Classwork 1: Network-Based Predictive Analytics

Social Networks and Churn Prediction

Intermediate Predictive Analytics

Due: One week from assignment date

Contents

1 Instructions

1.1 Learning Objectives

By completing this assignment, you will:

1. Construct social networks from relational data
2. Visualize networks with node attributes
3. Implement the Relational Neighbor Classifier (RNC)
4. Apply the Probabilistic RNC (PRNC) for churn prediction
5. Evaluate network-based predictions
6. Understand challenges in network data analysis

1.2 Submission Requirements

- **Format**: R Markdown (.Rmd) file that compiles to PDF
- **Code**: All code must be included and executable
- **Output**: Include all outputs (tables, plots, results)
- **Explanations**: Answer all questions with clear, concise text
- **Deadline**: Submit via the course management system

1.3 Grading Rubric

Component	Points	Description
Data Loading & Network Construction	15	Correct implementation
Network Visualization	15	Quality and clarity
RNC Implementation	20	Correct algorithm
PRNC Implementation	20	Proper probabilistic approach
Analysis & Interpretation	20	Thoughtful answers
Code Quality & Documentation	10	Clean, commented code
Total	**100**	

2 Part 1: Network Construction (25 points)

2.1 Exercise 1.1: Load and Explore Data (10 points)

You are provided with a dataset of customer relationships and churn information for a telecommunications company.

```r
# Load required libraries
library(igraph)
library(tidyverse)
library(knitr)

# Create sample customer data
set.seed(123)
customers <- data.frame(
  id = c(101, 102, 103, 104, 105, 106, 107, 108, 109, 110,
         111, 112, 113, 114, 115, 116, 117, 118, 119, 120),
  churn = c(0, 0, 1, 0, 1, 0, 0, 1, 0, 1,
            0, 1, 0, 0, 1, 0, 1, 0, 0, 1),
  tenure_months = c(24, 36, 12, 48, 8, 60, 18, 6, 30, 10,
                   42, 14, 54, 22, 9, 38, 11, 44, 28, 7),
  monthly_charges = c(65, 85, 45, 95, 40, 105, 55, 35, 75, 42,
                     88, 48, 98, 62, 38, 82, 44, 92, 68, 36)
)

# Create edge list (customer relationships)
edges <- data.frame(
  from = c(101, 101, 101, 102, 102, 103, 103, 104, 104, 105,
           106, 106, 107, 107, 108, 109, 110, 111, 112, 113,
           114, 115, 116, 117, 118, 119),
  to = c(102, 103, 105, 104, 106, 108, 112, 107, 109, 110,
         111, 113, 114, 115, 116, 117, 118, 119, 120, 120,
         116, 117, 118, 119, 120, 120)
)
```

Tasks:

a) Load the data and create a summary table showing:
   - Total number of customers
   - Number of churned vs. non-churned customers
   - Churn rate (percentage)
   - Mean tenure and monthly charges by churn status
b) Create the network graph using `igraph`:
   - Use `graph_from_data_frame()` to create an undirected graph
   - Print basic network statistics (number of nodes, edges, density)

Questions to answer:

1. What is the overall churn rate in this dataset?
2. How many connections (edges) exist in the network?
3. What is the network density? What does this value mean?

2.2 Exercise 1.2: Network Visualization (15 points)

Create informative network visualizations that highlight churn patterns.

```
# Your code here to:
# 1. Add customer attributes to graph nodes
# 2. Create color scheme based on churn status
# 3. Size nodes by tenure or monthly charges
# 4. Plot the network
```

Tasks:

a) Create a network visualization where:
   - Node color represents churn status (red = churned, blue = stayed)
   - Node size represents tenure (longer tenure = larger node)
   - Include clear labels and a legend

b) Create a second visualization where:
   - Node color represents churn status
   - Node size represents monthly charges
   - Use an appropriate layout algorithm (e.g., Fruchterman-Reingold)

c) Add a title and legend to both plots

Questions to answer:

1. Do churned customers appear clustered in the network? Explain.
2. Is there any visual relationship between node size (tenure/charges) and churn?
3. Which layout algorithm did you choose and why?

3 Part 2: Relational Neighbor Classifier (40 points)

3.1 Exercise 2.1: Basic RNC Implementation (20 points)

Implement the Relational Neighbor Classifier to predict customer churn.

```r
# Function to compute RNC predictions
compute_rnc <- function(graph, node_labels) {
  # Your implementation here
  # Hint: Use neighbors() function to get adjacent nodes
  # Return a vector of churn probabilities
}

# Test your function
rnc_predictions <- compute_rnc(g, customers$churn)
```

Tasks:

a) Complete the `compute_rnc()` function that:
   - Takes a graph and node labels as input
   - For each node, finds its neighbors
   - Calculates the proportion of neighbors that churned
   - Returns a vector of churn probabilities

b) Apply your function to the customer network

c) Create a data frame with:
   - Customer ID
   - Actual churn status
   - Predicted churn probability
   - Predicted churn class (using 0.5 threshold)

Questions to answer:

1. What happens to nodes with no neighbors? How did you handle this?
2. Show the predictions for the first 10 customers
3. How many correct predictions did you make using the 0.5 threshold?

3.2 Exercise 2.2: RNC Performance Evaluation (20 points)

Evaluate the performance of your RNC classifier.

```r
# Calculate performance metrics
library(caret)

# Your evaluation code here
```

Tasks:

a) Calculate and report:
    - Accuracy
    - Precision (for churn class)
    - Recall (for churn class)
    - F1-score
    - Confusion matrix
b) Create an ROC curve and calculate AUC (Area Under Curve)
c) Compare RNC performance to a baseline model that:
    - Predicts everyone will not churn (all 0s)
    - Or predicts based on overall churn rate

Questions to answer:

1. What is the accuracy of your RNC classifier?
2. Does the RNC perform better than the baseline? By how much?
3. Which is more problematic: false positives or false negatives in churn prediction? Why?
4. What does the AUC value tell you about model performance?

4 Part 3: Probabilistic RNC (30 points)

4.1 Exercise 3.1: PRNC Implementation (15 points)

Implement the Probabilistic Relational Neighbor Classifier with iterative refinement.

```r
# Function to compute PRNC with iterations
compute_prnc <- function(graph, initial_labels,
                         max_iterations = 10,
                         tolerance = 0.01) {
  # Initialize probabilities
  n <- vcount(graph)
  probs <- as.numeric(initial_labels)

  for (iter in 1:max_iterations) {
    # Your implementation here
    # Update probabilities based on neighbor probabilities
    # Check for convergence
  }

  return(probs)
}
```

Tasks:

a) Complete the `compute_prnc()` function that:
   - Initializes with known labels (as 0 or 1)
   - Iteratively updates probabilities using neighbor averages
   - Stops when convergence is reached or max iterations exceeded
   - Returns final probability estimates
b) Test your function with different numbers of iterations (1, 3, 5, 10)
c) Track and plot the change in predictions across iterations

Questions to answer:

1. How many iterations does it take to converge (changes $<$ tolerance)?
2. Show how predictions change for 3 specific customers across iterations
3. Why might PRNC perform differently than basic RNC?

4.2 Exercise 3.2: PRNC Evaluation and Comparison (15 points)

Compare PRNC performance with basic RNC.

Tasks:

a) Calculate the same performance metrics for PRNC as you did for RNC:
   - Accuracy, Precision, Recall, F1-score
   - ROC curve and AUC

b) Create a comparison table showing:
   - Baseline model performance
   - RNC performance
   - PRNC performance (with different iteration counts)

c) Visualize the comparison:
   - Create a bar chart comparing accuracy across methods
   - Create overlaid ROC curves for RNC and PRNC

Questions to answer:

1. Does PRNC outperform basic RNC? By how much?
2. How does the number of iterations affect PRNC performance?
3. For which customers does PRNC provide more confident predictions?
4. Overall, which method would you recommend for this churn prediction task? Why?

5 Part 4: Network Analysis and Challenges (30 points)

5.1 Exercise 4.1: Network Features (15 points)

Explore how network features relate to churn.

```r
# Calculate network features
customers$degree <- degree(g)
customers$betweenness <- betweenness(g)
customers$closeness <- closeness(g)
customers$clustering <- transitivity(g, type = "local")
```

Tasks:

a) Calculate the following network features for each customer:
   - Degree centrality (number of connections)
   - Betweenness centrality
   - Closeness centrality
   - Local clustering coefficient
b) Create visualizations:
   - Boxplots comparing each feature by churn status
   - Correlation heatmap of features
c) Perform statistical tests:
   - T-tests comparing each feature between churned and non-churned groups
   - Report p-values and interpret results

Questions to answer:

1. Which network features are significantly different between churners and non-churners?
2. What does a high degree centrality suggest about a customer's churn risk?
3. What does the clustering coefficient tell you about a customer's network position?
4. Could these features improve prediction beyond just using neighbors' churn status?

5.2 Exercise 4.2: Handling Challenges (15 points)

Address common challenges in network-based learning.

Tasks:

a) **Data Splitting Challenge:**
   - Implement a proper train-test split that maintains network structure
   - Use 70% for training, 30% for testing
   - Ensure test nodes are still connected in the network
   - Re-evaluate your RNC and PRNC models on this split

b) **Missing Data:**
   - Randomly remove churn labels for 20% of training customers
   - Show how many customers now have unknown neighbor labels
   - Discuss how this affects prediction quality

c) **Collective Inference:**
   - Identify customers where you have no labeled neighbors
   - Implement a strategy to handle these cases
   - Compare performance with and without this strategy

Questions to answer:

1. Why is random splitting problematic for network data?
2. How did you ensure test nodes remain connected?
3. What was the performance difference between full data and your train-test split?
4. How does missing label information propagate through the network?
5. What strategies can handle nodes with no labeled neighbors?

6 Part 5: Extensions and Reflections (25 points - Bonus)

6.1 Exercise 5.1: Advanced Analysis (15 points)

Choose ONE of the following extensions:

6.1.1 Option A: Weighted Networks

Modify your network to include edge weights representing: - Frequency of interaction - Strength of relationship - Duration of connection

Implement weighted versions of RNC and PRNC that give more importance to stronger connections.

6.1.2 Option B: Temporal Networks

Create a time-aware version where: - Only past churn information is used for prediction - Customers churn at different time points - Evaluate temporal prediction accuracy

6.1.3 Option C: Feature Integration

Combine network features with customer attributes: - Build a logistic regression model using: - Degree, betweenness, clustering (network features) - Tenure, charges (customer attributes) - RNC/PRNC predictions (relational features) - Compare this hybrid model to pure network methods

Deliverable: Implementation, results, and discussion

6.2 Exercise 5.2: Critical Reflection (10 points)

Write a 1-2 page reflection addressing:

Questions:

1. **Homophily Assumption:**
   - Does the "birds of a feather" principle hold in this data?
   - What evidence supports or contradicts homophily?
   - When might this assumption fail in real-world applications?
2. **Business Application:**
   - How would you deploy this model in a real telecom company?
   - What are the ethical considerations of using social network data?
   - Should companies intervene when they detect "churn clusters"?
3. **Limitations:**
   - What are the main limitations of network-based churn prediction?
   - How could you validate that network effects are real vs. spurious?
   - What additional data would strengthen your analysis?
4. **Future Directions:**
   - How could deep learning (Graph Neural Networks) improve prediction?
   - What role could real-time network updates play?
   - How might you handle dynamic networks where connections change?

7 Appendix: Helpful Functions and Tips

7.1 Useful igraph Functions

```r
# Network creation
g <- graph_from_data_frame(edges, directed = FALSE)

# Get neighbors of node i
neighbors(g, v = i)

# Count nodes and edges
vcount(g)  # number of vertices
ecount(g)  # number of edges

# Network density
edge_density(g)

# Add vertex attributes
V(g)$attribute_name <- values

# Get vertex attributes
V(g)$attribute_name

# Calculate centrality measures
degree(g)
betweenness(g)
closeness(g)
transitivity(g, type = "local")

# Layout algorithms
layout_nicely(g)
layout_with_fr(g)  # Fruchterman-Reingold
layout_with_kk(g)  # Kamada-Kawai

# Plotting
plot(g,
     vertex.color = colors,
     vertex.size = sizes,
     vertex.label = labels,
     edge.color = "gray50",
     main = "Title")
```

7.2 Performance Evaluation Template

```r
library(caret)
library(pROC)

# Confusion matrix
cm <- confusionMatrix(
  factor(predictions, levels = c(0, 1)),
  factor(actual, levels = c(0, 1)),
  positive = "1"
)
```

```r
print(cm)

# Extract metrics
accuracy <- cm$overall['Accuracy']
precision <- cm$byClass['Precision']
recall <- cm$byClass['Recall']
f1 <- cm$byClass['F1']

# ROC curve
roc_obj <- roc(actual, predicted_probs)
auc_value <- auc(roc_obj)

plot(roc_obj,
     main = paste("ROC Curve (AUC =", round(auc_value, 3), ")"),
     col = "blue", lwd = 2)
abline(a = 0, b = 1, lty = 2, col = "red")
```

7.3 Tips for Success

1. **Start early** - Network analysis can be computationally intensive
2. **Test incrementally** - Debug each function before moving on
3. **Visualize often** - Plots help identify issues and insights
4. **Document thoroughly** - Explain your reasoning and decisions
5. **Handle edge cases** - Consider isolated nodes, missing data, etc.
6. **Validate results** - Do your predictions make intuitive sense?

7.4 Common Pitfalls to Avoid

- **Don't** use future information when making predictions
- **Don't** forget to handle isolated nodes (no neighbors)
- **Don't** ignore the structure when splitting data
- **Don't** forget to set random seeds for reproducibility
- **Don't** skip the interpretation - numbers need context!

8 Submission Checklist

Before submitting, ensure you have:

- ☐ Completed all required exercises (Parts 1-4)
- ☐ Included all code chunks with comments
- ☐ Answered all questions in complete sentences
- ☐ Generated all requested visualizations
- ☐ Calculated all performance metrics
- ☐ Provided interpretations and insights
- ☐ Spell-checked and proofread your document
- ☐ Compiled your .Rmd to PDF successfully
- ☐ Checked that all outputs are visible in the PDF
- ☐ Named your file: `LastName_FirstName_Classwork1.Rmd`

8.1 Submission

Submit via the course portal: - **File 1**: Your .Rmd source file - **File 2**: Your compiled PDF document

Due Date: [Your instructor will specify]

Late Policy: [Your instructor will specify]

Good luck!

Questions? Contact your instructor during office hours or via email.