#1. A real estate company wants to develop a system that predicts house prices based on #square footage, number of bedrooms, and location. #Q: Identify the problem type and outline the step-by-step logic to solve it

In [1]:

```
#Goal: Build a model that takes (square footage, bedrooms, location) → predicts price.
#Features: Square footage (numeric), bedrooms (numeric), location (categorical)
#Handle Missing Values: Fill or remove missing entries.
#Encode Categorical Data: Convert Location into numbers (OneHotEncoding).
#Feature Scaling (optional): Scale features like square footage (StandardScaler/MinMaxSc
#Use train_test_split (e.g. 80% training, 20% testing).
#Start simple with Linear Regression.
#(Optional) Try advanced models like Decision Trees, Random Forest, or XGBoost for bette
#R² Score
#Feature engineering (e.g., add "price per square foot").
```

#2. A bank wants to build a model to detect fraudulent transactions by analyzing customer #spending behavior and transaction history. #Q: Identify the problem type and outline the step-by-step logic to solve it.

In [7]:

```
# 1. Collect Data - past transactions (amount, time, location, fraud label)
# 2. Preprocess - handle missing data, encode categorical columns, scale numeric feature
# 3. Split Data - use train_test_split() to create training and test sets
# 4. Train Model - start with LogisticRegression, then try RandomForest or XGBoost
# 5. Evaluate - check precision, recall, F1-score, and ROC-AUC (not just accuracy)
# 6. Improve - feature engineering (e.g. sudden spending spikes), hyperparameter tuning
# 7. Deploy - connect the model to the bank system for real-time fraud alerts
```

#3. A supermarket wants to segment its customers based on their shopping patterns to #provide personalized promotions. #Q: Identify the problem type and outline the step-by-step logic to solve it.

In [10]:

```
# 1. Collect Data - purchase history, product categories, frequency, spending amount
# 2. Preprocess - handle missing data, scale numeric features, encode categorical data
# 3. Choose Clustering Method - start with KMeans (or try Hierarchical/DBSCAN)
# 4. Find Optimal Clusters - use Elbow Method or Silhouette Score
# 5. Train the Model - fit KMeans on customer data
# 6. Label Segments - assign each customer to a cluster (e.g., Premium, Bargain Hunter)
# 7. Analyze & Act - create targeted promotions for each segment
```

#4. A company wants to estimate an employee's salary based on their years of experience, #job title, and education level. #Q: Identify the problem type and outline the step-by-step logic to solve it.

In [11]:

```
# 1. Collect Data - gather employee salary data (years of experience, job title, educati
# 2. Preprocess - handle missing values, encode job title and education, scale numerical
# 3. Split Data - use train_test_split() to create training and test sets
# 4. Choose Model - start with LinearRegression, try RandomForestRegressor or XGBoost fo
# 5. Train the Model - fit the model on the training set
# 6. Evaluate the Model - use MAE, RMSE, and R2 score to measure performance
# 7. Improve - add new features, tune hyperparameters for better predictions
# 8. Deploy - integrate into HR or payroll system for instant salary estimation
```

In [12]:

```
#5. An email provider wants to automatically classify incoming emails as spam or not spa
#based on their content and sender details.
#Q: Identify the problem type and outline the step-by-step logic to solve it.
```

In [13]:

```
# 1. Collect Data - gather emails labeled as spam or not spam (content + sender info)
# 2. Preprocess - clean text (remove punctuation, stopwords), tokenize, convert to numer
# 3. Split Data - use train_test_split() to create training and test sets
```

```
# 4. Choose Model - start with NaiveBayes (good for text), also try LogisticRegression o
# 5. Train the Model - fit on training data
# 6. Evaluate the Model - use accuracy, precision, recall, F1-score to measure performan
# 7. Improve - tune hyperparameters, add features (e.g. sender reputation)
# 8. Deploy - integrate into email system to filter spam in real-time
```

#6. A business wants to analyze customer reviews of its products and determine whether #the sentiment is positive or negative. #Q: Identify the problem type and outline the step-by-step logic to solve it.

In [15]:

```
# 1. Collect Data - gather product reviews labeled as positive or negative
# 2. Preprocess - clean text (remove punctuation, stopwords), tokenize, convert to numer
# 3. Split Data - use train_test_split() to create training and test sets
# 4. Choose Model - start with NaiveBayes (great for text), also try LogisticRegression
# 5. Train the Model - fit the model on the training data
# 6. Evaluate the Model - use accuracy, precision, recall, F1-score for performance
# 7. Improve - tune hyperparameters, add more labeled data, test advanced models (e.g. B
# 8. Deploy - integrate into business dashboard or review system to monitor sentiment in
```

#7. An insurance company wants to predict whether a customer is likely to file a claim in the #next year based on their driving history and demographics. #Q: Identify the problem type and outline the step-by-step logic to solve it.

In [16]:

```
# 1. Collect Data - gather past customer data (driving history, demographics, and whethe
# 2. Preprocess - handle missing values, encode categorical features (e.g. gender, regio
# 3. Split Data - use train_test_split() to create training and test sets
# 4. Choose Model - start with LogisticRegression, also try RandomForestClassifier or XG
# 5. Train the Model - fit the model on the training set
# 6. Evaluate the Model - use accuracy, precision, recall, F1-score, and ROC-AUC to meas
# 7. Improve - tune hyperparameters, add new features (e.g. driving score), balance clas
# 8. Deploy - integrate the model into the insurance system to predict claim risk in rea
```

#8. A streaming platform wants to recommend movies to users by grouping them based on #their viewing preferences and watch history. #Q: Identify the problem type and outline the step-by-step logic to solve it.

In [17]:

```
# 1. Collect Data - gather user viewing history (movies watched, ratings, genres)
# 2. Preprocess - clean data, handle missing values, encode movie genres or tags if need
# 3. Choose Approach -
#    - Collaborative filtering (users with similar tastes)
#    - Content-based filtering (recommend movies with similar attributes)
#    - Or a hybrid of both
# 4. Build User-Item Matrix - map users to the movies they have watched or rated
# 5. Train Model -
#    - For clustering users: use KMeans or other clustering
#    - For collaborative filtering: use matrix factorization (SVD)
# 6. Generate Recommendations - suggest movies to each user based on their cluster or ne
# 7. Evaluate - use metrics like precision@k, recall@k, or hit rate to see how good the
# 8. Deploy - integrate into the streaming platform so users see recommendations in real
```

#9. A hospital wants to predict the recovery time of patients after surgery based on their age, #medical history, and lifestyle habits. #Q: Identify the problem type and outline the step-by-step logic to solve it.

In [18]:

```
# 1. Collect Data - gather patient data (age, medical history, lifestyle habits, and act
# 2. Preprocess - handle missing values, encode categorical features (e.g. medical condi
# 3. Split Data - use train_test_split() to create training and test sets
# 4. Choose Model - start with LinearRegression, also try RandomForestRegressor or XGBoo
# 5. Train the Model - fit the model on the training data
# 6. Evaluate the Model - use MAE, RMSE, and R2 score to measure prediction performance
# 7. Improve - add new features (e.g. surgery type, post-op care quality), tune hyperpar
# 8. Deploy - integrate into hospital systems to estimate recovery times for new patient
```

In [ ]:

```

```