Slip 1

Packages: pandas, mlxtend, matplotlib, scikit-learn
Run (3 steps): 1) pip install mlxtend scikit-learn pandas matplotlib 2) place groceries.csv, iris.csv in notebook folder 3) run code below.

```
# Slip1 Q1 - Apriori (min_support=0.25)  & Q2 - Iris scatter

# Q1

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent_patterns import apriori, association_rules

def load_transactions(path='groceries.csv'):

    try:

        df = pd.read_csv(path, header=None)

        transactions = df.fillna('').astype(str).values.tolist()

        return [[it.strip() for it in row if it.strip()] for row in transactions]

    except:

        df = pd.read_csv(path)

        if 'items' in df.columns:

            return df['items'].dropna().apply(lambda x:[i.strip() for i in x.split(',')]).tolist()

        with open(path,'r',encoding='utf-8') as f:

            return [[it.strip() for it in line.strip().split(',') if it.strip()] for line in f if line.strip()]


transactions = load_transactions('groceries.csv')

te = TransactionEncoder(); te_ary = te.fit(transactions).transform(transactions)

ohe = pd.DataFrame(te_ary, columns=te.columns_)

freq = apriori(ohe, min_support=0.25, use_colnames=True)

rules = association_rules(freq, metric='confidence', min_threshold=0.5)

display(freq.sort_values('support', ascending=False).head(20))

display(rules[['antecedents','consequents','support','confidence','lift']].head(20))


# Q2

import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder

iris = pd.read_csv('iris.csv')
```

```
le = LabelEncoder(); iris['species_num'] = le.fit_transform(iris['species'])
plt.figure(figsize=(7,5))
sc = plt.scatter(iris['petal_length'], iris['petal_width'], c=iris['species_num'])
plt.xlabel('Petal Length'); plt.ylabel('Petal Width'); plt.title('Iris petal length vs width')
cbar = plt.colorbar(sc, ticks=range(len(le.classes_))); cbar.ax.set_yticklabels(le.classes_)
plt.show()
```

Slip 2

Packages: pandas, scikit-learn, numpy
Run (3 steps): 1) pip install scikit-learn pandas 2) put house_price.csv, Wholesale customers data.csv 3) run.

```
# Slip2 Q1 - Simple Linear Regression (remove nulls)
from sklearn.linear_model import LinearRegression
df = pd.read_csv('house_price.csv').dropna()
if 'price' in df.columns:
    y = df['price']; X = df[['sqft_living']] if 'sqft_living' in df.columns else df.drop(columns=['price'])
else:
    X = df.iloc[:,:-1]; y = df.iloc[:,-1]
Xtr,Xte,ytr,yte = train_test_split(X,y,test_size=0.2,random_state=42)
model = LinearRegression().fit(Xtr,ytr)
print("Train R2:", model.score(Xtr,ytr),"Test R2:", model.score(Xte,yte))

# Slip2 Q2 - Agglomerative Clustering (Wholesale)
from sklearn.cluster import AgglomerativeClustering
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
dfw = pd.read_csv('Wholesale customers data.csv')
Xw = dfw.select_dtypes(include=[int,float]).fillna(dfw.mean())
Xs = StandardScaler().fit_transform(Xw)
best_k, best_score = None, -1
for k in range(2,7):
    labs = AgglomerativeClustering(n_clusters=k).fit_predict(Xs)
```

```
    s = silhouette_score(Xs, labs)
    if s>best_score: best_k, best_score = k, s
print("Best k:", best_k, "silhouette:", best_score)
dfw['cluster'] = AgglomerativeClustering(n_clusters=best_k).fit_predict(Xs)
display(dfw.groupby('cluster').mean())
```

Slip 3

Packages: pandas, scikit-learn
Run: 1) pip install scikit-learn pandas 2) put house_price.csv, crash.csv 3) run.

```
# Slip3 Q1 - Multiple Linear Regression (house)
from sklearn.linear_model import LinearRegression
df = pd.read_csv('house_price.csv').dropna()
X = df.drop(columns=['price']) if 'price' in df.columns else df.iloc[:,:-1]
y = df['price'] if 'price' in df.columns else df.iloc[:,-1]
Xtr,Xte,ytr,yte = train_test_split(X,y,test_size=0.2,random_state=1)
model = LinearRegression().fit(Xtr,ytr)
print("Train R2:", model.score(Xtr,ytr), "Test R2:", model.score(Xte,yte))

# Slip3 Q2 - Logistic Regression (crash.csv)
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score
dcr = pd.read_csv('crash.csv').dropna()
target = 'survived' if 'survived' in dcr.columns else dcr.columns[-1]
y = dcr[target]; X = pd.get_dummies(dcr.drop(columns=[target]), drop_first=True)
Xtr,Xte,ytr,yte = train_test_split(X,y,test_size=0.2,random_state=0)
clf = LogisticRegression(max_iter=1000).fit(Xtr,ytr)
pred = clf.predict(Xte)
print("Acc:", accuracy_score(yte,pred)); print(classification_report(yte,pred))
```

Slip 4

Packages: pandas, scikit-learn, matplotlib
Run: 1) install 2) put mall_customers.csv 3) run.

```
# Slip4 Q1 - KMeans (mall_customers)
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
dfm = pd.read_csv('mall_customers.csv')
if 'Annual Income (k$)' in dfm.columns and 'Spending Score (1-100)' in dfm.columns:
    X = dfm[['Annual Income (k$)','Spending Score (1-100)']]
else:
    X = dfm.iloc[:, [2,3]]
Xs = StandardScaler().fit_transform(X)
inertias=[]
for k in range(1,11):
    inertias.append(KMeans(n_clusters=k, random_state=0).fit(Xs).inertia_)
plt.plot(range(1,11), inertias, marker='o'); plt.title('Elbow'); plt.show()
k=5
dfm['cluster'] = KMeans(n_clusters=k, random_state=0).fit_predict(Xs)
display(dfm.groupby('cluster').mean())

# Slip4 Q2 - Simple Linear Regression (reuse Slip2 Q1)
```

Slip 5

Packages: pandas, scikit-learn
Run: 1) install 2) put FuelConsumption.csv, iris.csv 3) run.

```
# Slip5 Q1 - Multiple Linear Regression (FuelConsumption)
from sklearn.linear_model import LinearRegression
dfc = pd.read_csv('FuelConsumption.csv').dropna()
y_col = 'CO2EMISSIONS' if 'CO2EMISSIONS' in dfc.columns else dfc.columns[-1]
y = dfc[y_col]; X = dfc.drop(columns=[y_col])
Xtr,Xte,ytr,yte = train_test_split(X,y,test_size=0.25,random_state=42)
model = LinearRegression().fit(Xtr,ytr)
print("Test R2:", model.score(Xte,yte))
```

```
# Slip5 Q2 - KNN on iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
dfi = pd.read_csv('iris.csv')
X = dfi.drop(columns=['species']); y = LabelEncoder().fit_transform(dfi['species'])
Xtr,Xte,ytr,yte = train_test_split(X,y,test_size=0.2,random_state=0)
best_k, best_acc = 1,0
for k in range(1,11):
    knn = KNeighborsClassifier(n_neighbors=k).fit(Xtr,ytr)
    acc = accuracy_score(yte, knn.predict(Xte))
    if acc>best_acc: best_acc, best_k = acc, k
print("Best k:", best_k, "accuracy:", best_acc)
```

Slip 6

Packages: pandas, scikit-learn, matplotlib
Run: 1) install 2) put boston_houses.csv, employee_income.csv 3) run.

```
# Slip6 Q1 - Polynomial Regression (Boston)
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
try:
    from sklearn.datasets import load_boston
    b = load_boston()
    X = pd.DataFrame(b.data, columns=b.feature_names); y = pd.Series(b.target, name='PRICE')
except:
    dft = pd.read_csv('boston_houses.csv')
    X = dft.drop(columns=['Price']); y = dft['Price']
poly = PolynomialFeatures(degree=2, include_bias=False)
Xp = poly.fit_transform(X[['RM']] if 'RM' in X.columns else X)
```

```
Xtr,Xte,ytr,yte = train_test_split(Xp,y,test_size=0.2,random_state=0)
model = LinearRegression().fit(Xtr,ytr)
print("R2:", r2_score(yte, model.predict(Xte)))

# Slip6 Q2 - KMeans on employee income
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
dfe = pd.read_csv('employee_income.csv')
Xe = dfe.select_dtypes(include=[int,float]).dropna()
Xs_e = StandardScaler().fit_transform(Xe[['income']] if 'income' in Xe.columns else Xe)
inertias=[]; scores=[]
from sklearn.metrics import silhouette_score
for k in range(2,8):
    km = KMeans(n_clusters=k, random_state=0).fit(Xs_e)
    inertias.append(km.inertia_); scores.append(silhouette_score(Xs_e, km.labels_))
plt.plot(range(2,8), inertias, marker='o'); plt.title('Elbow'); plt.show()
best_k = scores.index(max(scores))+2
dfe['cluster'] = KMeans(n_clusters=best_k, random_state=0).fit_predict(Xs_e)
display(dfe.groupby('cluster').mean())
```

Slip 7

Packages: pandas, scikit-learn
Run: 1) install 2) put Salary_positions.csv, weather.csv 3) run.

```
# Slip7 Q1 - Salary predictions (level 11 & 12)
from sklearn.linear_model import LinearRegression
dfs = pd.read_csv('Salary_positions.csv').dropna()
X = dfs[['level']]; y = dfs['salary']
mod = LinearRegression().fit(X,y)
print("R2:", mod.score(X,y)); print("Pred L11:", mod.predict([[11]])[0], "L12:", mod.predict([[12]])[0])
```

Slip7 Q2 - Naive Bayes on weather

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
dfw = pd.read_csv('weather.csv')
target = 'play' if 'play' in dfw.columns else dfw.columns[-1]
y = dfw[target]; X = pd.get_dummies(dfw.drop(columns=[target]), drop_first=True)
Xtr,Xte,ytr,yte = train_test_split(X,y,test_size=0.2,random_state=0)
clf = GaussianNB().fit(Xtr,ytr)
pred = clf.predict(Xte)
print("Acc:", accuracy_score(yte,pred)); print(classification_report(yte,pred))
```

Slip 8

Packages: scikit-learn, pandas
Run: 1) install 2) no local CSV needed for 20 newsgroups; put tennis.csv 3) run.

Slip8 Q1 - 20 Newsgroups MultinomialNB

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
data = fetch_20newsgroups(subset='all', remove=('headers','footers','quotes'))
from sklearn.model_selection import train_test_split
Xtr,Xte,ytr,yte = train_test_split(data.data, data.target, test_size=0.2, random_state=42)
vec = TfidfVectorizer(stop_words='english', max_df=0.7)
Xtrv = vec.fit_transform(Xtr); Xtev = vec.transform(Xte)
clf = MultinomialNB().fit(Xtrv,ytr)
print("Acc:", accuracy_score(yte, clf.predict(Xtev)))
```

Slip8 Q2 - Decision Tree (Tennis)

```
from sklearn.tree import DecisionTreeClassifier, export_text
dft = pd.read_csv('tennis.csv')
X = pd.get_dummies(dft.drop(columns=['Play']))
```

```
y = dft['Play']

clf = DecisionTreeClassifier().fit(X,y)

print(export_text(clf, feature_names=list(X.columns)))
```

Slip 9

Packages: pandas, scikit-learn
Run: 1) install 2) boston_houses.csv, UniversalBank.csv 3) run.

```
# Slip9 Q1 - Ridge & Lasso (RM vs Price)

from sklearn.linear_model import Ridge, Lasso

from sklearn.metrics import mean_squared_error

dfb = pd.read_csv('boston_houses.csv')[['RM','Price']].dropna()

X = dfb[['RM']]; y = dfb['Price']

Xtr,Xte,ytr,yte = train_test_split(X,y,test_size=0.2,random_state=0)

ridge = Ridge(alpha=1.0).fit(Xtr,ytr); lasso = Lasso(alpha=0.1).fit(Xtr,ytr)

print("Ridge RM=5:", ridge.predict([[5]])[0], "Lasso RM=5:", lasso.predict([[5]])[0])

print("Ridge MSE:", mean_squared_error(yte, ridge.predict(Xte)), "Lasso MSE:",
mean_squared_error(yte, lasso.predict(Xte)))


# Slip9 Q2 - Linear SVM (UniversalBank)

from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler

dfu = pd.read_csv('UniversalBank.csv')

if 'ID' in dfu.columns: dfu = dfu.drop(columns=['ID'])

dfu = pd.get_dummies(dfu, drop_first=True)

target = 'Personal Loan' if 'Personal Loan' in dfu.columns else dfu.columns[-1]

y = dfu[target]; X = dfu.drop(columns=[target])

Xtr,Xte,ytr,yte = train_test_split(X,y,test_size=0.2,random_state=0)

sc = StandardScaler(); Xtr_sc = sc.fit_transform(Xtr); Xte_sc = sc.transform(Xte)

clf = SVC(kernel='linear').fit(Xtr_sc,ytr)

print("Accuracy:", clf.score(Xte_sc,yte))
```

Slip 10

Packages: pandas, scikit-learn
Run: 1) install 2) put iris.csv 3) run.

```
# Slip10 Q1 - PCA reduce iris to 2D
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
dfi = pd.read_csv('iris.csv')
X = dfi.drop(columns=['species'])
Xs = StandardScaler().fit_transform(X)
pca = PCA(n_components=2); X2 = pca.fit_transform(Xs)
res = pd.DataFrame(X2, columns=['PC1','PC2']); res['species'] = dfi['species']
display(res.head())

# Slip10 Q2 - Iris scatter: reuse Slip1 Q2
```

Slip 11

Packages: pandas, scikit-learn
Run: 1) install 2) boston_houses.csv and data_banknote_authentication.txt 3) run.

```
# Slip11 Q1 - Polynomial Regression (Boston) reuse Slip6 Q1

# Slip11 Q2 - Decision Tree on banknote authentication (UCI)
dfbn = pd.read_csv('data_banknote_authentication.txt', header=None)
X = dfbn.iloc[:,:4]; y = dfbn.iloc[:,4]
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
Xtr,Xte,ytr,yte = train_test_split(X,y,test_size=0.2,random_state=0)
clf = DecisionTreeClassifier().fit(Xtr,ytr)
pred = clf.predict(Xte)
print("Accuracy:", accuracy_score(yte,pred))
```

Slip 12

Packages: pandas, scikit-learn
Run: 1) install 2) iris.csv, Salary_positions.csv 3) run.

```
# Slip12 Q1 - KNN on iris (reuse Slip5 Q2)
# Slip12 Q2 - compare linear vs polynomial regression (Salary_positions)
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score
dfsp = pd.read_csv('Salary_positions.csv')
X = dfsp[['level']]; y = dfsp['salary']
from sklearn.linear_model import LinearRegression
lin = LinearRegression().fit(X,y)
poly = PolynomialFeatures(degree=2); Xp = poly.fit_transform(X)
poly_model = LinearRegression().fit(Xp,y)
print("Linear R2:", r2_score(y, lin.predict(X)), "Poly R2:", r2_score(y, poly_model.predict(Xp)))
print("Pred L11 linear:", lin.predict([[11]]), "poly:", poly_model.predict(poly.transform([[11]])))
```

Slip 13

Packages: pandas, numpy, scikit-learn, tensorflow
Run: 1) pip install tensorflow 2) put GOOG.csv 3) run.

```
# Slip13 Q1 - LSTM simple next-day prediction (GOOG.csv)
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
dfg = pd.read_csv('GOOG.csv', parse_dates=['Date']).sort_values('Date')
close = dfg['Close'].values.reshape(-1,1)
scaler = MinMaxScaler(); csc = scaler.fit_transform(close)
def create_seq(data, seq_len=60):
    X,y = [],[]
    for i in range(seq_len, len(data)):
        X.append(data[i-seq_len:i,0]); y.append(data[i,0])
    return np.array(X), np.array(y)
seq_len=60
X,y = create_seq(csc, seq_len)
```

X = X.reshape((X.shape[0], X.shape[1], 1))

split = int(0.8*len(X))

Xtr,Xte,ytr,yte = X[:split], X[split:], y[:split], y[split:]

model = Sequential([LSTM(50, return_sequences=True, input_shape=(seq_len,1)), LSTM(50), Dense(1)])

model.compile(optimizer='adam', loss='mse')

model.fit(Xtr,ytr, epochs=5, batch_size=32, validation_data=(Xte,yte))

pred = model.predict(Xte[-1].reshape(1,seq_len,1))

print("Pred next close:", scaler.inverse_transform(pred.reshape(-1,1))[0,0])

Slip13 Q2 - Simple Linear Regression (repeat Slip2 Q1)

Slip 14

Packages: tensorflow, numpy
Run: 1) pip install tensorflow 2) no CSV needed for MNIST 3) run.

Slip14 Q1 - CNN on MNIST (Keras dataset)

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

from tensorflow.keras.utils import to_categorical

(x_train,y_train),(x_test,y_test) = mnist.load_data()

x_train = x_train.reshape(-1,28,28,1)/255.0; x_test = x_test.reshape(-1,28,28,1)/255.0

y_train = to_categorical(y_train); y_test = to_categorical(y_test)

model = Sequential([Conv2D(32,(3,3),activation='relu', input_shape=(28,28,1)), MaxPooling2D((2,2)), Conv2D(64,(3,3),activation='relu'), MaxPooling2D((2,2)), Flatten(), Dense(64,activation='relu'), Dense(10,activation='softmax')])

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

model.fit(x_train,y_train, epochs=3, batch_size=128, validation_split=0.1)

print("Test acc:", model.evaluate(x_test,y_test, verbose=0)[1])

Slip14 Q2 - find & remove nulls

dfu = pd.read_csv('your_dataset.csv') # replace with actual file

print(dfu.isnull().sum())

dfu.dropna().to_csv('your_dataset_clean.csv', index=False)

print("Saved your_dataset_clean.csv")

Slip 15

Packages: pandas, scikit-learn, tensorflow
Run: 1) install 2) put house_price.csv 3) run.

Slip15 Q1 - ANN classify house price above/below mean

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

dfh = pd.read_csv('house_price.csv').dropna()

X = dfh.drop(columns=['price']); y = (dfh['price'] > dfh['price'].mean()).astype(int)

Xs = StandardScaler().fit_transform(X)

Xtr,Xte,ytr,yte = train_test_split(Xs,y,test_size=0.2,random_state=0)

model = Sequential([Dense(64, activation='relu', input_shape=(Xtr.shape[1],)), Dense(32, activation='relu'), Dense(1, activation='sigmoid')])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.fit(Xtr,ytr, epochs=20, batch_size=32, validation_data=(Xte,yte))

Slip15 Q2 - Multiple Linear Regression (reuse Slip3 Q1)

Slip 16

Packages: tensorflow, numpy
Run: 1) install 2) optional synthetic data 3) run.

Slip16 Q1 - two-layer NN (relu + sigmoid)

import numpy as np

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

X = np.random.rand(1000,10); y = (X.sum(axis=1) > 5).astype(int)

model = Sequential([Dense(64, activation='relu', input_shape=(10,)), Dense(1, activation='sigmoid')])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.fit(X,y, epochs=10, batch_size=32)

Slip16 Q2 - Simple linear regression for Boston (reuse Slip6 Q1)

Slip 17

Packages: scikit-learn, xgboost (optional), pandas
Run: 1) pip install xgboost if you want XGBoost 2) put pima-indians-diabetes.csv 3) run.

```
# Slip17 Q1 - Ensembles on Pima diabetes (RF, Voting, Stacking)
from sklearn.ensemble import RandomForestClassifier, VotingClassifier, StackingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
try:
    from xgboost import XGBClassifier; xgb_available=True
except:
    xgb_available=False
dfp = pd.read_csv('pima-indians-diabetes.csv')
X = dfp.drop(columns=['Outcome']); y = dfp['Outcome']
Xtr,Xte,ytr,yte = train_test_split(X,y,test_size=0.2,random_state=0)
rf = RandomForestClassifier(n_estimators=100, random_state=0).fit(Xtr,ytr)
print("RF acc:", accuracy_score(yte, rf.predict(Xte)))
estimators = [('rf', rf), ('lr', LogisticRegression(max_iter=1000))]
if xgb_available:
    xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss').fit(Xtr,ytr);
estimators.append(('xgb', xgb))
vc = VotingClassifier(estimators=estimators, voting='hard').fit(Xtr,ytr)
print("Voting acc:", accuracy_score(yte, vc.predict(Xte)))
stack = StackingClassifier(estimators=estimators, final_estimator=LogisticRegression()).fit(Xtr,ytr)
print("Stacking acc:", accuracy_score(yte, stack.predict(Xte)))
# Slip17 Q2 - Multiple Linear Regression (reuse Slip3 Q1)
```

Slip 18

Packages: pandas, scikit-learn, matplotlib
Run: 1) install 2) put diabetes.csv 3) run.

```
# Slip18 Q1 - KMeans on diabetes dataset
```

```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
dfd = pd.read_csv('diabetes.csv')
X = dfd.select_dtypes(include=[int,float]).dropna()
Xs = StandardScaler().fit_transform(X)
inertias=[]
for k in range(2,8):
    inertias.append(KMeans(n_clusters=k, random_state=0).fit(Xs).inertia_)
plt.plot(range(2,8), inertias, marker='o'); plt.title('Elbow'); plt.show()
k=3
dfd['cluster'] = KMeans(n_clusters=k, random_state=0).fit_predict(Xs)
display(dfd.groupby('cluster').mean())

# Slip18 Q2 - Polynomial regression on Salary_positions (reuse Slip12 Q2)
```

Slip 19

Packages: pandas, scikit-learn
Run: 1) install 2) put Salary_positions.csv (and weather.csv if needed) 3) run.

```
# Slip19 Q1 - simple vs polynomial regression (Salary_positions)
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score
dfs = pd.read_csv('Salary_positions.csv')
X = dfs[['level']]; y = dfs['salary']
from sklearn.linear_model import LinearRegression
lin = LinearRegression().fit(X,y)
poly = PolynomialFeatures(degree=2); Xp = poly.fit_transform(X)
poly_model = LinearRegression().fit(Xp,y)
print("Linear R2:", r2_score(y, lin.predict(X)), "Poly R2:", r2_score(y, poly_model.predict(Xp)))

# Slip19 Q2 - Naive Bayes on weather (reuse Slip7 Q2)
```

Slip 20

Packages: pandas, scikit-learn
Run: 1) install 2) put boston_houses.csv, tennis.csv 3) run.

Slip20 Q1 - Ridge & Lasso (reuse Slip9 Q1)

Slip20 Q2 - Decision Tree (tennis) reuse Slip8 Q2

Slip 21

Packages: pandas, scikit-learn
Run: 1) install 2) put house_price.csv, Salary_positions.csv 3) run.

Slip21 Q1 - Multiple Linear Regression (reuse Slip3 Q1)

Slip21 Q2 - Salary regression (reuse Slip7 Q1)

Slip 22

Packages: pandas, mlxtend, scikit-learn
Run: 1) install 2) put house_price.csv, groceries.csv 3) run.

Slip22 Q1 - Simple Linear Regression (reuse Slip2 Q1)

Slip22 Q2 - Apriori groceries (reuse Slip1 Q1)

Slip 23

Packages: pandas, scikit-learn
Run: 1) install 2) put Salary_positions.csv, your dataset 3) run.

Slip23 Q1 - simple vs polynomial regression (reuse Slips 12/19)

Slip23 Q2 - find & remove nulls

dfu = pd.read_csv('your_dataset.csv')

print(dfu.isnull().sum())

dfu_clean = dfu.dropna()

dfu_clean.to_csv('your_dataset_clean.csv', index=False)

print("Saved your_dataset_clean.csv")

Slip 24

Packages: pandas, scikit-learn
Run: 1) install 2) put diabetes_indian.csv 3) run.

```
# Slip24 Q1 - KNN optimal K (Indian diabetes)
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
dfi = pd.read_csv('diabetes_indian.csv')
X = dfi.drop(columns=['Outcome']) if 'Outcome' in dfi.columns else dfi.iloc[:,:-1]
y = dfi['Outcome'] if 'Outcome' in dfi.columns else dfi.iloc[:,-1]
Xs = StandardScaler().fit_transform(X)
Xtr,Xte,ytr,yte = train_test_split(Xs,y,test_size=0.2,random_state=0)
best_k, best_acc = 1,0
for k in range(1,21):
    knn = KNeighborsClassifier(n_neighbors=k).fit(Xtr,ytr)
    acc = accuracy_score(yte, knn.predict(Xte))
    if acc>best_acc: best_acc, best_k = acc, k
print("Best k:", best_k, "acc:", best_acc)

# Slip24 Q2 - Apriori (reuse Slip1 Q1)
```

Slip 25

Packages: pandas, scikit-learn
Run: 1) install 2) put house_price.csv, Salary_positions.csv 3) run.

```
# Slip25 Q1 - Multiple Linear Regression (reuse Slip3 Q1)
# Slip25 Q2 - Salary regression (reuse Slip7 Q1)
```

Slip 26

Packages: scikit-learn, pandas
Run: 1) install 2) put iris.csv 3) run.

```
# Slip26 Q1 - 20 newsgroups classification (reuse Slip8 Q1)
# Slip26 Q2 - SVM kernels comparison on iris
from sklearn import svm
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
```

```
dfi = pd.read_csv('iris.csv')
X = dfi.drop(columns=['species']); y = LabelEncoder().fit_transform(dfi['species'])
Xtr,Xte,ytr,yte = train_test_split(X,y,test_size=0.2,random_state=0)
for kernel in ['linear','rbf','poly','sigmoid']:
    clf = svm.SVC(kernel=kernel).fit(Xtr,ytr)
    print(kernel, "accuracy:", accuracy_score(yte, clf.predict(Xte)))
```

Slip 27

Packages: pandas, scikit-learn
Run: 1) install 2) put iris.csv, employee CSV 3) run.

```
# Slip27 Q1 - PCA reduce iris then classify
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
dfi = pd.read_csv('iris.csv')
X = dfi.drop(columns=['species']); y = dfi['species']
Xs = StandardScaler().fit_transform(X)
pca = PCA(n_components=2); X2 = pca.fit_transform(Xs)
Xtr,Xte,ytr,yte = train_test_split(X2,y,test_size=0.2,random_state=0)
clf = RandomForestClassifier().fit(Xtr,ytr)
print("Acc:", clf.score(Xte,yte))

# Slip27 Q2 - KMeans employees (reuse Slip6 Q2)
```

Slip 28

Packages: pandas, scikit-learn
Run: 1) install 2) put data.csv 3) run.

```
# Slip28 Q1 - Min-Max scaling
from sklearn.preprocessing import MinMaxScaler
dfd = pd.read_csv('data.csv')
num = dfd.select_dtypes(include=[int,float])
sc = MinMaxScaler(); scaled = sc.fit_transform(num)
```

```
scaled_df = pd.DataFrame(scaled, columns=num.columns)
display(scaled_df.head())

# Slip28 Q2 - Standard scaling
from sklearn.preprocessing import StandardScaler
sc2 = StandardScaler(); scaled2 = sc2.fit_transform(num)
scaled2_df = pd.DataFrame(scaled2, columns=num.columns)
display(scaled2_df.head())
```

Slip 29

Packages: pandas, scikit-learn
Run: 1) install 2) put data.csv with target 3) run.

```
# Slip29 Q1 - Decision Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
dfd = pd.read_csv('data.csv')
X = dfd.drop(columns=['target']); y = dfd['target']
Xtr,Xte,ytr,yte = train_test_split(X,y,test_size=0.2,random_state=0)
clf = DecisionTreeClassifier().fit(Xtr,ytr); pred = clf.predict(Xte)
print("Accuracy:", accuracy_score(yte,pred))

# Slip29 Q2 - Random Forest
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100).fit(Xtr,ytr)
print("RF accuracy:", rf.score(Xte,yte))
```

Slip 30

Packages: pandas, mlxtend
Run: 1) install 2) put groceries.csv 3) run.

```
# Slip30 Q1 - Apriori (reuse Slip1 Q1)

# Slip30 Q2 - FP-Growth
from mlxtend.frequent_patterns import fpgrowth
```

```
def load_transactions(path='groceries.csv'):
    try:
        df = pd.read_csv(path, header=None)
        transactions = df.fillna('').astype(str).values.tolist()
        return [[it.strip() for it in row if it.strip()] for row in transactions]
    except:
        df = pd.read_csv(path)
        if 'items' in df.columns:
            return df['items'].dropna().apply(lambda x:[i.strip() for i in x.split(',')]).tolist()
        with open(path,'r',encoding='utf-8') as f:
            return [[it.strip() for it in line.strip().split(',') if it.strip()] for line in f if line.strip()]
transactions = load_transactions('groceries.csv')
from mlxtend.preprocessing import TransactionEncoder
te = TransactionEncoder(); te_ary = te.fit(transactions).transform(transactions)
ohe = pd.DataFrame(te_ary, columns=te.columns_)
freq = fpgrowth(ohe, min_support=0.25, use_colnames=True)
display(freq.sort_values('support', ascending=False).head(30))
```