1 Design a complete k-NN based recommendation system for an online bookstore.

Include:

 Feature selection and preprocessing

 Distance metric selection and justification

 Algorithm implementation details

 Performance optimization techniques0

1. k-NN Based Recommendation System for an Online Bookstore

Feature Selection and Preprocessing

- **Features**:
  - User features: Age, Gender, Location, Reading Preferences (Genres)
  - Book features: Genre, Author, Publication Year, Ratings, Price
  - Interaction features: User ratings, Purchase history, Time spent on book pages
- **Preprocessing**:
  - Normalize numerical features (e.g., ratings, price) using Min-Max scaling or Z-score normalization.
  - Encode categorical features (e.g., genre, author) using one-hot encoding or label encoding.
  - Handle missing values by imputation (mean for numerical, mode for categorical).
  - Create a user-item interaction matrix where rows represent users and columns represent books, with values as ratings or purchase indicators.

Distance Metric Selection and Justification

- **Distance Metric**: Euclidean distance is commonly used for k-NN, but for recommendation systems, cosine similarity can be more effective as it measures the angle between two vectors, focusing on the orientation rather than magnitude.
- **Justification**: Cosine similarity is particularly useful in high-dimensional spaces (like user-item matrices) where the magnitude of the vectors can vary significantly. It helps in identifying users with similar preferences regardless of their rating scale.

Algorithm Implementation Details

1. **Data Preparation**: Create the user-item interaction matrix.
2. **Distance Calculation**: Compute pairwise distances (or similarities) between users or items using the selected metric.
3. **Recommendation Generation**:
   - For a target user, find the k nearest neighbors based on the distance metric.
   - Aggregate the ratings of the k neighbors to predict the target user's rating for unrated books (e.g., using weighted averages).
4. **Output**: Recommend the top N books with the highest predicted ratings.

Performance Optimization Techniques

- **Dimensionality Reduction**: Use techniques like PCA (Principal Component Analysis) to reduce the dimensionality of the user-item matrix, improving computation time.
- **Efficient Data Structures**: Use KD-trees or Ball trees for faster nearest neighbor searches.
- **Parallel Processing**: Implement parallel computation for distance calculations to speed up the recommendation process.
- **Hybrid Approaches**: Combine k-NN with collaborative filtering or content-based filtering to enhance recommendation quality.

2 Implement a locally weighted regression algorithm for predicting house prices.

Design should include:

- Kernel function selection
- Feature engineering
- Weight calculation method
- Cross-validation approach

2. Locally Weighted Regression Algorithm for Predicting House Prices

Kernel Function Selection

- **Kernel Function**: Gaussian (Radial Basis Function) kernel is commonly used.
- **Justification**: The Gaussian kernel provides a smooth weighting function that decreases with distance, allowing for localized fitting of the regression model.

Feature Engineering

- **Features**:
  - Square footage, Number of bedrooms, Number of bathrooms, Location (ZIP code), Age of the house, Lot size, etc.
- **Preprocessing**: Normalize numerical features and encode categorical features (e.g., location) using one-hot encoding.

Weight Calculation Method

- For a given query point x, the weight for each training point x_i is calculated using the Gaussian kernel: $w_i = \exp\left(-\frac{||x - x_i||^2}{2\tau^2}\right)$ where τ is the bandwidth parameter controlling the width of the kernel.

Cross-Validation Approach

- Use k-fold cross-validation to evaluate the model's performance. Split the dataset into k subsets, train the model on k-1 subsets, and validate on the remaining subset. Repeat this process k times and average the results to obtain a robust estimate of model performance.

3 create a case-based reasoning system for medical diagnosis with:

 Case representation structure

 Similarity metrics

 Case adaptation rules

 Case base maintenance strategy

3. Case-Based Reasoning System for Medical Diagnosis

Case Representation Structure

- Each case can be represented as a structured object containing:
  - Patient Information: Age, Gender, Medical History
  - Symptoms: List of symptoms presented by the patient
  - Diagnosis: The diagnosis made in the past
  - Treatment: Treatment prescribed for the diagnosis
  - Outcome: Result of the treatment

Similarity Metrics

- **Similarity Metrics**:
  - **Euclidean Distance**: For numerical features (e.g., age).
  - **Jaccard Similarity**: For categorical features (e.g., symptoms).
  - **Weighted Similarity**: Combine different metrics with weights based on feature importance.

Case Adaptation Rules

- Adaptation rules can include:
  - If a new case has symptoms similar to a past case but with different severity, adjust the treatment based on the severity.
  - If a patient has a different medical history, modify the diagnosis based on the historical outcomes of similar cases.

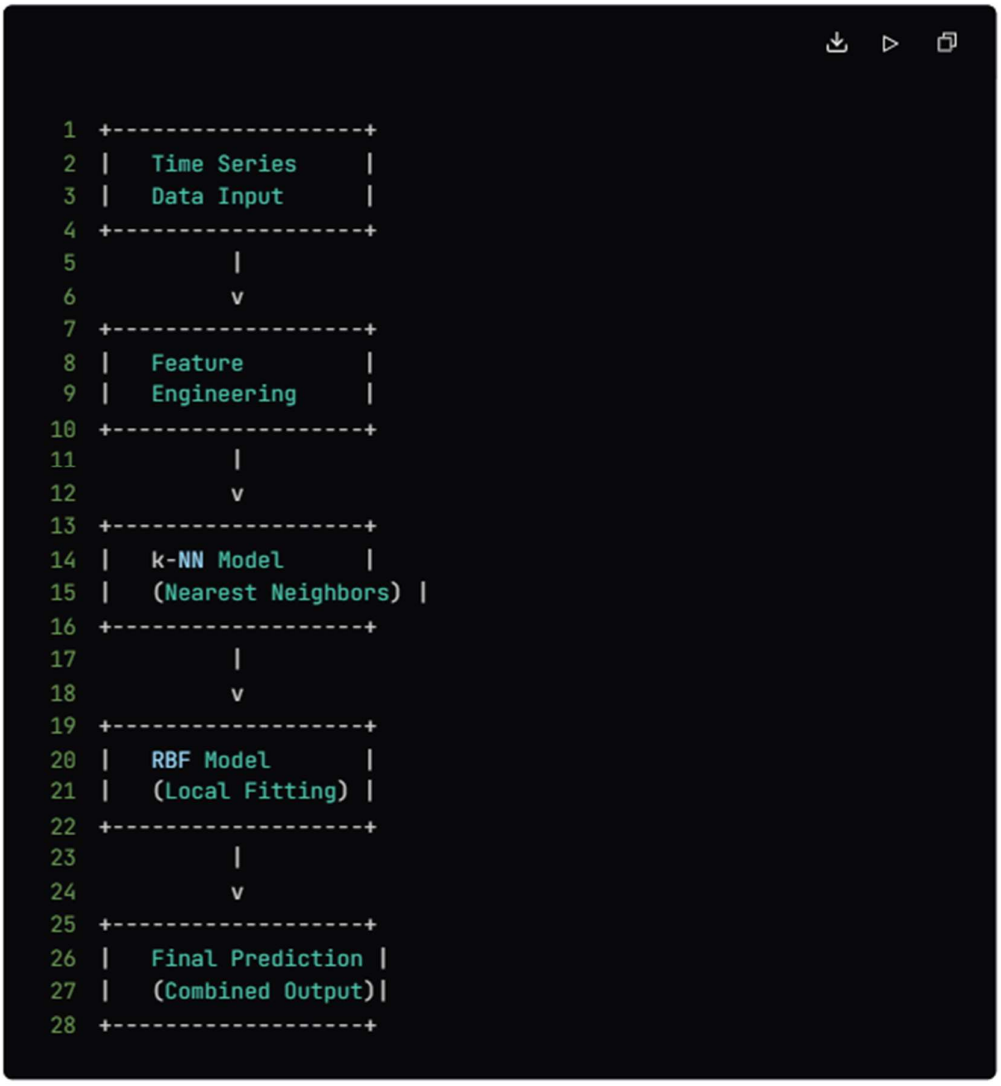
Case Base Maintenance Strategy

- **Case Base Maintenance**:
  - Regularly update the case base with new cases and outcomes.
  - Remove outdated or irrelevant cases based on a defined threshold (e.g., age of the case).
  - Implement a feedback loop where the system learns from new cases and improves its similarity metrics and adaptation rules.

4 Design a hybrid system combining k-NN and radial basis functions for time series prediction. Include architecture diagram and implementation details.

4. Hybrid System Combining k-NN and Radial Basis Functions for Time Series Prediction

Architecture Diagram



Implementation Details

1. **Data Preparation**: Preprocess the time series data (e.g., normalization, handling missing values).
2. **Feature Engineering**: Extract relevant features such as trends, seasonality, and lagged values.
3. **k-NN Model**:
   - Use k-NN to find similar historical time series segments.
   - Predict future values based on the average of the k nearest neighbors.
4. **RBF Model**:
   - For each of the k nearest neighbors, fit a local regression model using the RBF kernel.
   - Predict future values based on the local model.
5. **Final Prediction**: Combine the predictions from both models (e.g., averaging or weighted averaging) to produce the final output.

5i) Differentiate between generative and discriminative learning models. In a multinational company, there are people speaking different languages of their own mother tongue. The auto teller engine hosted by the company has a task of determining the language that someone is speaking by determining the linguistic differences without learning any language. Which learning model it has to follow ? Why ?

ii) For the application of your choice, explain the machine learning process indicating

1) Type of machine learning model

2) Dataset needed and how much ?

3) input parameters and expected outcome

4) Possible evaluation strategy.

5. Generative vs. Discriminative Learning Models

i) Differentiation

- **Generative Models**: These models learn the joint probability distribution $P(X, Y)$ and can generate new data points. Examples include Gaussian Mixture Models and Naive Bayes.
- **Discriminative Models**: These models learn the conditional probability $P(Y|X)$ and focus on the decision boundary between classes. Examples include Logistic Regression and Support Vector Machines.

Application in Multinational Company:

- The auto teller engine should use a **discriminative model**. Since it needs to classify the language being spoken based on linguistic features without prior knowledge of the languages, a discriminative approach will focus on the differences between languages based on the observed features.

ii) Machine Learning Process for Language Detection

1. **Type of Machine Learning Model**: Discriminative model (e.g., Logistic Regression, SVM).
2. **Dataset Needed**: A dataset containing audio samples of different languages, ideally with thousands of samples for each language to ensure robustness.
3. **Input Parameters and Expected Outcome**:
    - **Input Parameters**: Audio features (e.g., MFCCs, pitch, tone) extracted from the audio samples.
    - **Expected Outcome**: The predicted language for each audio sample.
4. **Possible Evaluation Strategy**:
    - Use k-fold cross-validation to evaluate model performance.
    - Metrics: Accuracy, Precision, Recall, F1-score to assess the model's performance in classifying languages correctly.

This comprehensive approach covers the design and implementation of various systems and models in machine learning, providing a solid foundation for practical applications.