CSCI316: Big Data Mining Techniques and Implementation

Comprehensive Question Bank

Question 1: k-Nearest Neighbors (kNN) Algorithm Implementation and Analysis (12 marks)

Part A (4 marks): Given the following dataset of mobile phone features and their classifications:

Phone	Battery Life (hours)	Screen Size (inches)	Price ($)	Class
A	24	5.5	300	Budget
B	18	6.1	800	Premium
C	20	5.8	450	Mid-range
D	30	5.2	250	Budget
E	15	6.5	1200	Premium

Classify a new phone X with features (22, 5.9, 400) using k=3. Show all distance calculations using Euclidean distance and explain your classification decision.

Part B (4 marks): Implement the kNN classifier function from scratch in Python. Your function should:

- Accept input data, training dataset, labels, and k value
- Calculate Euclidean distances
- Return the predicted class
- Handle ties appropriately

```python
def knn_classifier(input_point, training_data, labels, k):
    # Your implementation here
    pass
```

Part C (4 marks): Discuss the advantages and disadvantages of the kNN algorithm. Explain why kNN has "poor scalability for large datasets" and suggest two methods to improve its performance on big data.

Question 2: Data Preprocessing and Pipeline Construction (10 marks)

Part A (3 marks): You are working with a customer dataset that has the following issues:

- Missing values in the 'income' column (15% missing)
- Categorical variable 'education_level' with values: ['High School', 'Bachelor', 'Master', 'PhD']

- Numerical features with vastly different scales: age (20-80), income (20000-200000), credit_score (300-850)

Explain three different approaches to handle missing values and justify which approach you would choose for the income column.

Part B (4 marks): Create a complete preprocessing pipeline using scikit-learn that addresses all the issues mentioned in Part A. Your pipeline should include:

- Missing value imputation
- Categorical encoding
- Feature scaling
- Custom feature creation (create a new feature combining two existing ones)

```python
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
# Your pipeline implementation here
```

Part C (3 marks): Compare and contrast one-hot encoding vs. ordinal encoding for categorical variables. When would you use each approach? Provide specific examples.

Question 3: Model Selection and Evaluation Framework (11 marks)

Part A (5 marks): You are building a house price prediction model and have the following RMSE results:

Model	Training RMSE	Validation RMSE	Test RMSE
Linear Regression	68,500	69,000	67,800
Decision Tree	0	71,400	70,200
Random Forest	22,000	50,200	49,800

Analyze these results and answer:

1. Which model shows signs of overfitting? Explain your reasoning.
2. Which model would you select for production? Justify your choice.
3. What does a training RMSE of 0 indicate, and why is this problematic?

Part B (3 marks): Implement k-fold cross-validation from scratch. Your function should split the data into k folds and return the average performance score.

```python
```

```
def k_fold_cross_validation(X, y, model, k=5, scoring_func=None):
    # Your implementation here
    pass
```

Part C (3 marks): Explain the difference between random sampling and stratified sampling for creating train-test splits. When would stratified sampling be particularly important? Provide a code example using scikit-learn.

Question 4: Feature Engineering and Data Exploration (9 marks)

Part A (4 marks): Given a housing dataset with the following correlation matrix excerpt:

```
median_house_value    1.000000
median_income         0.687160
rooms_per_household   0.146285
total_rooms           0.135097
housing_median_age    0.114110
households           -0.077747
population_per_household -0.021985
bedrooms_per_room     -0.259984
```

1. Identify the three most important features for predicting house values.
2. Explain why `rooms_per_household` has higher correlation than `total_rooms`.
3. What does the negative correlation of `bedrooms_per_room` suggest about housing preferences?

Part B (3 marks): Create three new features from existing housing data attributes. Write the code to generate these features and explain the intuition behind each:

```python
# Example: rooms_per_household = total_rooms / households
# Your feature engineering code here
```

Part C (2 marks): Explain why correlation analysis might miss important relationships between features and target variables. What additional analysis techniques would you recommend?

Question 5: Hyperparameter Tuning and Model Optimization (8 marks)

Part A (4 marks): You are tuning a Random Forest model using GridSearchCV with the following parameter grid:

```python
```

```
param_grid = [
    {'n_estimators': [10, 50, 100],
     'max_features': [2, 4, 6]},
    {'bootstrap': [False],
     'n_estimators': [10, 50],
     'max_features': [2, 4]}
]
```

1. How many total model combinations will be evaluated?
2. If using 5-fold cross-validation, how many total model trainings will occur?
3. Explain what each hyperparameter controls in a Random Forest model.

Part B (4 marks): Implement a simplified grid search function that evaluates all parameter combinations and returns the best parameters:

```python
def simple_grid_search(model_class, param_grid, X_train, y_train, X_val, y_val, scoring_func):
    # Your implementation here
    pass
```

Question 6: Production Deployment and System Monitoring (5 marks)

Part A (2 marks): List and explain four key metrics you would monitor when deploying a machine learning model in production. Why is each metric important?

Part B (2 marks): Compare offline training vs. online training approaches for model updates. Provide scenarios where each approach would be preferred.

Part C (1 mark): Explain the concept of "data drift" and why it's important to monitor in production ML systems.

Additional Practice Questions

Question 7: Advanced kNN Implementation (8 marks)

Part A (4 marks): Modify the basic kNN algorithm to implement weighted kNN, where closer neighbors have more influence on the prediction. Show the mathematical formula and implement the weighted voting mechanism.

Part B (4 marks): Discuss how you would optimize kNN for big data scenarios. Explain at least two data structures or algorithms that could improve search efficiency.

Question 8: Pipeline Integration and Error Handling (7 marks)

Part A (3 marks): Design a robust data preprocessing pipeline that handles:

- Multiple data types (numerical, categorical, text)
- Different missing value patterns
- Outlier detection and treatment

Part B (4 marks): Implement error handling and validation checks for your preprocessing pipeline. Consider edge cases like empty datasets, single-class problems, or perfectly correlated features.

Question 9: Performance Metrics Deep Dive (6 marks)

Part A (3 marks): Calculate RMSE and MAE manually for the following predictions vs. actual values:

- Predicted: [100, 150, 200, 250, 300]
- Actual: [110, 140, 180, 260, 290]

Show all calculation steps and explain when you would prefer RMSE over MAE.

Part B (3 marks): Explain the mathematical relationship between different error metrics (RMSE, MAE, MAPE) and discuss their sensitivity to outliers with examples.

Question 10: Real-world Application Scenario (10 marks)

You are tasked with building a recommendation system for an e-commerce platform using the knowledge from this course.

Part A (3 marks): Frame this as a machine learning problem. Define:

- Problem type (classification/regression/clustering)
- Input features you would use
- Target variable and how you would measure success

Part B (4 marks): Design the complete data pipeline from raw data to model prediction, including:

- Data collection strategy
- Preprocessing steps
- Feature engineering approach
- Model selection criteria

Part C (3 marks): Discuss deployment considerations:

- How would you handle real-time predictions?

- What monitoring metrics would you track?
- How would you handle model updates and retraining?

Coding Implementation Templates

Template 1: Complete ML Pipeline

python

```python
# Template for end-to-end ML pipeline implementation
class MLPipeline:
    def __init__(self):
        self.preprocessor = None
        self.model = None
        self.is_fitted = False

    def fit(self, X, y):
        # Implement training logic
        pass

    def predict(self, X):
        # Implement prediction logic
        pass

    def evaluate(self, X, y):
        # Implement evaluation logic
        pass
```

Template 2: Custom Transformer

python

```python
from sklearn.base import BaseEstimator, TransformerMixin

class CustomFeatureEngineer(BaseEstimator, TransformerMixin):
    def __init__(self, feature_combinations=None):
        self.feature_combinations = feature_combinations

    def fit(self, X, y=None):
        # Implement fitting logic
        return self

    def transform(self, X):
        # Implement transformation logic
        pass
```

Assessment Rubric Guidelines

Theoretical Questions (40% of marks):

- **Excellent (90-100%)**: Complete understanding, clear explanations, correct terminology
- **Good (75-89%)**: Good understanding with minor gaps, mostly correct explanations
- **Satisfactory (60-74%)**: Basic understanding, some correct elements but lacks depth
- **Needs Improvement (<60%)**: Limited understanding, significant errors or omissions

Practical Implementation (60% of marks):

- **Code Correctness (40%)**: Functionality, logic, handling edge cases
- **Code Quality (30%)**: Readability, efficiency, best practices
- **Documentation (30%)**: Comments, variable names, explanation of approach

Manual Calculations:

- Must show all intermediate steps
- Correct mathematical notation
- Clear reasoning for each decision
- Final answer highlighted

This question bank provides comprehensive coverage of the course material while following the assessment format guidelines, emphasizing both theoretical understanding and practical implementation skills.