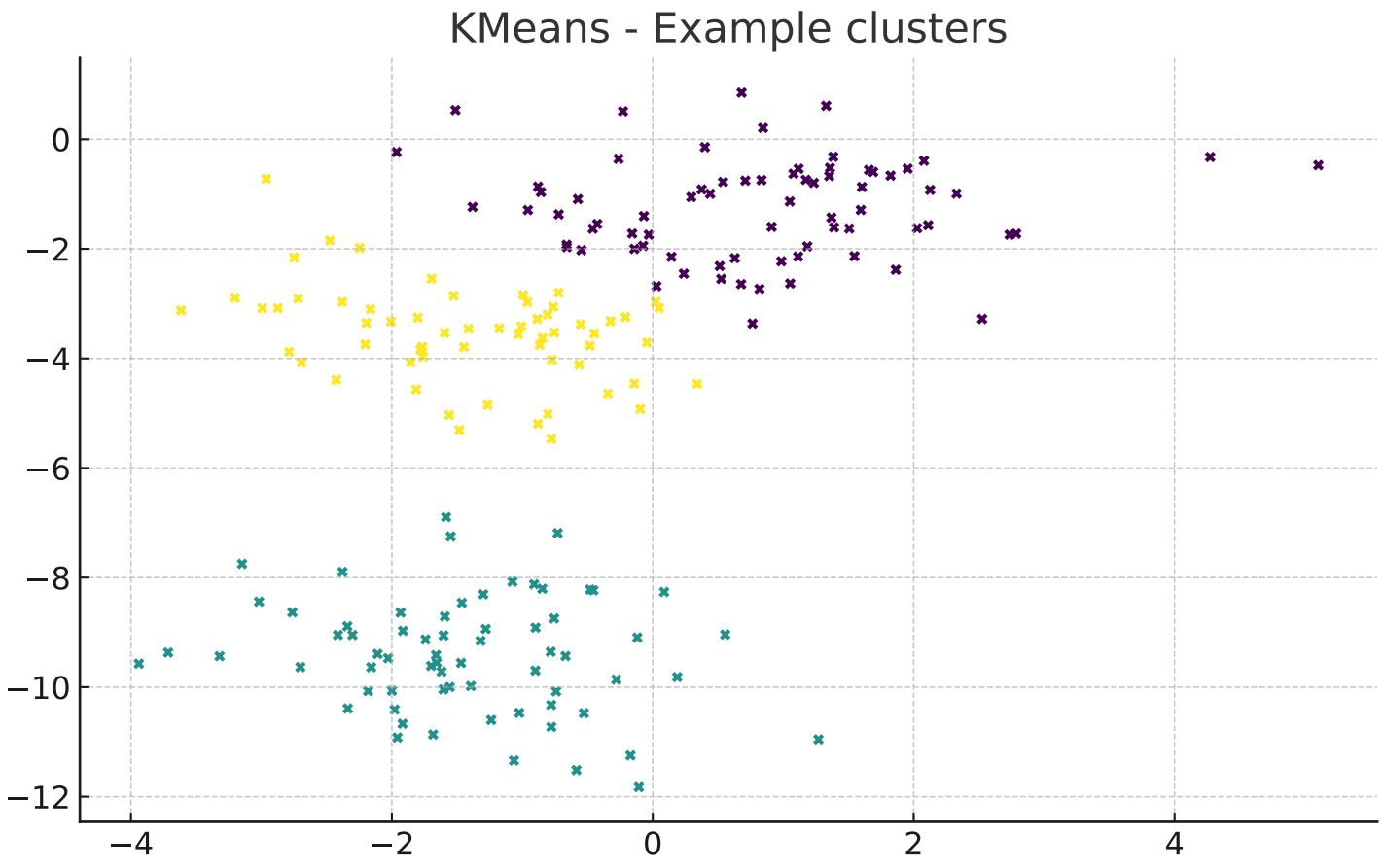
Problem 2 - Naive Bayes — Dataset: NaiveBayes.csv

```
# Naive Bayes - NaiveBayes.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report

df = pd.read_csv('NaiveBayes.csv')
X = df.drop(columns=['target'])
y = df['target']
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```



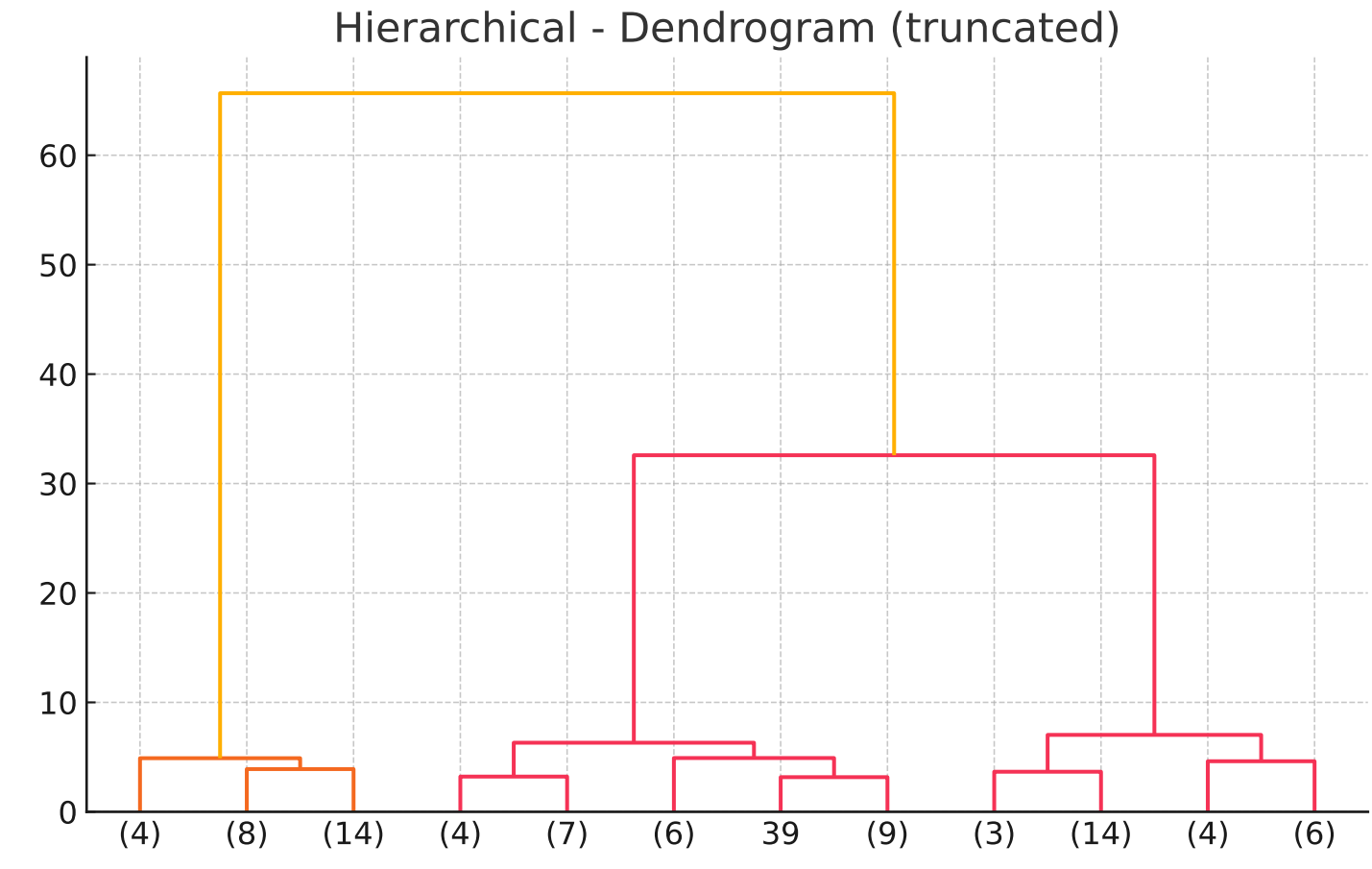
Problem 3 - K-Means (total_graduates) — Dataset: Cities_r2.csv

```
# K-Means - Cities_r2.csv
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

df = pd.read_csv('Cities_r2.csv')
X = df[['total_graduates']].dropna().values
scaler = StandardScaler()
Xs = scaler.fit_transform(X)
kmeans = KMeans(n_clusters=3, random_state=0)
labels = kmeans.fit_predict(Xs)
df['cluster'] = labels
print(df[['total_graduates','cluster']].head())
```



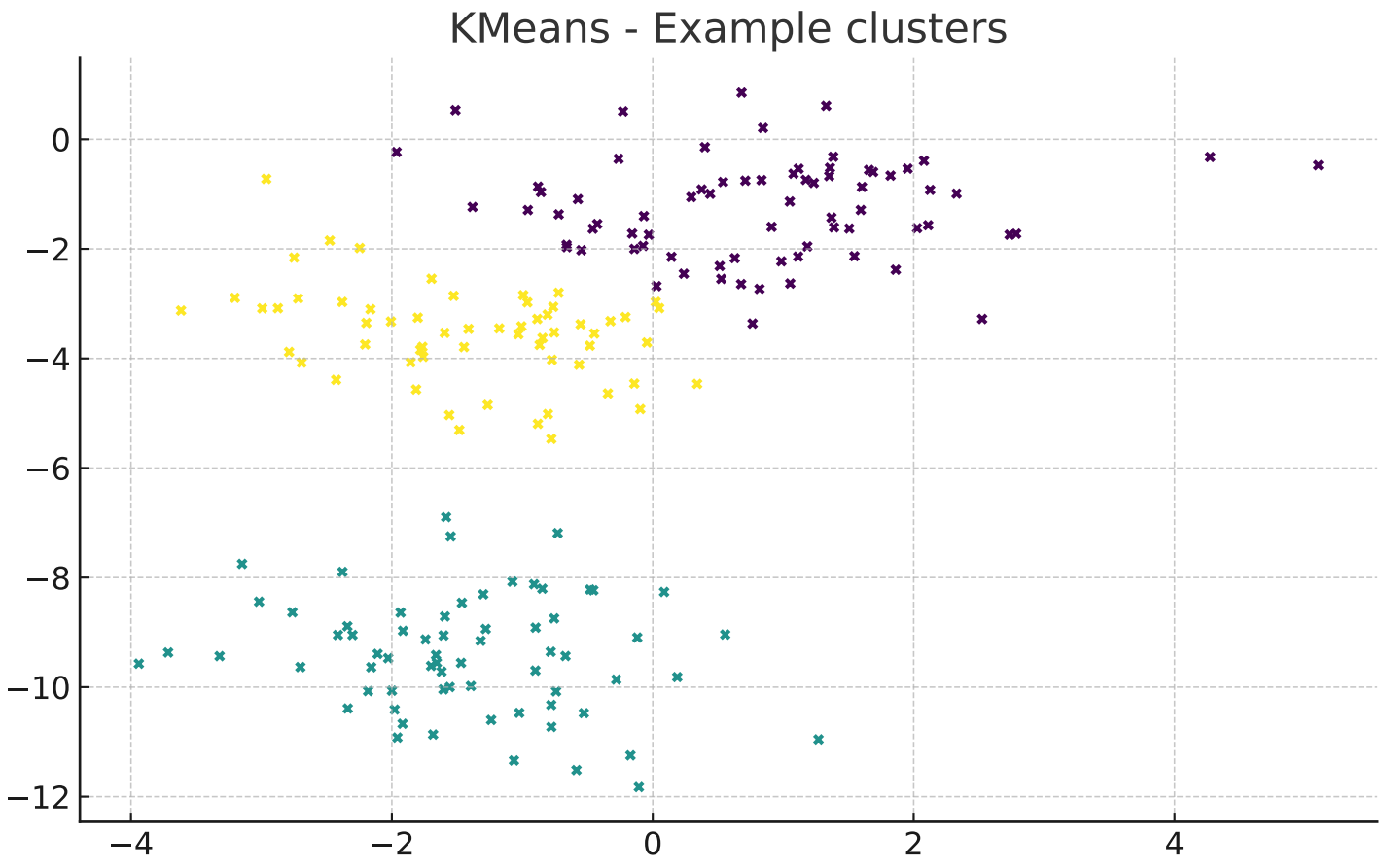
Problem 4 - Hierarchical (effective_literacy_rate_total) — Dataset: Cities_r2.csv

```
# Hierarchical - Cities_r2.csv
import pandas as pd
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

df = pd.read_csv('Cities_r2.csv')
X = df[['effective_literacy_rate_total']].dropna().values
Xs = StandardScaler().fit_transform(X)
Z = linkage(Xs, method='ward')
labels = fcluster(Z, 3, criterion='maxclust')
df['cluster'] = labels
print(df[['effective_literacy_rate_total','cluster']].head())

# Dendrogram plot shown below
```



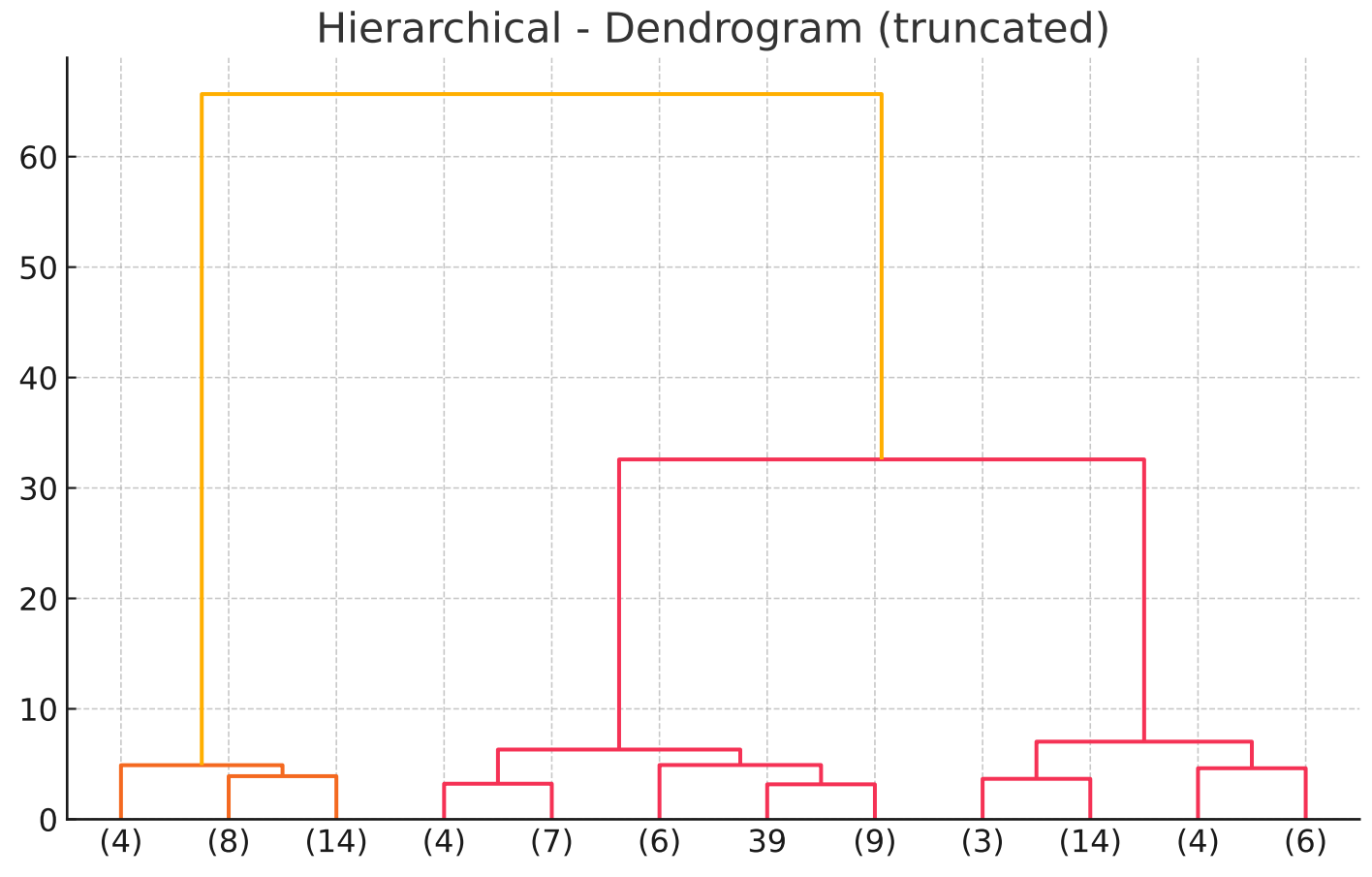
Problem 5 - K-Means (effective_literacy_rate_total) — Dataset: Cities_r2.csv

```
# K-Means - Cities_r2.csv
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

df = pd.read_csv('Cities_r2.csv')
X = df[['effective_literacy_rate_total']].dropna().values
scaler = StandardScaler()
Xs = scaler.fit_transform(X)
kmeans = KMeans(n_clusters=3, random_state=0)
labels = kmeans.fit_predict(Xs)
df['cluster'] = labels
print(df[['effective_literacy_rate_total','cluster']].head())
```



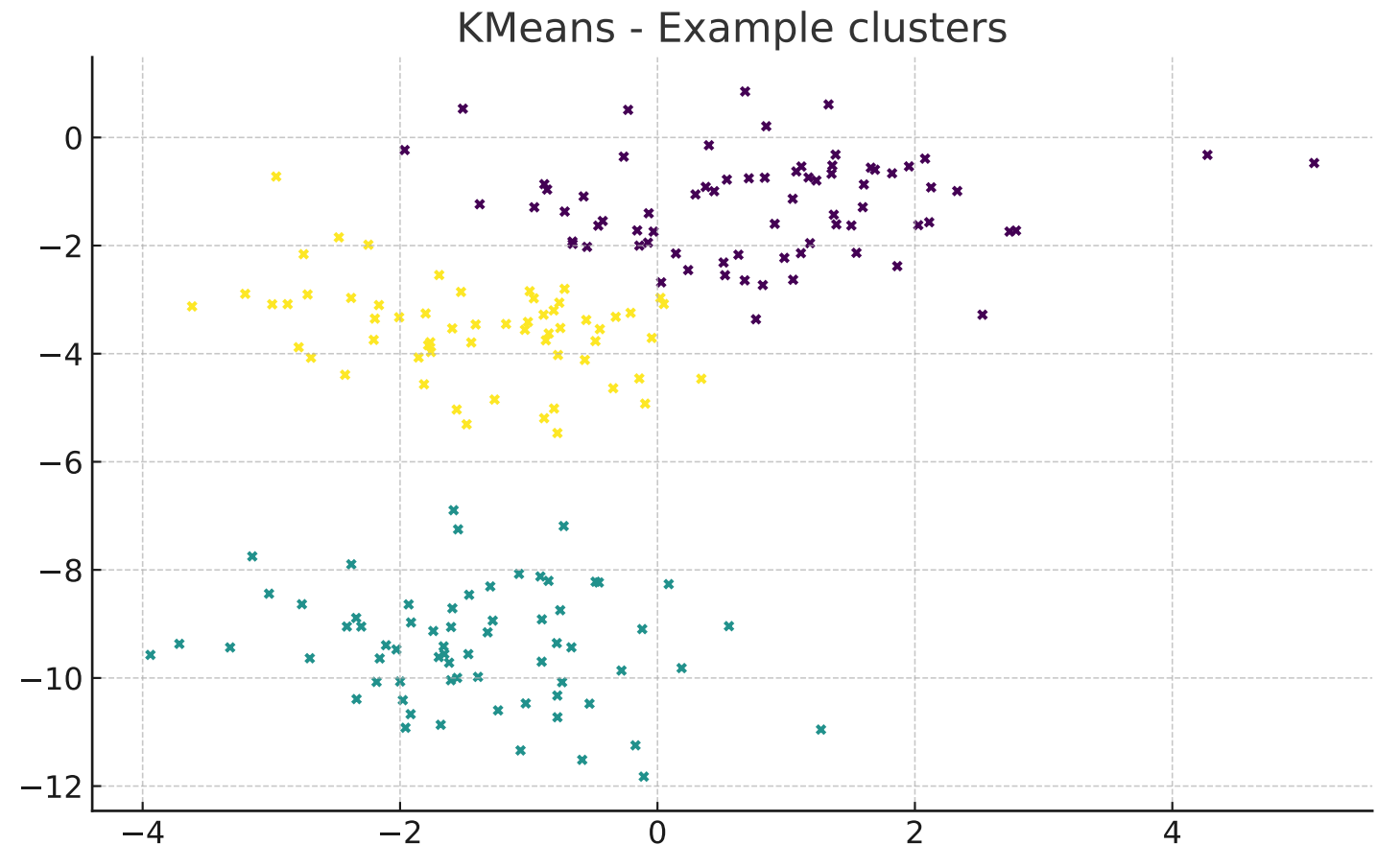
Problem 6 - Hierarchical (CRuns) — Dataset: hitters.csv

```
# Hierarchical - hitters.csv
import pandas as pd
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

df = pd.read_csv('hitters.csv')
X = df[['CRuns']].dropna().values
Xs = StandardScaler().fit_transform(X)
Z = linkage(Xs, method='ward')
labels = fcluster(Z, 3, criterion='maxclust')
df['cluster'] = labels
print(df[['CRuns','cluster']].head())

# Dendrogram plot shown below
```



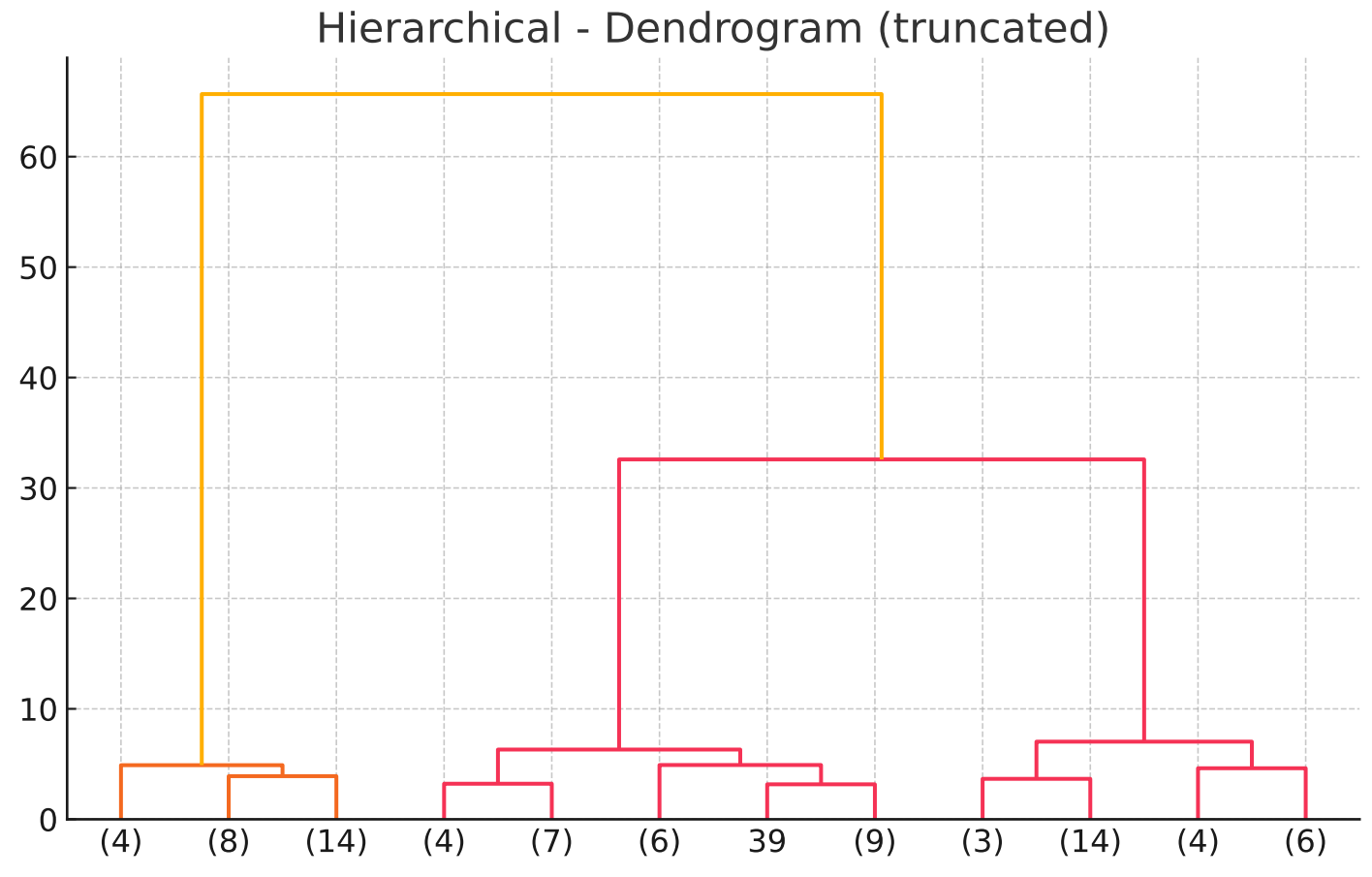
Problem 7 - K-Means (EstimatedSalary) — Dataset: Social_Network_Ads.csv

```
# K-Means - Social_Network_Ads.csv
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

df = pd.read_csv('Social_Network_Ads.csv')
X = df[['EstimatedSalary']].dropna().values
scaler = StandardScaler()
Xs = scaler.fit_transform(X)
kmeans = KMeans(n_clusters=3, random_state=0)
labels = kmeans.fit_predict(Xs)
df['cluster'] = labels
print(df[['EstimatedSalary','cluster']].head())
```



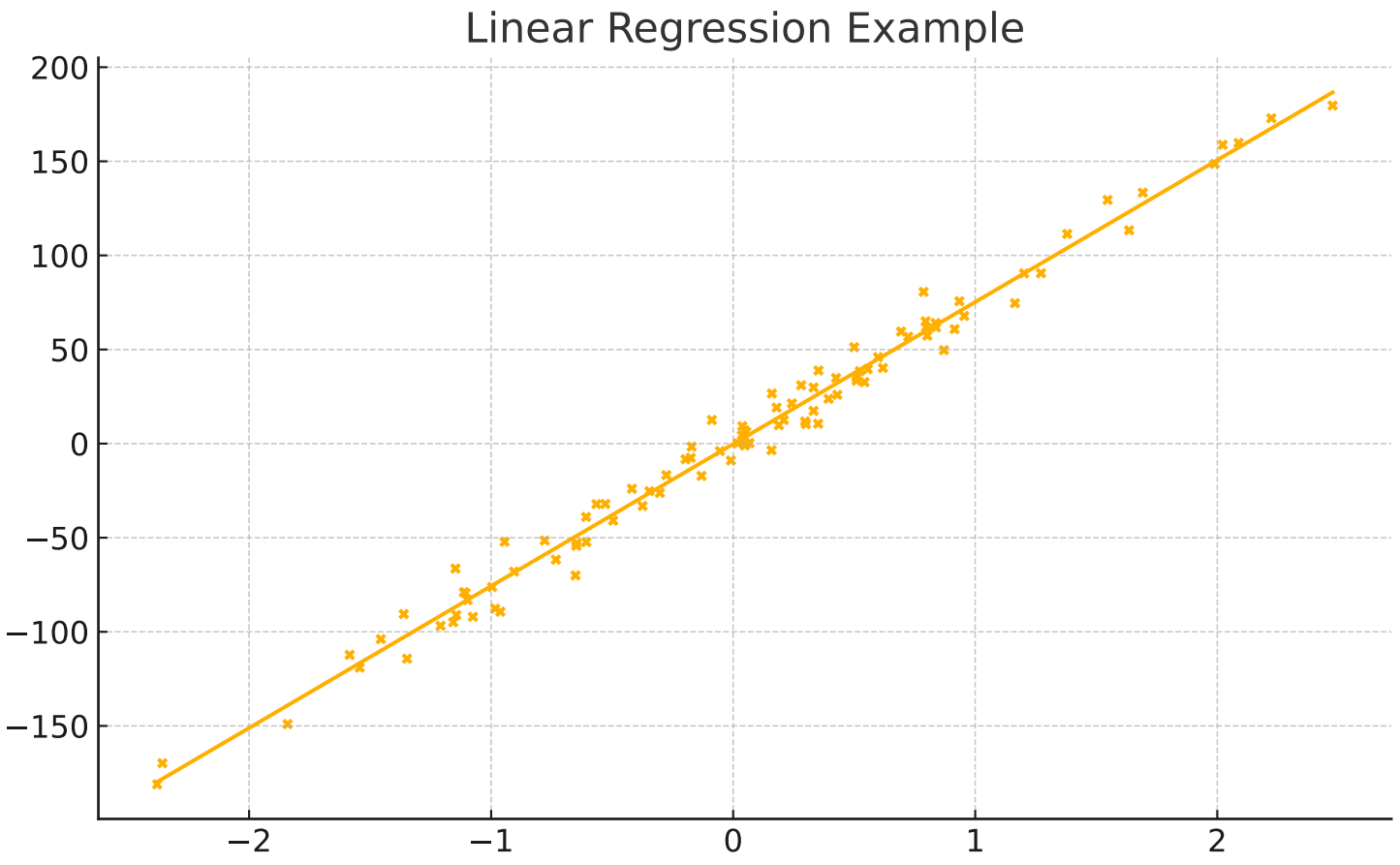
Problem 8 - Hierarchical (PROFIT) — Dataset: 50_Startups.csv

```
# Hierarchical - 50_Startups.csv
import pandas as pd
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

df = pd.read_csv('50_Startups.csv')
X = df[['PROFIT']].dropna().values
Xs = StandardScaler().fit_transform(X)
Z = linkage(Xs, method='ward')
labels = fcluster(Z, 3, criterion='maxclust')
df['cluster'] = labels
print(df[['PROFIT','cluster']].head())

# Dendrogram plot shown below
```



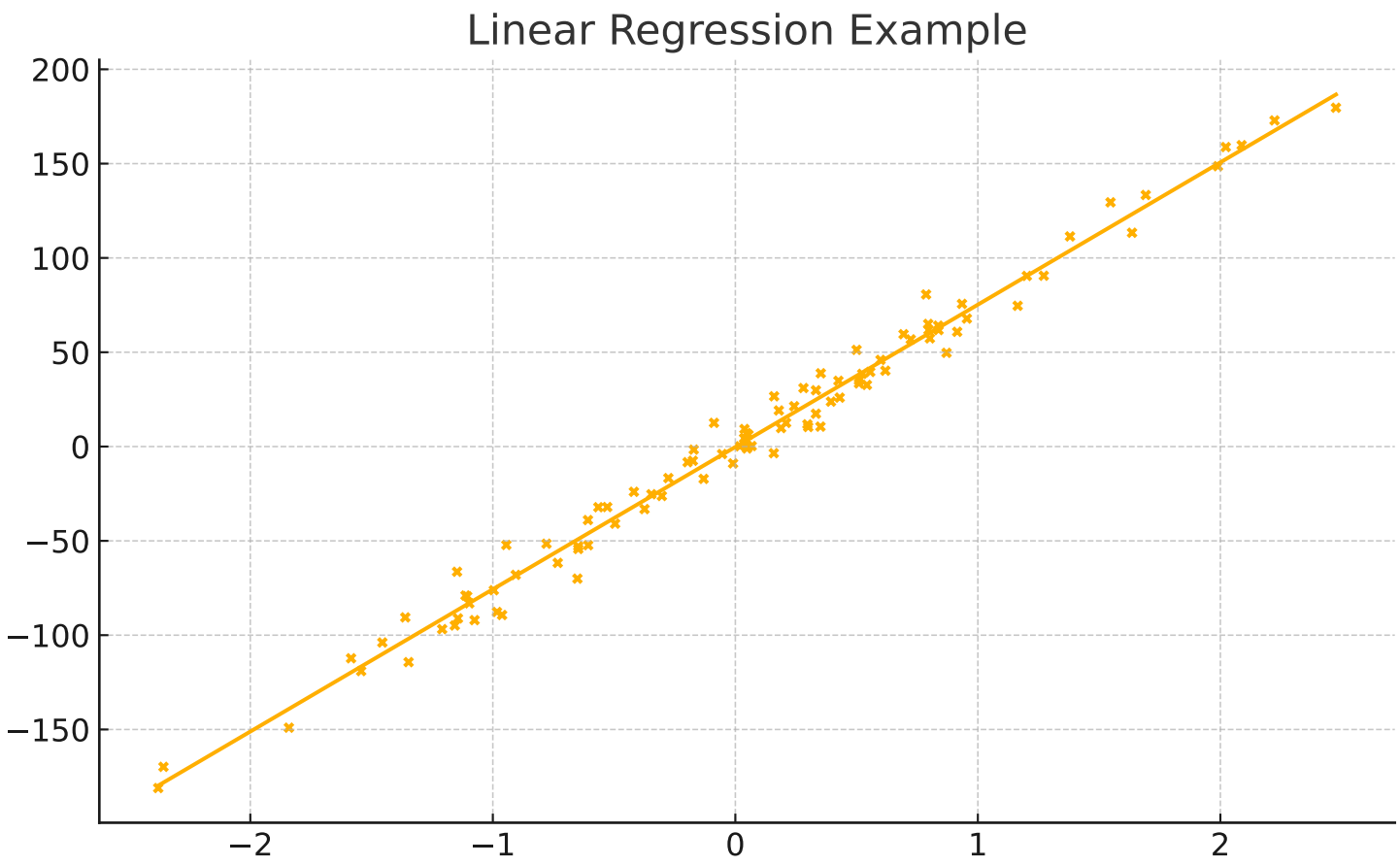
Problem 9 - Simple Linear Regression (diabetes) — Dataset: diabetes.csv

```
# Simple Linear Regression - diabetes.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
import matplotlib.pyplot as plt

df = pd.read_csv('diabetes.csv')
X = df[['BMI']].values
y = df['target'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print('Coefficients:', lr.coef_, 'Intercept:', lr.intercept_)
print('R2:', r2_score(y_test, y_pred))
# Plot below
```



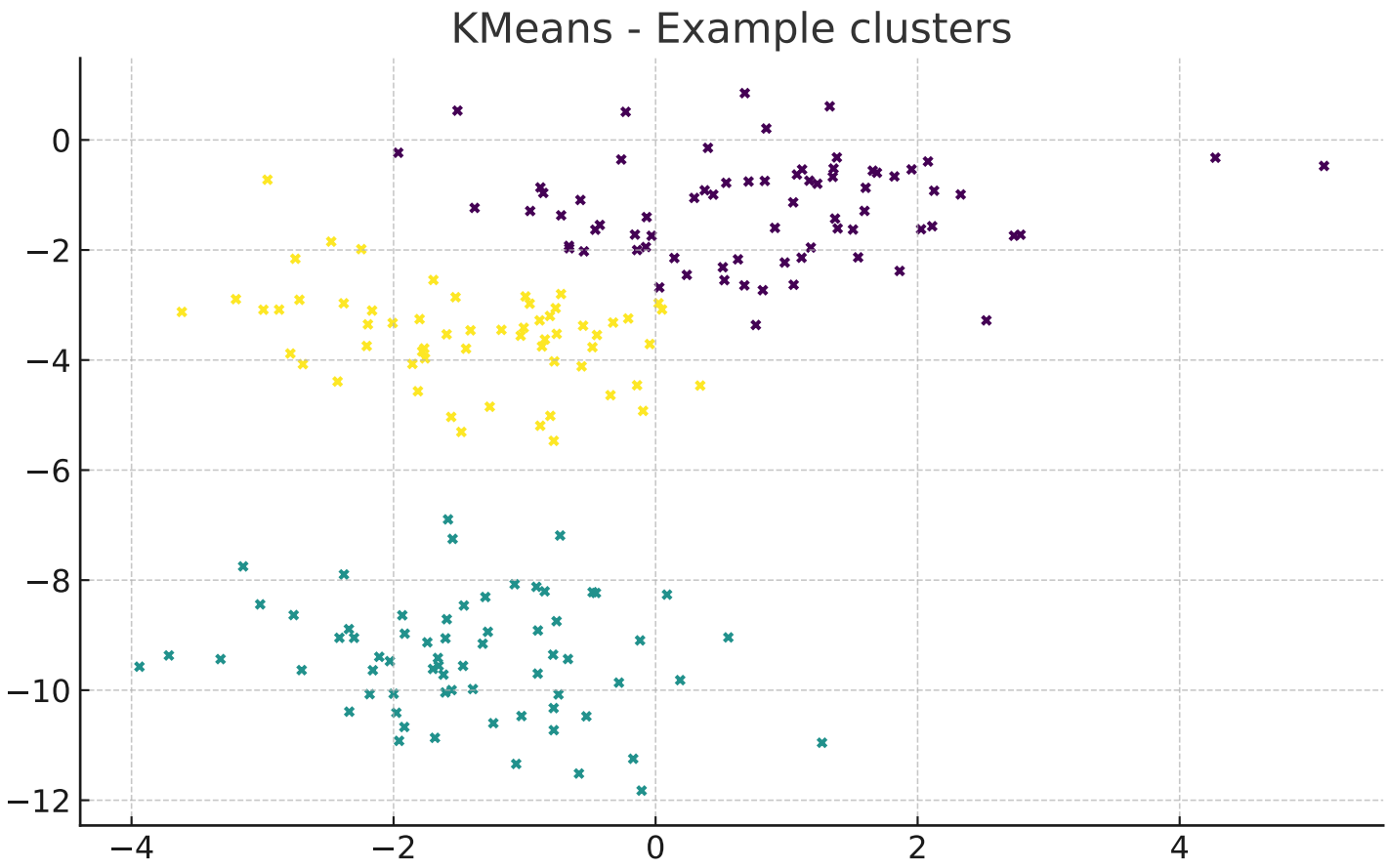
Problem 10 - Simple Linear Regression (SAT vs GPA) — Dataset: 1.01. Simple linear regression

```
# Simple Linear Regression - 1.01. Simple linear regression
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
import matplotlib.pyplot as plt

df = pd.read_csv('1.01. Simple linear regression')
X = df[['SAT']].values
y = df['GPA'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print('Coefficients:', lr.coef_, 'Intercept:', lr.intercept_)
print('R2:', r2_score(y_test, y_pred))
# Plot below
```



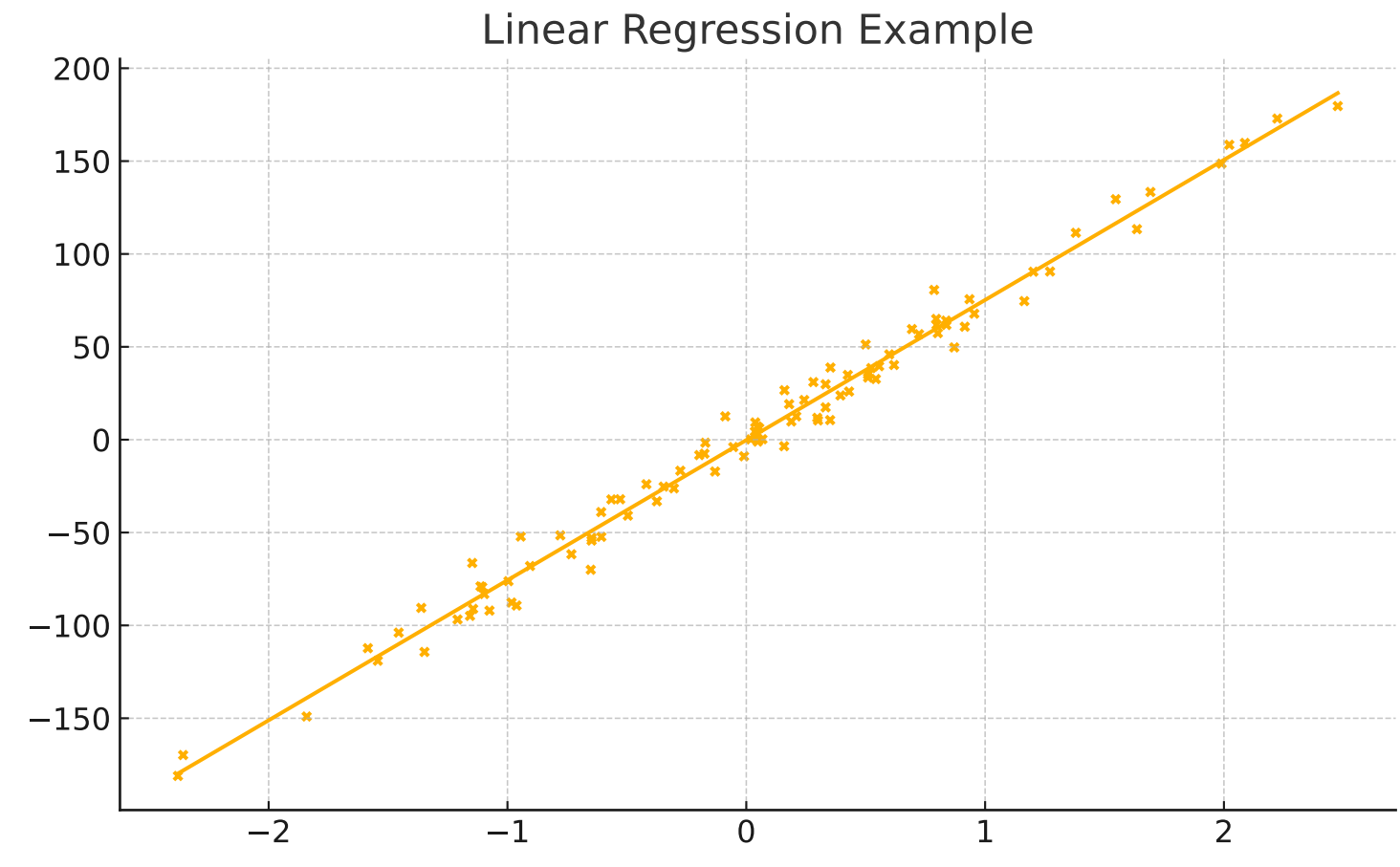
Problem 11 - K-Means (8 points example) — Dataset: inline points

```
# K-Means - 8 points example (P1..P8)
import numpy as np
from sklearn.cluster import KMeans
pts = np.array([[0.1,0.6],[0.15,0.71],[0.08,0.9],[0.16,0.85],[0.2,0.3],[0.25,0.5],[0.24,0.1],[0.3,0.2]])
# Initial centroids m1=P1 and m2=P8
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2, init=np.array([pts[0], pts[7]]), n_init=1, random_state=0).fit(pts)
print('Labels:', kmeans.labels_)
print('Centroids:', kmeans.cluster_centers_)
```



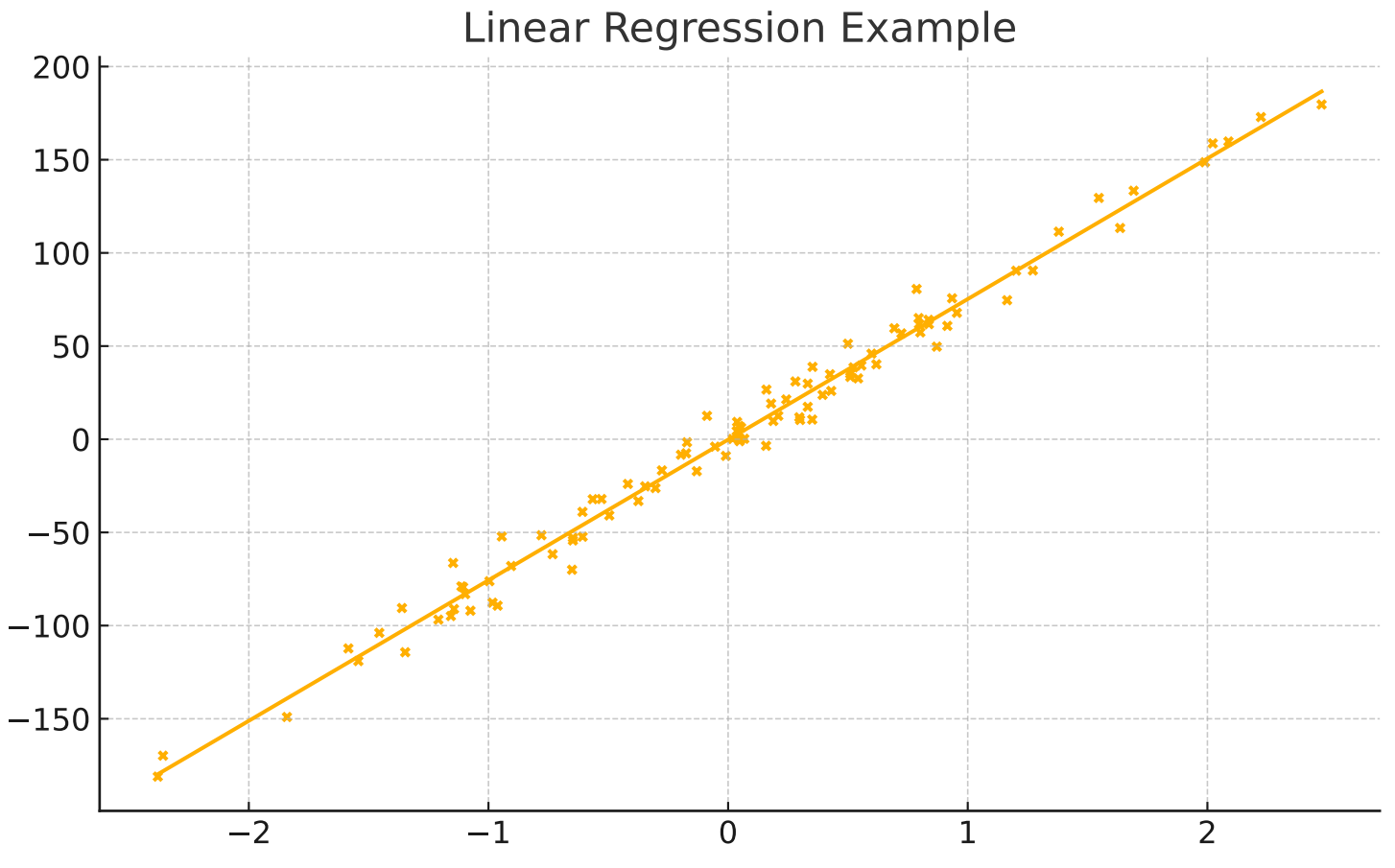
Problem 12 - Simple Linear Regression (advertising TV) — Dataset: advertising.csv

```
# Simple Linear Regression - advertising.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
import matplotlib.pyplot as plt

df = pd.read_csv('advertising.csv')
X = df[['TV']].values
y = df['Sales'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print('Coefficients:', lr.coef_, 'Intercept:', lr.intercept_)
print('R2:', r2_score(y_test, y_pred))
# Plot below
```



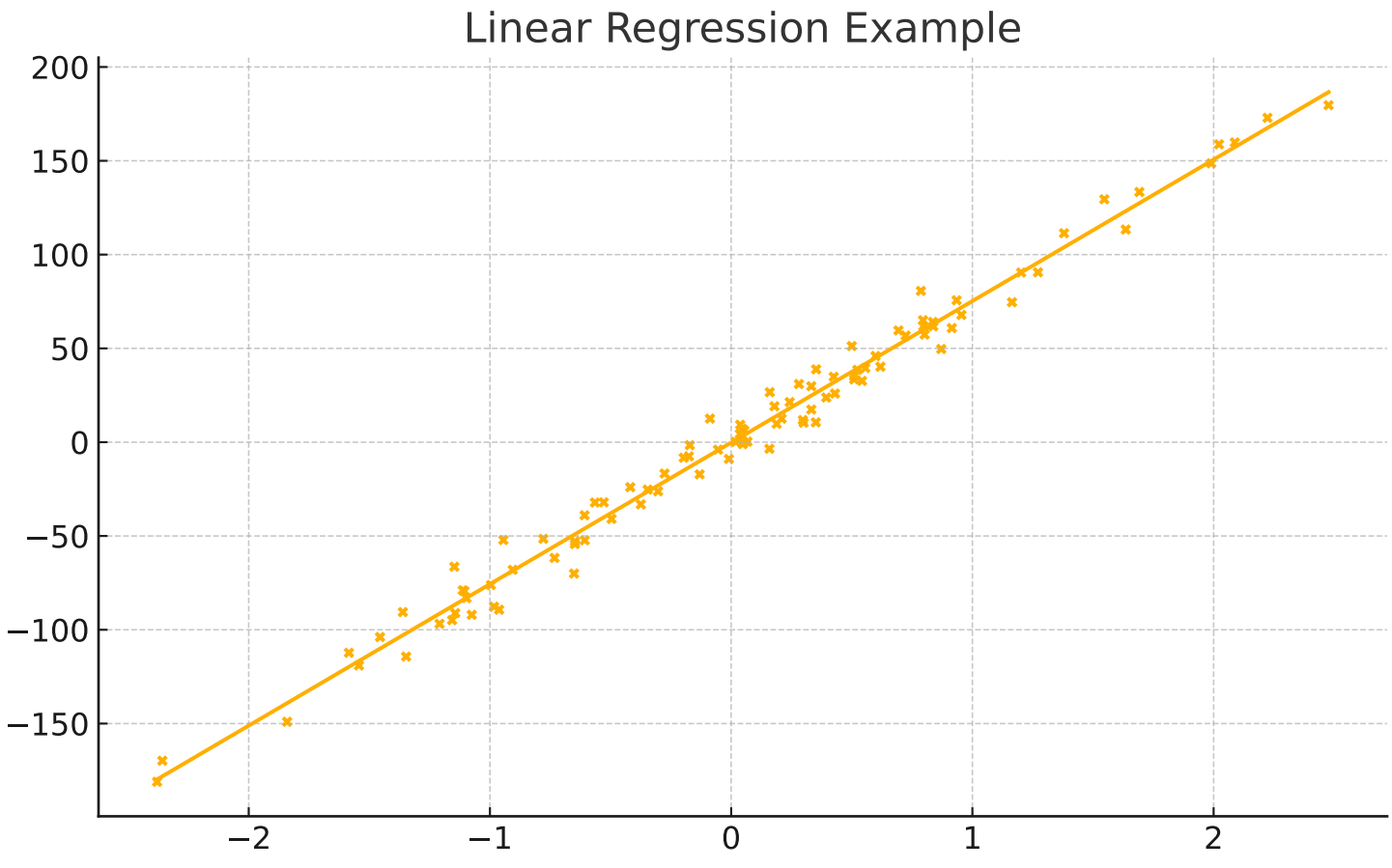
Problem 13 - Simple Linear Regression (advertising Radio) — Dataset: advertising.csv

```
# Simple Linear Regression - advertising.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
import matplotlib.pyplot as plt

df = pd.read_csv('advertising.csv')
X = df[['Radio']].values
y = df['Sales'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print('Coefficients:', lr.coef_, 'Intercept:', lr.intercept_)
print('R2:', r2_score(y_test, y_pred))
# Plot below
```



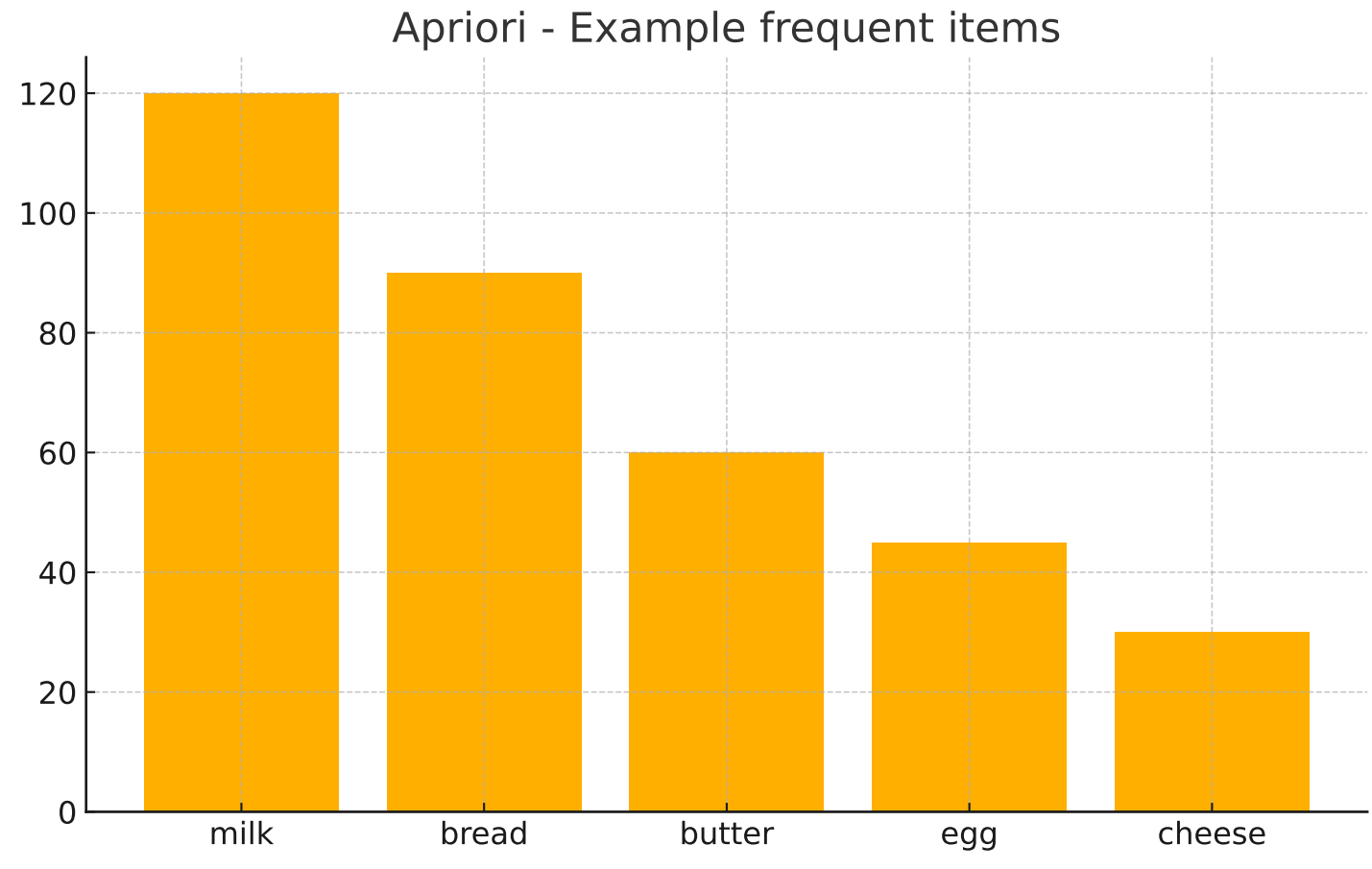
Problem 14 - Simple Linear Regression (advertising Newspaper) — Dataset: advertising.csv

```
# Simple Linear Regression - advertising.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error
import matplotlib.pyplot as plt

df = pd.read_csv('advertising.csv')
X = df[['Newspaper']].values
y = df['Sales'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print('Coefficients:', lr.coef_, 'Intercept:', lr.intercept_)
print('R2:', r2_score(y_test, y_pred))
# Plot below
```



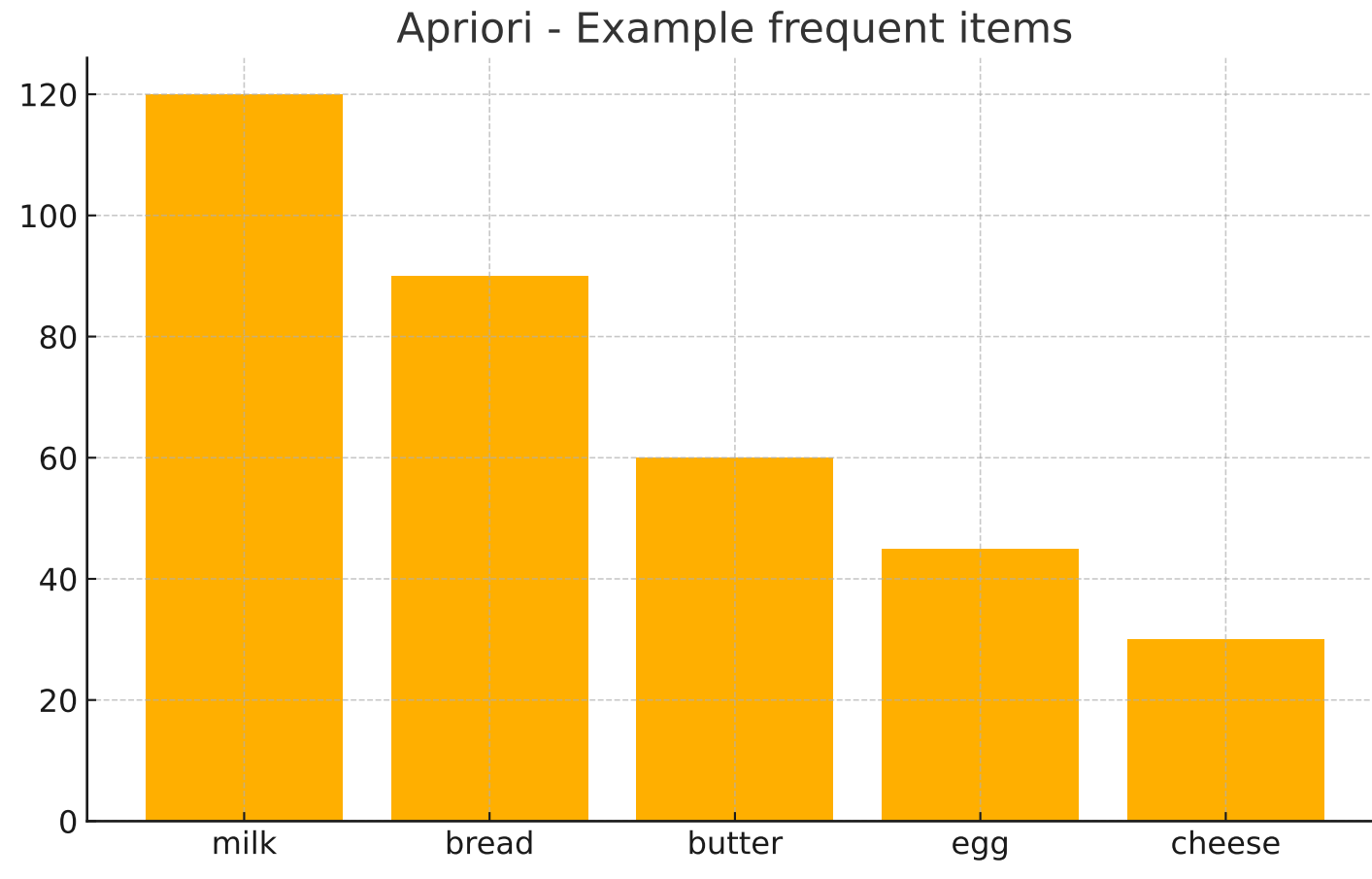
Problem 15 - Apriori (Order1.csv) — Dataset: Order1.csv

```
# Apriori - Order1.csv (placeholder)
# mlxtend may not be installed in your environment. Minimal code template:
# from mlxtend.preprocessing import TransactionEncoder
# from mlxtend.frequent_patterns import apriori, association_rules
# df = pd.read_csv('Order1.csv', header=None)
# transactions = df.apply(lambda row: row.dropna().tolist(), axis=1).tolist()
# te = TransactionEncoder()
# te_ary = te.fit(transactions).transform(transactions)
# df_trans = pd.DataFrame(te_ary, columns=te.columns_)
# frequent = apriori(df_trans, min_support=0.01, use_colnames=True)
# rules = association_rules(frequent, metric='confidence', min_threshold=0.3)
# print(frequent.head()); print(rules.head())
```



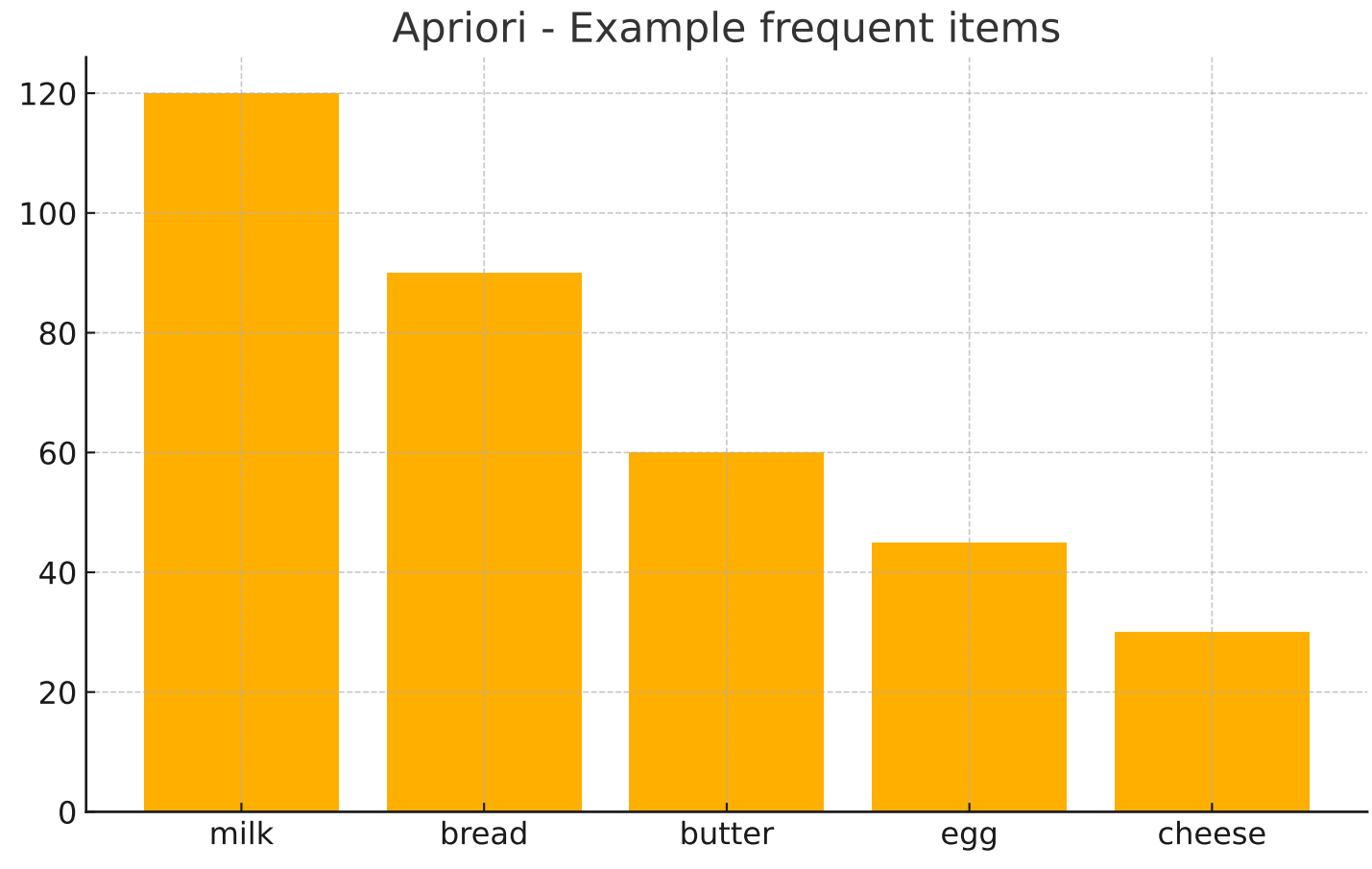
Problem 16 - Apriori (Order2.csv) — Dataset: Order2.csv

```
# Apriori - Order2.csv (placeholder)
# mlxtend may not be installed in your environment. Minimal code template:
# from mlxtend.preprocessing import TransactionEncoder
# from mlxtend.frequent_patterns import apriori, association_rules
# df = pd.read_csv('Order2.csv', header=None)
# transactions = df.apply(lambda row: row.dropna().tolist(), axis=1).tolist()
# te = TransactionEncoder()
# te_ary = te.fit(transactions).transform(transactions)
# df_trans = pd.DataFrame(te_ary, columns=te.columns_)
# frequent = apriori(df_trans, min_support=0.01, use_colnames=True)
# rules = association_rules(frequent, metric='confidence', min_threshold=0.3)
# print(frequent.head()); print(rules.head())
```



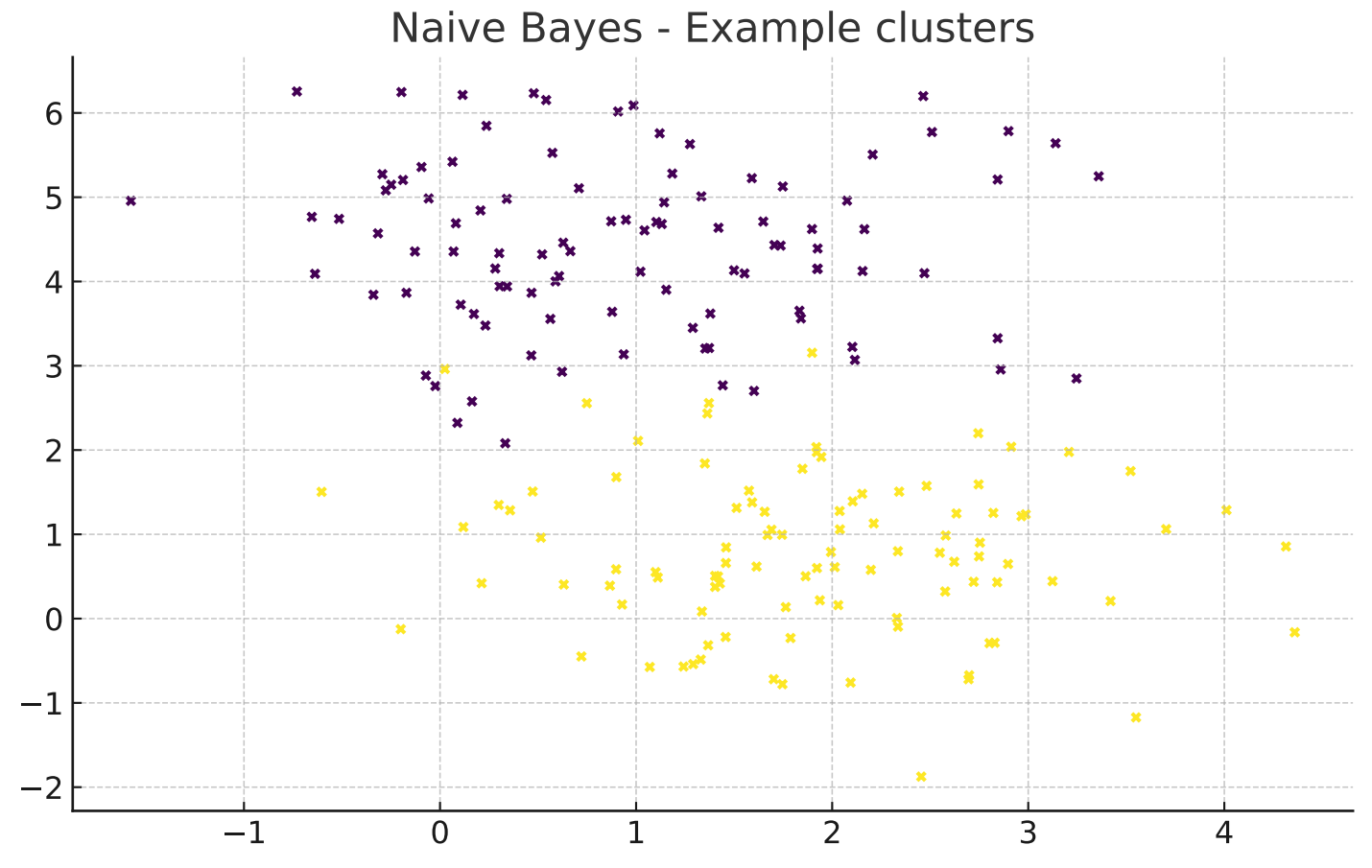
Problem 17 - Apriori (Order3.csv) — Dataset: Order3.csv

```
# Apriori - Order3.csv (placeholder)
# mlxtend may not be installed in your environment. Minimal code template:
# from mlxtend.preprocessing import TransactionEncoder
# from mlxtend.frequent_patterns import apriori, association_rules
# df = pd.read_csv('Order3.csv', header=None)
# transactions = df.apply(lambda row: row.dropna().tolist(), axis=1).tolist()
# te = TransactionEncoder()
# te_ary = te.fit(transactions).transform(transactions)
# df_trans = pd.DataFrame(te_ary, columns=te.columns_)
# frequent = apriori(df_trans, min_support=0.01, use_colnames=True)
# rules = association_rules(frequent, metric='confidence', min_threshold=0.3)
# print(frequent.head()); print(rules.head())
```



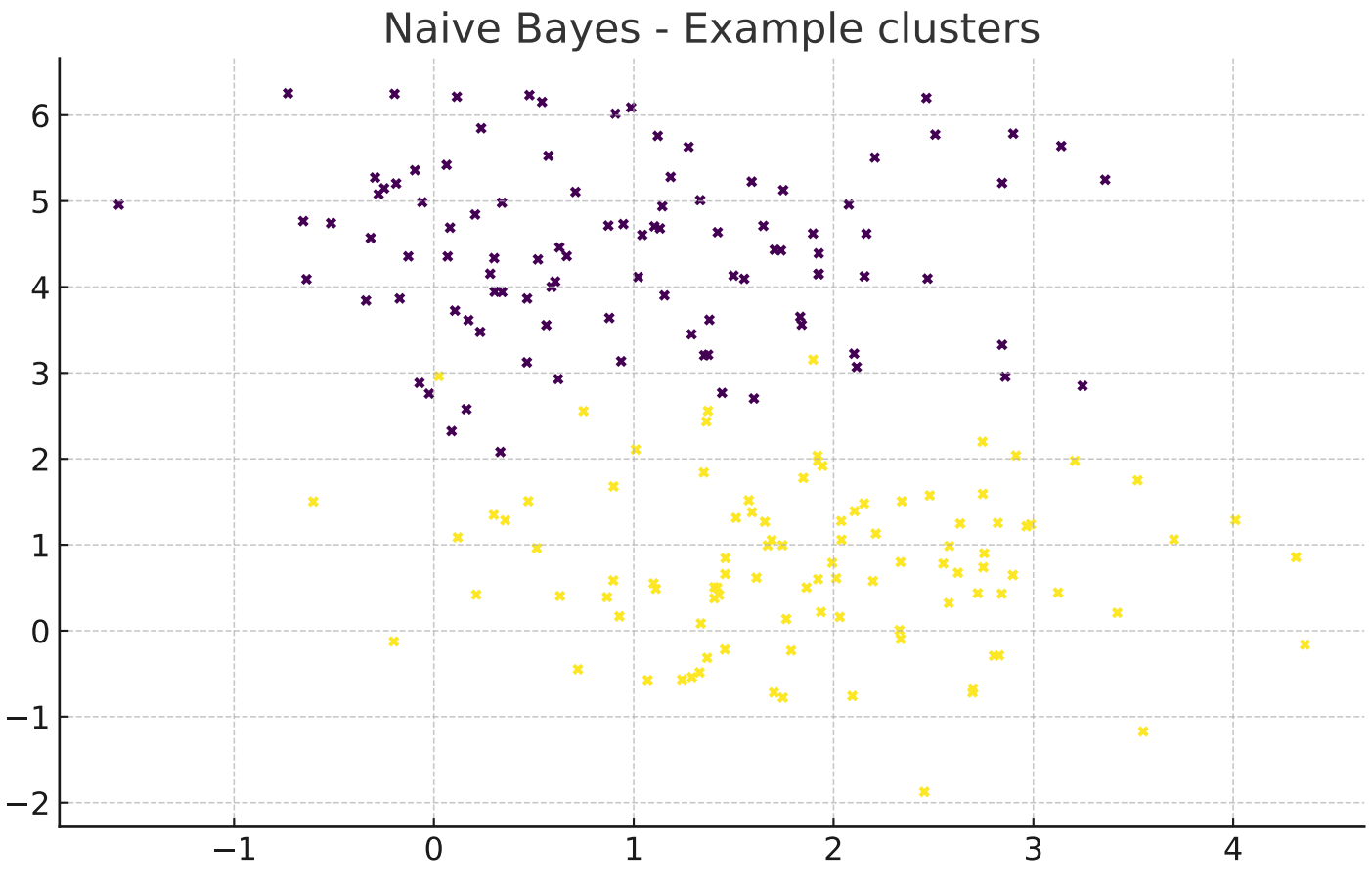
Problem 18 - Naive Bayes (pima) — Dataset: pima-indians-diabetes.csv

```
# Naive Bayes - pima-indians-diabetes.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report

df = pd.read_csv('pima-indians-diabetes.csv')
X = df.drop(columns=['target'])
y = df['target']
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```



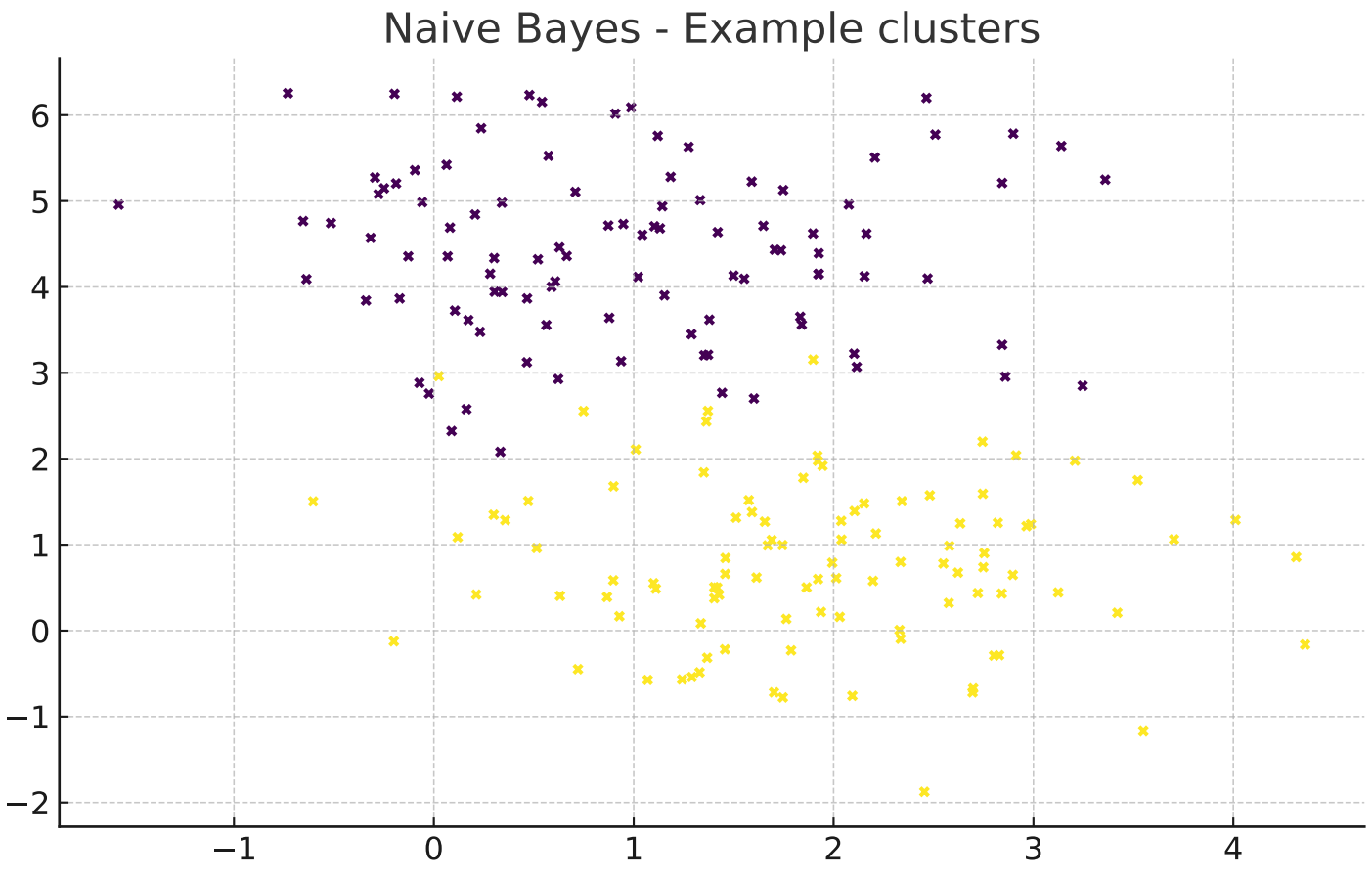
Problem 19 - Naive Bayes (Social_Network_Ads) — Dataset: Social_Network_Ads.csv

```
# Naive Bayes - Social_Network_Ads.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report

df = pd.read_csv('Social_Network_Ads.csv')
X = df.drop(columns=['target'])
y = df['target']
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```



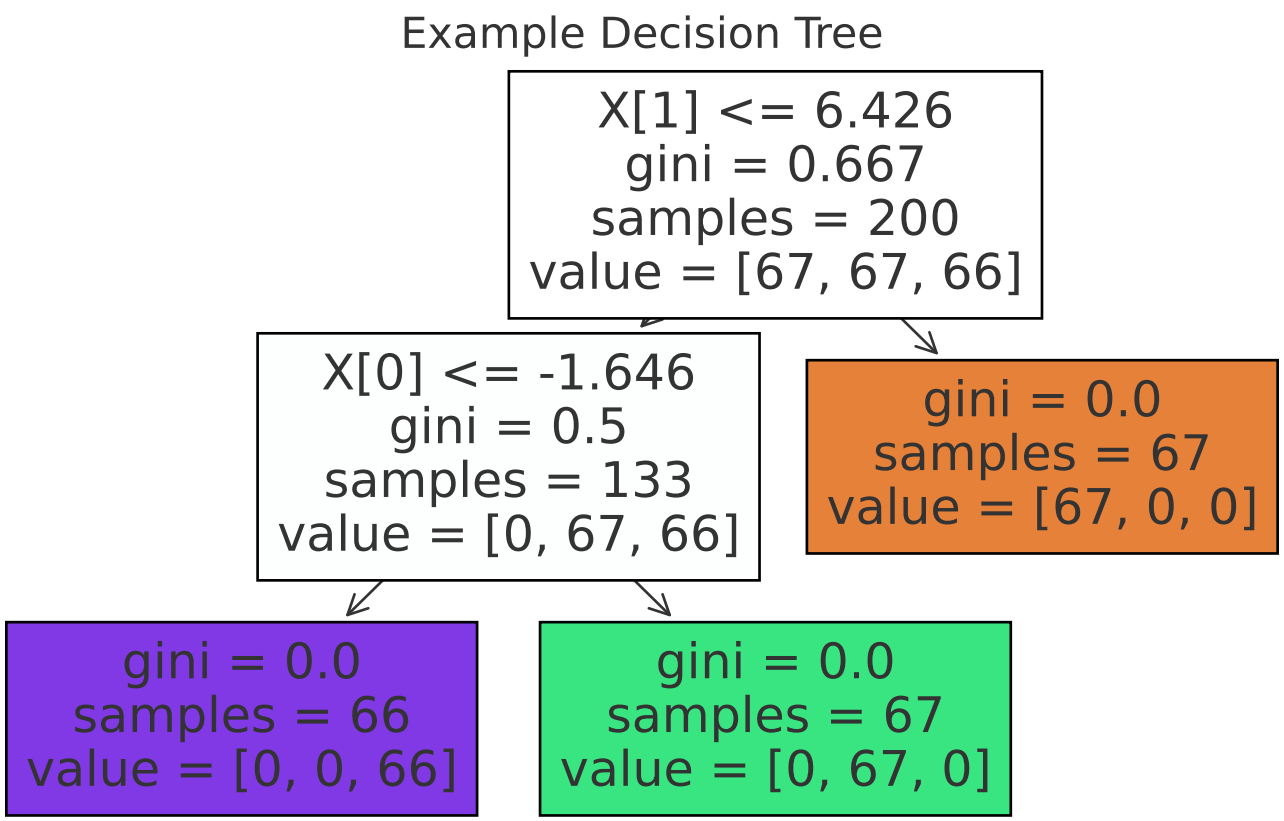
Problem 20 - Naive Bayes (Social_Network_Ads variant) — Dataset: Social_Network_Ads.csv

```
# Naive Bayes - Social_Network_Ads.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report

df = pd.read_csv('Social_Network_Ads.csv')
X = df.drop(columns=['target'])
y = df['target']
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

model = GaussianNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
print(classification_report(y_test, y_pred))
```



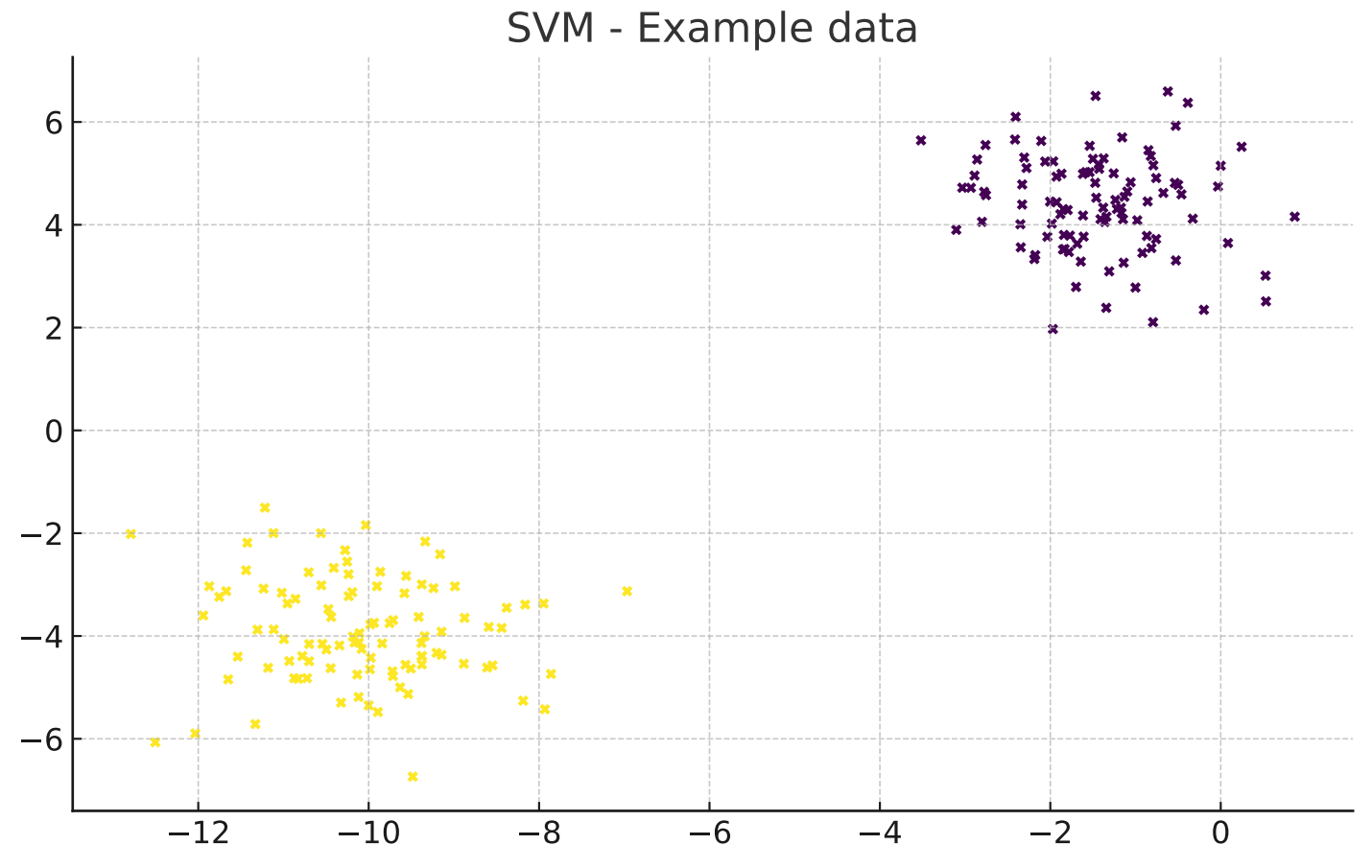
Problem 21 - Decision Tree (pima) — Dataset: pima-indians-diabetes.csv

```
# Decision Tree - pima-indians-diabetes.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt

df = pd.read_csv('pima-indians-diabetes.csv')
X = df.drop(columns=['target'])
y = df['target']
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))

# Plot tree (example)
plt.figure(figsize=(6,4))
plot_tree(clf, max_depth=3, filled=True)
plt.show()
```



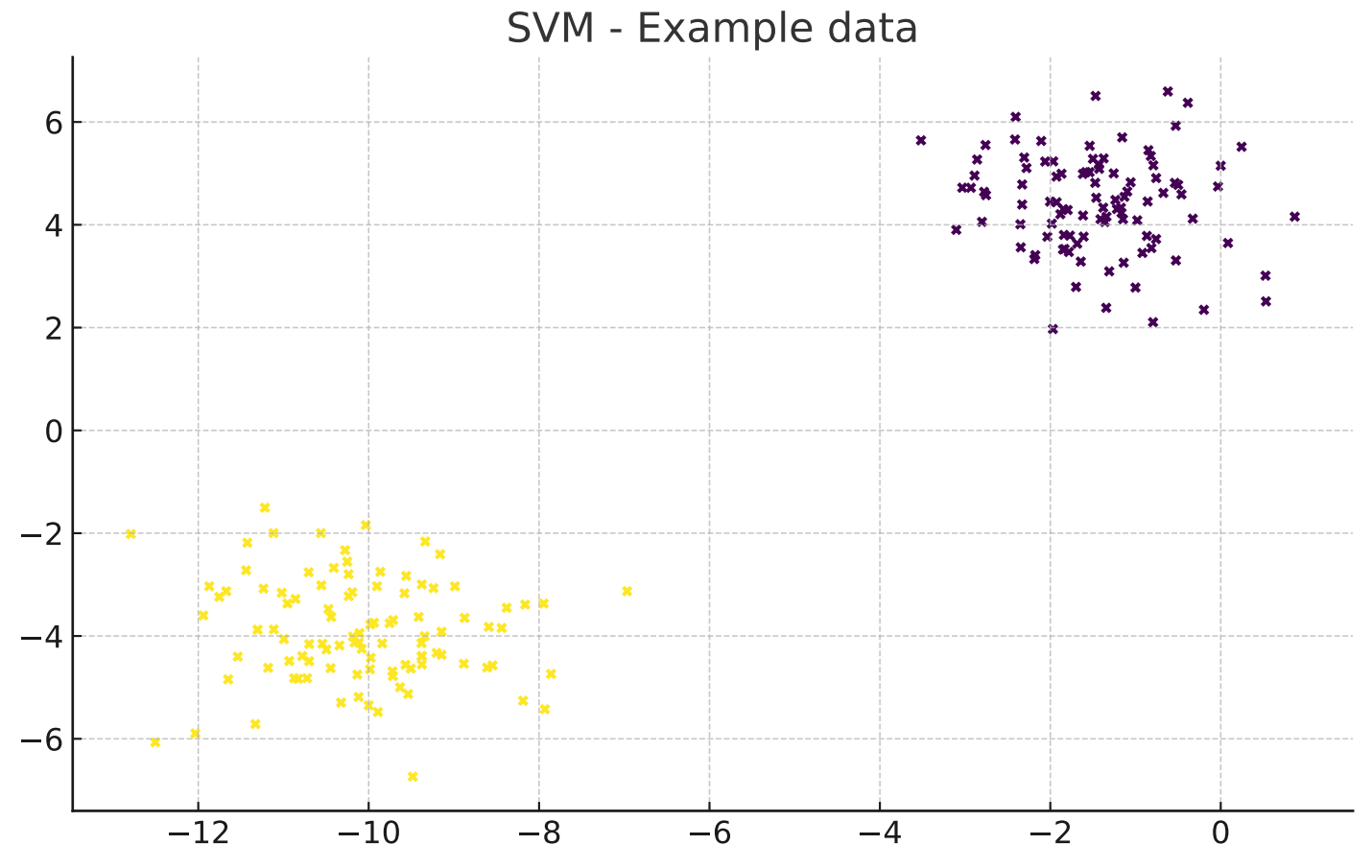
Problem 22 - SVM (Mobile Price Range) — Dataset: test.csv

```
# SVM - test.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

df = pd.read_csv('test.csv')
X = df.drop(columns=['label'])
y = df['label']
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

for kernel in ['linear','poly','rbf']:
    clf = SVC(kernel=kernel, random_state=1)
    clf.fit(X_train, y_train)
    print('Kernel', kernel, 'Accuracy:', accuracy_score(y_test, clf.predict(X_test)))
```



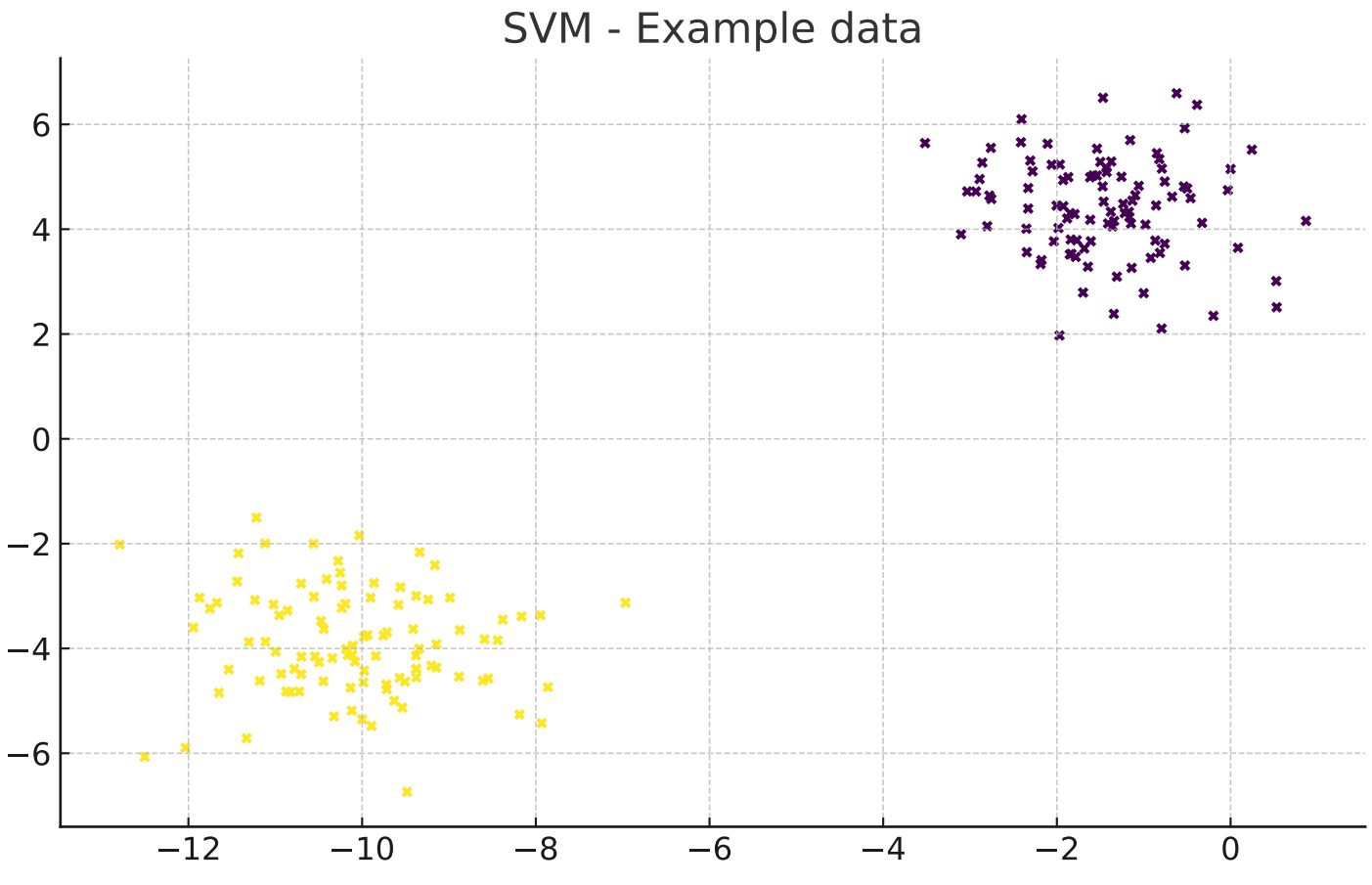
Problem 23 - SVM (UniversalBank) — Dataset: UniversalBank.csv

```
# SVM - UniversalBank.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

df = pd.read_csv('UniversalBank.csv')
X = df.drop(columns=['label'])
y = df['label']
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

for kernel in ['linear','poly','rbf']:
    clf = SVC(kernel=kernel, random_state=1)
    clf.fit(X_train, y_train)
    print('Kernel', kernel, 'Accuracy:', accuracy_score(y_test, clf.predict(X_test)))
```



Problem 24 - SVM (user_behavior_dataset) — Dataset: user_behavior_dataset.csv

```
# SVM - user_behavior_dataset.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

df = pd.read_csv('user_behavior_dataset.csv')
X = df.drop(columns=['label'])
y = df['label']
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

for kernel in ['linear','poly','rbf']:
    clf = SVC(kernel=kernel, random_state=1)
    clf.fit(X_train, y_train)
    print('Kernel', kernel, 'Accuracy:', accuracy_score(y_test, clf.predict(X_test)))
```



Problem 25 - SVM (bank_transactions_data_2.csv) — Dataset: bank_transactions_data_2.csv

```
# SVM - bank_transactions_data_2.csv
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

df = pd.read_csv('bank_transactions_data_2.csv')
X = df.drop(columns=['label'])
y = df['label']
X = pd.get_dummies(X, drop_first=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

for kernel in ['linear','poly','rbf']:
    clf = SVC(kernel=kernel, random_state=1)
    clf.fit(X_train, y_train)
    print('Kernel', kernel, 'Accuracy:', accuracy_score(y_test, clf.predict(X_test)))
```