

Task Category	Question Numbers	General Structure
Clustering Methods	8, 9-15, 11, 12 (KMeans), 15, 17-19 (DBSCAN), 20 (AgglomerativeClustering), 9, 12 (GaussianMixture), 4	Clustering: 1. Load and clean data 2. Preprocess data (standardize) 3. Choose and initialize the clustering model (KMeans, DBSCAN, Agglomerative, GaussianMixture) 4. Fit the model 5. Visualize clusters
Classification Methods	1, 2, 3, 5, 6, 7 (DecisionTreeClassifier), 3, 42, 43, 41 (SVC), 5, 48, 49, 50 (LogisticRegression), 6, 31, 32 (RandomForestClassifier), 46, 47 (MultinomialNB), 21, 22, 23, 24, 25, 26, 27, 28, 29	Classification: 1. Load and clean data 2. Preprocess data (standardize/encode) 3. Choose and initialize the classification model (DecisionTree, SVC, Logistic Regression, Random Forest, Naive Bayes) 4. Fit the model 5. Evaluate model using accuracy, confusion matrix, classification report
Dimensionality Reduction Methods	33, 34, 35, 36, 37, 38, 39, 40, 63, 64, 65, 30 (PCA), 34, 35, 36, 37, 38 (LinearDiscriminantAnalysis), 66, 67, 68, 69	Dimensionality Reduction: 1. Load and clean data 2. Preprocess data (standardize) 3. Choose and initialize the dimensionality reduction method (PCA or LDA) 4. Fit and transform the data 5. Visualize the reduced data in 2D or 3D
Regression Methods	51, 52, 53, 39, 40, 54, 55, 56, 57 (LinearRegression), 52 (PolynomialFeatures), 59 (Ridge), 60, 61, 58 (Lasso), 62	Regression: 1. Load and clean data 2. Preprocess data (standardize) 3. Choose and initialize the regression model (Linear, Polynomial, Ridge, Lasso) 4. Fit the model 5. Evaluate using metrics like MSE, R^2, and visualize predictions vs true values

Text Classification Methods	46, 47 (MultinomialNB)	Text Classification: 1. Load and clean data 2. Preprocess text data (tokenization, vectorization using TfidfVectorizer or CountVectorizer) 3. Initialize and train the MultinomialNB model 4. Evaluate using accuracy, confusion matrix, classification report
------------------------------------	------------------------	---

1. Classification Methods - DecisionTreeClassifier, SVC, LogisticRegression, RandomForestClassifier, MultinomialNB

python

Copy

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import cross_val_score

# Step 1: Load and clean data
df = pd.read_csv('your_dataset.csv') # Replace with your dataset path

# Handling missing values - either drop or fill missing values
df = df.dropna() # Drop rows with missing values
# Alternatively, you can fill missing values with a placeholder
# df['text_column'].fillna('Missing', inplace=True)

# Step 2: Preprocess data (feature extraction)
X = df[['feature1', 'feature2', 'feature3']].values # Independent
variables (replace with your actual feature names)
y = df['target'].values # Dependent variable (replace with your
actual target column)

# Split the data into training and test sets (80% training, 20%
testing)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Standardize the data (important for models like SVM or Logistic
Regression)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
# Step 3: Choose and initialize the model
# Replace the model based on your task (classification method)

# Example for Decision Tree (you can replace this with other models)
from sklearn.tree import DecisionTreeClassifier
# model = DecisionTreeClassifier(max_depth=3) # Decision Tree example

# For SVM: model = SVC()
# For Logistic Regression: model = LogisticRegression()
# For Random Forest: model = RandomForestClassifier()
# For Naive Bayes: model = MultinomialNB()

# Step 4: Fit the model
model.fit(X_train, y_train)

# Step 5: Make predictions and evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Step 6: Confusion matrix (to see classification performance)
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)

# Optional: Cross-validation for model evaluation
cv_scores = cross_val_score(model, X, y, cv=5)
print("Cross-validation scores:", cv_scores)

# Step 7: Visualizing the decision boundaries (if applicable for 2D
data)
# For multi-dimensional data, visualizing decision boundaries is more
complex
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap='viridis')
plt.title('Decision Boundary Visualization')
plt.show()
```

2. Clustering Methods - KMeans, DBSCAN, AgglomerativeClustering

python

Copy

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering

# Step 1: Load and clean data
df = pd.read_csv('your_dataset.csv') # Replace with your dataset path

# Handling missing values
df = df.dropna()

# Step 2: Preprocess data (feature extraction)
X = df[['feature1', 'feature2', 'feature3']].values # Replace with
your actual features

# Step 3: Standardize the data (important for clustering models)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 4: Choose and initialize the model
# Example for KMeans (you can replace this with other clustering
methods)
# model = KMeans(n_clusters=3, random_state=42)

# For DBSCAN: model = DBSCAN(eps=0.5, min_samples=5)
# For Agglomerative Clustering: model =
AgglomerativeClustering(n_clusters=3)

# Step 5: Fit the model
model.fit(X_scaled)

# Step 6: Get cluster labels (for clustering models)
labels = model.labels_ # For DBSCAN and AgglomerativeClustering
```

```
# Step 7: Visualize clusters (for 2D data)
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=labels, cmap='viridis')
plt.title('Clustering Visualization')
plt.show()
```

3. Dimensionality Reduction Methods - PCA, LDA

python

Copy

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler

# Step 1: Load and clean data
df = pd.read_csv('your_dataset.csv') # Replace with your dataset path

# Handling missing values
df = df.dropna()

# Step 2: Preprocess data (feature extraction)
X = df[['feature1', 'feature2', 'feature3']].values # Replace with
your features
y = df['target'].values # Replace with your target column

# Step 3: Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 4: Choose and initialize the dimensionality reduction model
# Example for PCA (you can replace this with LDA)
model = PCA(n_components=2)

# For LDA: model = LinearDiscriminantAnalysis(n_components=2)

# Step 5: Fit and transform the data
X_reduced = model.fit_transform(X_scaled)

# Step 6: Visualize reduced dimensions (for 2D data)
plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=y, cmap='viridis')
plt.title('Dimensionality Reduction Visualization')
```

```
plt.show()
```

4. Regression Methods - Linear, Polynomial, Ridge, Lasso

python

Copy

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures, StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso
from sklearn.metrics import mean_squared_error, r2_score

# Step 1: Load and clean data
df = pd.read_csv('your_dataset.csv') # Replace with your dataset path

# Handling missing values
df = df.dropna()

# Step 2: Preprocess data (feature extraction)
X = df[['feature1', 'feature2', 'feature3']].values # Independent
variables
y = df['target'].values # Dependent variable

# Step 3: Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Step 4: Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Step 5: Choose and initialize the model
# Example for Linear Regression (you can replace this with other
regression models)
model = LinearRegression()

# For Polynomial Regression: model = PolynomialFeatures(degree=2)
```

```
# For Ridge Regression: model = Ridge()
# For Lasso Regression: model = Lasso()

# Step 6: Fit the model
if isinstance(model, PolynomialFeatures):
    poly = model.fit_transform(X_train)
    model = LinearRegression() # Use Linear Regression after
    PolynomialFeatures
    model.fit(poly, y_train)
else:
    model.fit(X_train, y_train)

# Step 7: Make predictions and evaluate the model
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"MSE: {mse}")
print(f"R^2: {r2}")

# Step 8: Visualize results (for regression plots)
plt.scatter(X_test[:, 0], y_test, color='blue', label='True values')
plt.scatter(X_test[:, 0], y_pred, color='red', label='Predicted
values')
plt.title('Regression Model Prediction')
plt.legend()
plt.show()
```

5. Text Classification - MultinomialNB

python

Copy

```
# Import necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix

# Step 1: Load and clean data
df = pd.read_csv('your_text_dataset.csv') # Replace with your text
dataset path

# Handling missing values
df = df.dropna()

# Step 2: Preprocess text data
X = df['text_column'].values # Replace with your text column
y = df['target_column'].values # Replace with your target column

# Step 3: Convert text to numerical features using TfidfVectorizer
vectorizer = TfidfVectorizer(stop_words='english')
X_features = vectorizer.fit_transform(X)

# Step 4: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_features, y,
test_size=0.2, random_state=42)

# Step 5: Initialize and fit the Multinomial Naive Bayes model
model = MultinomialNB()
model.fit(X_train, y_train)

# Step 6: Make predictions and evaluate the model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
```

```
print(f"Accuracy: {accuracy}")
```

```
# Step 7: Confusion Matrix (for classification performance)
```

```
conf_matrix = confusion_matrix(y_test, y_pred)
```

```
print("Confusion Matrix:")
```

```
print(conf_matrix)
```