Classification Assignments 1. Medical Diagnosis Using SVM • Scenario: A hospital wants to automate the diagnosis of skin cancer using image data. • Task: Train an SVM to classify skin lesions as benign or malignant using image features. provide code only

```
import numpy as np
from sklearn import svm
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

np.random.seed(0)
X = np.random.randn(100, 2)
y = (X[:, 0] + X[:, 1] > 0.5).astype(int)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

model = svm.SVC(kernel='rbf', C=1.0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Model Accuracy:", accuracy_score(y_test, y_pred))
```

✅ **Recommended** — clean, short, and works without any dataset file.

2. Loan Default Prediction with Decision Trees • Scenario: A bank needs to assess loan applicants for risk of default. • Task: Use decision trees to classify applicants based on income, credit score, and employment history.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

np.random.seed(0)
data = pd.DataFrame({
    'income': np.random.randint(20000, 120000, 100),
    'credit_score': np.random.randint(300, 850, 100),
    'employment_years': np.random.randint(0, 30, 100)
})
data['default'] = ((data['income'] < 50000) & (data['credit_score'] <
600) & (data['employment_years'] < 5)).astype(int)

X = data[['income', 'credit_score', 'employment_years']]
y = data['default']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = DecisionTreeClassifier(criterion='entropy', random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Model Accuracy:", accuracy_score(y_test, y_pred))
```

3. Disease Classification with KNN • Scenario: A clinic wants to predict whether a patient has diabetes based on health metrics. • Task: Apply KNN to classify patients using features like glucose level, BMI, and age.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

np.random.seed(0)
data = pd.DataFrame({
    'glucose': np.random.randint(70, 200, 100),
    'bmi': np.random.uniform(18, 40, 100),
    'age': np.random.randint(20, 80, 100)
})
data['diabetes'] = ((data['glucose'] > 130) | (data['bmi'] >
30)).astype(int)

X = data[['glucose', 'bmi', 'age']]
y = data['diabetes']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Model Accuracy:", accuracy_score(y_test, y_pred))
```

4. Email Spam Detection Using Naive Bayes • Scenario: An email service provider wants to filter spam messages. • Task: Use Naive Bayes to classify emails based on word frequency and metadata.

ChatGPT said:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score
from sklearn.feature_extraction.text import CountVectorizer

np.random.seed(0)
emails = [
    "Win money now", "Limited offer", "Meeting schedule", "Project
deadline",
    "Free entry in contest", "Lunch at 1pm", "Discount coupons", "Team
update",
    "Earn from home", "Weekly report"
]
labels = [1,1,0,0,1,0,1,0,1,0]

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(emails)
y = np.array(labels)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = MultinomialNB()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

```
print("Model Accuracy:", accuracy_score(y_test, y_pred))
```

5. Customer Sentiment Analysis with SVM • Scenario: An e-commerce company wants to analyze customer reviews. • Task: Train an SVM to classify reviews as positive or negative using text data.

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.feature_extraction.text import CountVectorizer

np.random.seed(0)
reviews = [
    "Great product", "Very bad experience", "Loved it", "Not worth the
money",
    "Excellent service", "Terrible quality", "Highly recommend",
"Disappointed",
    "Amazing purchase", "Worst item ever"
]
labels = [1,0,1,0,1,0,1,0,1,0]

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(reviews)
y = np.array(labels)

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = SVC(kernel='linear', C=1.0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Model Accuracy:", accuracy_score(y_test, y_pred))
```

Regression Assignments

6. House Price Prediction Using Linear Regression • Scenario: A real estate firm wants to estimate property prices. • Task: Use linear regression to predict prices based on location, size, and amenities.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

np.random.seed(0)
data = pd.DataFrame({
    'location_score': np.random.randint(1, 10, 100),
    'size_sqft': np.random.randint(500, 4000, 100),
    'amenities': np.random.randint(1, 5, 100)
})
data['price'] = data['location_score'] * 50000 + data['size_sqft'] *
200 + data['amenities'] * 30000 + np.random.randint(10000, 50000, 100)

X = data[['location_score', 'size_sqft', 'amenities']]
y = data['price']

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("R2 Score:", r2_score(y_test, y_pred))
```

7. Churn Probability with Logistic Regression • Scenario: A telecom company wants to predict customer churn. • Task: Apply logistic regression to model churn likelihood using usage and billing data.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

np.random.seed(0)
data = pd.DataFrame({
    'monthly_usage': np.random.randint(50, 500, 100),
    'billing_amount': np.random.randint(300, 2000, 100),
    'customer_tenure': np.random.randint(1, 60, 100)
})
data['churn'] = ((data['monthly_usage'] < 150) &
(data['billing_amount'] > 1000) & (data['customer_tenure'] <
12)).astype(int)

X = data[['monthly_usage', 'billing_amount', 'customer_tenure']]
y = data['churn']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Model Accuracy:", accuracy_score(y_test, y_pred))
```

8. Stock Price Forecasting with Least Squares • Scenario: A financial analyst wants to forecast stock prices. • Task: Use least squares regression on historical stock data to predict future trends.

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

np.random.seed(0)
days = np.arange(1, 101)
prices = 100 + 2*days + np.random.randint(-10, 10, 100)

X = days.reshape(-1, 1)
y = prices

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("R2 Score:", r2_score(y_test, y_pred))
```

9. Crop Yield Estimation Using Linear Regression • Scenario: An agricultural agency wants to predict crop yields. • Task: Model yield based on rainfall, temperature, and soil quality.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

np.random.seed(0)
data = pd.DataFrame({
    'rainfall': np.random.randint(50, 300, 100),
    'temperature': np.random.randint(15, 35, 100),
    'soil_quality': np.random.randint(1, 10, 100)
})
data['yield'] = data['rainfall']*2 + data['temperature']*10 +
data['soil_quality']*50 + np.random.randint(-20, 20, 100)

X = data[['rainfall', 'temperature', 'soil_quality']]
y = data['yield']

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("R2 Score:", r2_score(y_test, y_pred))
```

10. Heart Disease Risk Prediction with Logistic Regression • Scenario: A health startup wants to assess heart disease risk. • Task: Use logistic regression on patient data to predict risk levels.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

np.random.seed(0)
data = pd.DataFrame({
    'age': np.random.randint(30, 80, 100),
    'cholesterol': np.random.randint(150, 300, 100),
    'blood_pressure': np.random.randint(80, 180, 100)
})
data['heart_disease'] = ((data['age'] > 50) & (data['cholesterol'] >
240) & (data['blood_pressure'] > 140)).astype(int)

X = data[['age', 'cholesterol', 'blood_pressure']]
y = data['heart_disease']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Model Accuracy:", accuracy_score(y_test, y_pred))
```

Random Forest Assignments

11. Fraud Detection in Financial Transactions • Scenario: A bank wants to detect fraudulent transactions. • Task: Use random forest to classify transactions as legitimate or fraudulent.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

np.random.seed(0)
data = pd.DataFrame({
    'amount': np.random.randint(10, 10000, 100),
    'transaction_time': np.random.randint(0, 24, 100),
    'location_score': np.random.randint(1, 10, 100)
})
data['fraud'] = ((data['amount'] > 7000) & (data['location_score'] <
4) & (data['transaction_time'] > 20)).astype(int)

X = data[['amount', 'transaction_time', 'location_score']]
y = data['fraud']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Model Accuracy:", accuracy_score(y_test, y_pred))
```

12. Employee Attrition Prediction • Scenario: An HR department wants to reduce employee turnover. • Task: Train a random forest model to predict attrition using job satisfaction and performance data.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

np.random.seed(0)
data = pd.DataFrame({
    'job_satisfaction': np.random.randint(1, 10, 100),
    'performance_score': np.random.randint(1, 10, 100),
    'years_at_company': np.random.randint(1, 15, 100)
})
data['attrition'] = ((data['job_satisfaction'] < 4) &
(data['performance_score'] < 5) & (data['years_at_company'] <
3)).astype(int)

X = data[['job_satisfaction', 'performance_score',
'years_at_company']]
y = data['attrition']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Model Accuracy:", accuracy_score(y_test, y_pred))
```

13. Air Quality Classification • Scenario: A city council wants to monitor pollution levels. • Task: Use random forest to classify AQI levels based on sensor data.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

np.random.seed(0)
data = pd.DataFrame({
    'pm2_5': np.random.randint(10, 300, 100),
    'pm10': np.random.randint(20, 400, 100),
    'no2': np.random.randint(5, 200, 100),
    'so2': np.random.randint(2, 100, 100)
})
data['aqi_level'] = pd.cut(data['pm2_5'], bins=[0, 50, 100, 200, 300],
labels=[0, 1, 2, 3]).astype(int)

X = data[['pm2_5', 'pm10', 'no2', 'so2']]
y = data['aqi_level']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Model Accuracy:", accuracy_score(y_test, y_pred))
```

14. Product Recommendation System • Scenario: An online store wants to personalize product suggestions. • Task: Use random forest to predict user preferences based on browsing and purchase history.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

np.random.seed(0)
data = pd.DataFrame({
    'browsing_time': np.random.randint(1, 60, 100),
    'pages_viewed': np.random.randint(1, 20, 100),
    'past_purchases': np.random.randint(0, 10, 100)
})
data['preference'] = ((data['browsing_time'] > 30) &
(data['pages_viewed'] > 10) | (data['past_purchases'] >
5)).astype(int)

X = data[['browsing_time', 'pages_viewed', 'past_purchases']]
y = data['preference']

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

print("Model Accuracy:", accuracy_score(y_test, y_pred))
```

Clustering Assignments (K-Means)

15. Customer Segmentation for Marketing • Scenario: A retail chain wants to target promotions effectively. • Task: Use K-means to segment customers by spending behavior and demographics.

```
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

np.random.seed(0)
data = pd.DataFrame({
    'age': np.random.randint(18, 65, 100),
    'annual_income': np.random.randint(20000, 150000, 100),
    'spending_score': np.random.randint(1, 100, 100)
})

scaler = StandardScaler()
X_scaled = scaler.fit_transform(data)

kmeans = KMeans(n_clusters=3, random_state=42)
data['cluster'] = kmeans.fit_predict(X_scaled)

print(data.head())
```

16. Document Clustering for News Articles • Scenario: A media company wants to organize articles by topic. • Task: Apply K-means to cluster articles using TF-IDF features.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.cluster import KMeans
import numpy as np

np.random.seed(0)
documents = [
    "Politics and government news",
    "Sports events and football updates",
    "Technology and AI advancements",
    "Elections and political debates",
    "New gadgets and software releases",
    "Cricket and Olympic highlights"
]

vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(documents)
```

```
kmeans = KMeans(n_clusters=3, random_state=42)
labels = kmeans.fit_predict(X)

for doc, label in zip(documents, labels):
    print(label, "-", doc)
```

17. Image Compression Using K-Means • Scenario: A mobile app needs to reduce image sizes. • Task: Use K-means to compress images by clustering pixel colors.

```
import numpy as np
from sklearn.cluster import KMeans
from skimage import io

image = io.imread('image.jpg')
pixels = image.reshape(-1, 3)

kmeans = KMeans(n_clusters=8, random_state=42)
kmeans.fit(pixels)
compressed_pixels =
kmeans.cluster_centers_[kmeans.labels_].astype('uint8')
compressed_image = compressed_pixels.reshape(image.shape)

io.imsave('compressed_image.jpg', compressed_image)
print("Image compressed and saved.")
```

18. Traffic Pattern Analysis • Scenario: A smart city project wants to optimize traffic flow. • Task: Cluster GPS data to identify congestion zones and peak hours.

```
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans

np.random.seed(0)
data = pd.DataFrame({
    'latitude': np.random.uniform(18.4, 19.2, 100),
    'longitude': np.random.uniform(72.7, 73.1, 100),
    'hour': np.random.randint(0, 24, 100)
})
```

```
kmeans = KMeans(n_clusters=3, random_state=42)
data['cluster'] = kmeans.fit_predict(data[['latitude', 'longitude',
'hour']])

print(data.head())
```

Association Rule Mining Assignments

19. Market Basket Analysis for Retail • Scenario: A supermarket wants to optimize product placement. • Task: Use Apriori algorithm to find frequent itemsets and generate association rules.

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

data = [
    ['milk', 'bread', 'eggs'],
    ['milk', 'bread'],
    ['milk', 'eggs'],
    ['bread', 'butter'],
    ['milk', 'bread', 'butter', 'eggs']
]

df = pd.DataFrame(data)
encoded = pd.get_dummies(df.stack()).groupby(level=0).sum()
frequent_itemsets = apriori(encoded, min_support=0.4,
use_colnames=True)
rules = association_rules(frequent_itemsets, metric='lift',
min_threshold=1.0)

print(rules[['antecedents', 'consequents', 'support', 'confidence',
'lift']])
```

20. Web Navigation Pattern Mining • Scenario: A website wants to improve user experience. • Task: Apply association rule mining to discover common navigation paths.

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

data = [
    ['Home', 'Products', 'Cart', 'Checkout'],
    ['Home', 'Products', 'About'],
    ['Home', 'Products', 'Cart'],
    ['Products', 'Cart', 'Checkout'],
    ['Home', 'About', 'Contact']
]

df = pd.DataFrame(data)
encoded = pd.get_dummies(df.stack()).groupby(level=0).sum()
frequent_itemsets = apriori(encoded, min_support=0.3,
use_colnames=True)
rules = association_rules(frequent_itemsets, metric='confidence',
min_threshold=0.5)

print(rules[['antecedents', 'consequents', 'support', 'confidence',
'lift']])
```

21. Medical Prescription Pattern Discovery • Scenario: A hospital wants to analyze co-prescribed drugs. • Task: Use association rules to find frequent drug combinations.

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

data = [
    ['Paracetamol', 'Ibuprofen'],
    ['Paracetamol', 'Cough Syrup'],
    ['Ibuprofen', 'Antibiotic'],
    ['Paracetamol', 'Ibuprofen', 'Antibiotic'],
    ['Cough Syrup', 'Antibiotic']
]

df = pd.DataFrame(data)
encoded = pd.get_dummies(df.stack()).groupby(level=0).sum()
frequent_itemsets = apriori(encoded, min_support=0.3,
use_colnames=True)
rules = association_rules(frequent_itemsets, metric='lift',
min_threshold=1.0)
```

```
print(rules[['antecedents', 'consequents', 'support', 'confidence',
'lift']])
```

22. Online Course Recommendation • Scenario: An ed-tech platform wants to suggest relevant courses. • Task: Mine user enrollment data to find course associations.

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

data = [
    ['Python Basics', 'Data Science'],
    ['Web Development', 'JavaScript'],
    ['Python Basics', 'Machine Learning'],
    ['Data Science', 'Deep Learning', 'AI Fundamentals'],
    ['Web Development', 'ReactJS']
]

df = pd.DataFrame(data)
encoded = pd.get_dummies(df.stack()).groupby(level=0).sum()
frequent_itemsets = apriori(encoded, min_support=0.3,
use_colnames=True)
rules = association_rules(frequent_itemsets, metric='lift',
min_threshold=1.0)

print(rules[['antecedents', 'consequents', 'support', 'confidence',
'lift']])
```

23. Retail Loss Prevention • Scenario: A store wants to detect suspicious buying patterns. • Task: Use association rules to identify unusual item combinations linked to theft.

```
import pandas as pd
from mlxtend.frequent_patterns import apriori, association_rules

data = [
    ['Razor', 'Shaving Cream', 'Wallet'],
    ['Bread', 'Butter'],
    ['Razor', 'Perfume', 'Chocolates'],
    ['Perfume', 'Wallet', 'Watch'],
    ['Bread', 'Milk'],
    ['Razor', 'Watch']
]

df = pd.DataFrame(data)
encoded = pd.get_dummies(df.stack()).groupby(level=0).sum()
frequent_itemsets = apriori(encoded, min_support=0.2,
use_colnames=True)
rules = association_rules(frequent_itemsets, metric='lift',
min_threshold=1.0)

print(rules[['antecedents', 'consequents', 'support', 'confidence',
'lift']])
```

24. Predictive Maintenance Using Classification and Regression • Scenario: A manufacturing plant wants to predict machine failures. • Task: Use classification to detect failure types and regression to estimate time to-failure.

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier,
RandomForestRegressor
from sklearn.metrics import accuracy_score, r2_score

np.random.seed(0)
data = pd.DataFrame({
    'temperature': np.random.randint(50, 100, 100),
    'vibration': np.random.randint(1, 10, 100),
    'pressure': np.random.randint(20, 80, 100)
})
data['failure_type'] = ((data['temperature'] > 80) &
(data['vibration'] > 5)).astype(int)
data['time_to_failure'] = 1000 - (data['temperature']*5 +
data['vibration']*20 + data['pressure']*3) +
np.random.randint(-50,50,100)

X = data[['temperature', 'vibration', 'pressure']]
```

```
y_class = data['failure_type']
y_reg = data['time_to_failure']

X_train, X_test, y_train_class, y_test_class = train_test_split(X,
y_class, test_size=0.2, random_state=42)
X_train_r, X_test_r, y_train_reg, y_test_reg = train_test_split(X,
y_reg, test_size=0.2, random_state=42)

clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train_class)
y_pred_class = clf.predict(X_test)
print("Classification Accuracy:", accuracy_score(y_test_class,
y_pred_class))

reg = RandomForestRegressor(n_estimators=100, random_state=42)
reg.fit(X_train_r, y_train_reg)
y_pred_reg = reg.predict(X_test_r)
print("Regression R2 Score:", r2_score(y_test_reg, y_pred_reg))
```

25. Healthcare Personalization Using Clustering and Association Rules • Scenario: A health provider wants to personalize treatment plans. • Task: Cluster patients by symptoms and mine treatment patterns using association rules.

```
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from mlxtend.frequent_patterns import apriori, association_rules

np.random.seed(0)
data = pd.DataFrame({
    'fever': np.random.randint(0, 2, 100),
    'cough': np.random.randint(0, 2, 100),
    'fatigue': np.random.randint(0, 2, 100),
    'headache': np.random.randint(0, 2, 100),
    'treatment': np.random.choice(['Medication A', 'Medication B',
'Therapy', 'Rest'], 100)
})
```

```
kmeans = KMeans(n_clusters=3, random_state=42)
data['cluster'] = kmeans.fit_predict(data[['fever', 'cough',
'fatigue', 'headache']])

encoded = pd.get_dummies(data[['cluster', 'treatment']])
frequent_itemsets = apriori(encoded, min_support=0.3,
use_colnames=True)
rules = association_rules(frequent_itemsets, metric='lift',
min_threshold=1.0)

print(rules[['antecedents', 'consequents', 'support', 'confidence',
'lift']])
```