LP-1: Minimal Python Solutions — All ML Problems
Author: Generated by ChatGPT

Instructions: Replace dataset file paths with your local paths if needed. Each section contains minimal, runnable Python code (uses scikit-learn, pandas, matplotlib where required).

Decision Tree (Classification) — madfhantr.csv
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix

df = pd.read_csv('madfhantr.csv')
# Example: target column named 'Loan_Status' (change if different)
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))
```

Naive Bayes (Classification)
```python
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')  # or 'pima-indians-diabetes.csv' or 'Social_Network_Ads.csv'
X = df.drop(columns=['target'])    # replace 'target' with actual target column
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))
```

Support Vector Machine (SVM)
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
```

```
from sklearn.metrics import accuracy_score, classification_report

df = pd.read_csv('test.csv')   # replace filename as needed
X = df.drop(columns=['label']) # replace 'label' with actual target
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, probability=False, random_state=1)
    clf.fit(X_train, y_train)
    print(f"Kernel={kernel} ->", accuracy_score(y_test, clf.predict(X_test)))
```

K-Means Clustering
```python
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

df = pd.read_csv('Cities_r2.csv')
# Example clustering on 'total_graduates' column
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())
```

Hierarchical Clustering
```python
import pandas as pd
from scipy.cluster.hierarchy import dendrogram, linkage, fcluster
from sklearn.preprocessing import StandardScaler

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')
# To get flat clusters, e.g., 3 clusters:
labels = fcluster(Z, 3, criterion='maxclust')
df['cluster'] = labels
print(df[['effective_literacy_rate_total','cluster']].head())
```

Simple Linear Regression

```python
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')  # or '1.01. Simple linear regression' or 'advertising.csv'
# Use one feature, e.g., 'BMI' or 'SAT' etc. Replace columns as appropriate.
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), 'MSE:', mean_squared_error(y_test, y_pred))

# Plot regression line
plt.scatter(X_test, y_test)
xs = X_test
plt.plot(xs, y_pred)
plt.title('Linear Regression')
plt.xlabel('Feature')
plt.ylabel('Target')
plt.show()
```

Market Basket Analysis (Apriori)
```python
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

# Load transactions: each row is a list of items; adapt reading if CSV layout differs.
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.sort_values('support', ascending=False).head())
print(rules.head())
```

Quick Examples: Naive Bayes and Decision Tree
```python
# See previous sections for full templates. For Decision Tree on 'pima-indians-diabetes.csv':
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

df = pd.read_csv('pima-indians-diabetes.csv', header=None)
# if headers absent, set column names or use indexing; assume last column is target:
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
clf = DecisionTreeClassifier(random_state=0)
clf.fit(X_train, y_train)
print(classification_report(y_test, clf.predict(X_test)))
```
