Challenge 2: Network Intrusion Detection System - Progress & Plan

Complete Handoff Document for Next Chat Session

Last Updated: November 27, 2024
Student: Anton Horvat
Course: AI, Machine Learning & Data - Semester 4
Project Status: Data Collection Complete, Data Understanding In Progress

---

📊 PROJECT OVERVIEW

What We're Building:

A **Machine Learning-based Intrusion Detection System (IDS)** that classifies network traffic into 5 categories:

- **Normal** - Legitimate traffic
- **DoS** - Denial of Service attacks
- **Probe** - Port scanning/reconnaissance
- **R2L** - Remote to Local unauthorized access
- **U2R** - User to Root privilege escalation

Key Focus:

Explainable AI - Security analysts must understand WHY traffic is flagged as malicious.

Dataset:

NSL-KDD from University of New Brunswick

- 125,973 training samples
- 22,544 test samples
- 41 network features + attack_type + difficulty_level
- Downloaded automatically from GitHub

✅ COMPLETED WORK

1. Documentation (100% Complete)

Full Proposal Document ✅

- Location: `/mnt/user-data/outputs/Network_IDS_Proposal_Anton_Horvat.docx`
- 16KB, complete structure matching Challenge 1
- Sections: What/Why/Who/When/How with IBM methodology
- Teacher APPROVED

ML Document - Domain Understanding ✅

- Location: `/mnt/user-data/outputs/ML_Document_Network_IDS_Anton_Horvat.docx`
- 15KB, complete ML perspective
- Sections: Problem Formulation, Target Variable, Dataset Selection, Feature Engineering, Model Considerations, Evaluation Strategy, Expected Challenges, Success Criteria
- Algorithm comparison table included

Quick Reference Cheat Sheet ✅

- Location: `/mnt/user-data/outputs/Quick_Reference_Cheat_Sheet.docx`
- For quick reference during teacher meetings
- Includes all 3 XAI principles with exact definitions

Implementation Handoff ✅

- Location: `/mnt/user-data/outputs/Challenge2_Implementation_Handoff.md`
- Complete roadmap of all sections needed

---

2. Data Provisioning Notebook - Section 1 Complete (20% Complete)

File: `Challenge2_Network_IDS_Data_Provisioning.ipynb`

- Location: `/mnt/user-data/outputs/Challenge2_Network_IDS_Data_Provisioning.ipynb`

Completed Sections:

✅ **Title & Overview**

- Professional header with project info
- Explanation of 5 attack categories

✅ 🛠️ **SETUP / Data Collection**

- **Cell 1:** All library imports
  - Core libraries: pandas, numpy, matplotlib, seaborn
  - ML algorithms: Decision Tree, Random Forest, Neural Network, XGBoost, SVM, Logistic Regression
  - Explainability: SHAP
  - Class imbalance: SMOTE
  - Availability checks for optional libraries
  - Reproducibility: random_state = 42
- **Cell 2:** Analysis 🔬 of Setup
  - Explains library choices
  - Cybersecurity context
  - Learning outcomes connections
- **Cell 3:** NSL-KDD Column Names Definition
  - All 43 columns defined with detailed comments
  - Organized into categories:
    - Basic features (9)
    - Content features (13)
    - Time-based features (9)
    - Host-based features (10)
    - Target variables (2)
- **Cell 4:** Load Training and Test Data
  - Automatic download from GitHub
  - URLs: https://raw.githubusercontent.com/defcom17/NSL_KDD/master/KDDTrain%2B.txt
  - Caches files locally (no re-download)
  - Simple, clean output
- **Cell 5:** Analysis 🔬 of Data Loading
  - Why predefined split is used

- Security context for each feature category
    - Data provenance (UNB official vs Kaggle)
    - Learning outcomes connections
    - Student voice ("I did", "I learned")
- **Cell 6:** Quick Data Quality Check
    - Shows first 5 rows
    - Dataset info
    - Attack type distribution

Key Decisions Made:

- ✅ Use GitHub download (not manual)
- ✅ Keep all 43 columns (41 features needed for prediction)
- ✅ Map specific attacks → 5 categories (done in Data Understanding)
- ✅ Use NSL-KDD only (not multiple datasets)
- ✅ Handle imbalance with SMOTE in Iteration 2 (not by adding more datasets)

---

3. Data Understanding - Visualization 1 Complete (25% Complete)

Visualization 1: Attack Type Distribution ✅

Code Created:

```python
# Maps specific attacks (neptune, satan, etc.) → 5 categories (DoS, Probe, etc.)
# Creates bar chart with:
# - 5 bars (Normal, Probe, DoS, R2L, U2R)
# - Color-coded by severity (green → dark red)
# - Count labels above bars
# - Percentage labels inside bars
# - Clean, professional styling
```

Markdown Analysis Created:

```markdown
```

```
## VISUALIZATION 1: Attack Type Distribution
**Question:** What is the distribution of different attack types?
### Why I used this visualization: [explanation]
### Conclusion: [detailed findings]
- Normal: ~67,343 (53.5%)
- DoS: ~45,927 (36.5%)
- Probe: ~11,656 (9.3%)
- R2L: ~995 (0.8%)
- U2R: ~52 (0.04%) ← SEVERE IMBALANCE!
```

Key Insights:

- Severe class imbalance identified
- U2R extremely rare (only 52 samples!)
- Justifies SMOTE + class weights strategy
- Validates realistic dataset (U2R IS rare in real networks)

🟪 WHAT NEEDS TO BE DONE

Immediate Next Steps (Data Understanding - Complete Section 2):

Add 3-4 More Visualizations:

Visualization 2: Feature Patterns (RECOMMENDED NEXT)

- Box plots showing Normal vs Attack differences
- Top 4-5 features: duration, src_bytes, dst_bytes, count, serror_rate
- Shows what the model will learn from
- Demonstrates interpretability

Visualization 3: Confusion Matrix Template

- Create reusable template
- Will use multiple times (Iteration 0, 1, 2)
- Shows which attacks get confused

Visualization 4: Feature Importance Template

- Bar chart of top 10-15 features

- Will use for Decision Tree and Random Forest
- Demonstrates explainability

Optional Visualization 5: Algorithm Comparison Template

- Bar chart comparing accuracy/F1
- Will use in Iteration 1

Statistical Summary:

- Data types check
- Missing values verification
- Numerical features summary (describe())
- Categorical features summary (value_counts)

Final Analysis 🔬 for Data Understanding:

- Summary of all findings
- What we learned about the data
- Decisions for preprocessing
- Transition to Data Preparation

Then: Data Preparation Phase (Section 3)

What Needs to Happen:

1. **Encode Categorical Features**

```python
# protocol_type: tcp/udp/icmp → 0/1/2
# service: http/ftp/smtp/... (70+ services) → numbers
# flag: SF/S0/REJ/... → numbers
```

2. **Encode Target Variable**

```python
```

```
# Normal/DoS/Probe/R2L/U2R → 0/1/2/3/4
# Use LabelEncoder
# Save encoder for later use
```

3. **Handle Missing Values**
    - Check if any exist (NSL-KDD is clean)
    - Document the check
4. **Analysis** 🔬
    - Why encoding was necessary
    - How it works
    - What the numbers mean

Then: Remaining Sections (4-12)

Section 4: Data Provisioning (Modeling Phase)

- Create X_train, X_test (feature matrices)
- Create y_train, y_test (target arrays)
- Select features to use (probably all 41)

Section 5: Sample the Data

- Decision: Use full dataset (125K samples is manageable)
- Document why

Section 6: Preprocessing

- StandardScaler for features
- Save scaler for later use

Section 7: Splitting

- Document NSL-KDD predefined split
- Explain why we use it

Section 8: Modelling - ITERATION ZERO

- Decision Tree baseline
- Training and predictions
- Confusion matrix visualization
- Feature importance visualization
- Evaluation metrics
- Extract decision rules (white-box explainability)

Section 9: ITERATION 1 - Algorithm Comparison

- Random Forest
- Neural Network
- XGBoost
- Compare all 3
- Algorithm comparison visualization

Section 10: ITERATION 2 - Optimization

- SMOTE for class imbalance
- Hyperparameter tuning (GridSearchCV)
- Final model training
- Final evaluation

Section 11: Explainability

- SHAP implementation
- SHAP visualizations
- Demonstrate XAI principles

Section 12: Save Model

- Save final model
- Save scaler
- Save encoders

📁 FILE STRUCTURE

```
Challenge 2 Files:
├── Documentation/
│   ├── Network_IDS_Proposal_Anton_Horvat.docx (✅ Complete)
│   ├── ML_Document_Network_IDS_Anton_Horvat.docx (✅ Complete)
│   ├── Quick_Reference_Cheat_Sheet.docx (✅ Complete)
│   ├── Dataset_Availability_Proof.docx (✅ Complete)
│   └── Challenge2_Implementation_Handoff.md (✅ Complete)
│
├── Notebooks/
│   ├── Challenge2_Network_IDS_Data_Provisioning.ipynb (🔄 20% Complete)
│   │   └── Section 1: Setup/Data Collection (✅ Done)
│   │   └── Section 2: Data Understanding (🔄 25% Done - 1 viz complete)
│   │   └── Sections 3-12: (⬜ To Do)
│   │
│   └── Challenge2_Data_Understanding_Visualizations.ipynb (🔄 Started)
│       └── Visualization 1: Attack Distribution (✅ Done)
│       └── Visualizations 2-4: (⬜ To Do)
│
└── Data/
    ├── KDDTrain+.txt (✅ Downloaded via notebook)
    └── KDDTest+.txt (✅ Downloaded via notebook)
```

🎯 KEY DECISIONS & RATIONALE

1. Dataset Choice: NSL-KDD Only

Decision: Use NSL-KDD (125K samples), NOT multiple datasets

Why:

- ✅ Industry-standard benchmark
- ✅ Enables comparison with 1000+ academic papers
- ✅ Sufficient samples for learning ML
- ❌ Multiple datasets = different features (41 vs 80 vs 49)
- ❌ Multiple datasets = complex feature alignment
- ❌ Multiple datasets = data engineering project, not ML project

Imbalance Solution: SMOTE + class weights (not more data)

---

2. Features: All 41 Network Features

Decision: Use all 41 features, NOT reduce to common subset

Why:

- All features are network traffic characteristics (INPUTS)
- Model needs all of them to detect attacks
- 5 attack categories are the OUTPUT (what we predict)
- Specific attack names (neptune, satan) are GROUPED into 5 categories

Confusion Clarification:

- 43 columns total = 41 features + attack_type + difficulty_level
- 41 features = INPUTS (what model learns from)
- 5 categories = OUTPUT (what model predicts)

---

3. Attack Mapping

Decision: Map specific attacks → 5 main categories

Mapping:

```python
attack_mapping = {
    'normal': 'Normal',
    'neptune', 'smurf', 'pod', etc. → 'DoS',
    'satan', 'ipsweep', 'nmap', etc. → 'Probe',
    'guess_passwd', 'ftp_write', etc. → 'R2L',
    'buffer_overflow', 'rootkit', etc. → 'U2R'
}
```

Why:

- Industry-standard taxonomy
- Actionable for security analysts
- Matches NIST/MITRE frameworks
- More realistic than predicting specific attack variants

---

4. Visualization Strategy

Decision: 4-5 total visualizations

Essential (Must Have):

1. ✅ Attack distribution (Done)
2. Confusion matrix (Iteration Zero)
3. Feature importance (Decision Tree/Random Forest)
4. Algorithm comparison (Iteration 1)

Optional (Nice to Have): 5. Feature patterns (box plots) 6. Final confusion matrix (shows improvement) 7. SHAP summary (explainability)

Why Not More:

- ❌ 41x41 correlation heatmap = unreadable
- ❌ Individual histograms for all features = time waste
- ❌ Multiple confusion matrices for every algorithm = redundant
- ✅ 4-5 viz = perfect balance of thoroughness and efficiency

---

5. Writing Style

Decision: Student voice, first person, conversational

Examples:

- ✅ "I learned that...", "I decided to...", "I'm using..."
- ✅ "From the wine assignment, I learned..."
- ✅ "My teacher warned me about Kaggle..."
- ❌ NOT: "We define...", "One should...", "The system..."

Why:

- Sounds authentic (like student documenting work)
- Shows personal learning journey
- More engaging than formal documentation

🔬 THREE XAI PRINCIPLES (Critical for This Project)

1. TRANSPARENCY

Definition: Process by which input data results in predictions is reproducible, reliably described, and decisions are motivated.

How I'm Doing It:

- Document exact data source (UNB-CIC GitHub)
- Record dataset characteristics (125K train, 22K test)
- Explain feature categories and meanings
- Use random_state for reproducibility
- Document all preprocessing steps

2. INTERPRETABILITY

Definition: Humans can comprehend project cohesion and results by making them comparable to domain knowledge and baselines.

How I'm Doing It:

- Connect features to security concepts (num_failed_logins → brute force)
- Compare model findings to known attack patterns (DoS = high packet rate)
- Use human-understandable feature names (not X1, X2, X3)
- Link predictions back to security analyst workflow

3. EXPLAINABILITY

Definition: Tools and methods that turn black-box models into grey/white-box models by showing decision-making process and feature importance.

How I'm Doing It:

- White-box: Decision Tree rules (if packet_rate > 100 → DoS)
- Grey-box: SHAP values for Random Forest/Neural Networks
- Feature importance rankings
- Show which features contributed to each prediction

📝 QUICK INSTRUCTIONS FOR NEXT CHAT

What to Say:

> "I'm working on Challenge 2 - Network Intrusion Detection. I have the progress document. We completed Setup/Data Collection and started Data Understanding (1 visualization done). I need to finish Data Understanding by adding 2-3 more visualizations, then move to Data Preparation (encoding). Here's the progress document."

What to Provide:

1. This handoff document
2. The current Data Provisioning notebook (if you want to continue it)
3. Any specific preferences for next visualizations

What Claude Will Do:

1. Read this entire document
2. Understand current progress
3. Create the next visualizations
4. Continue following the structure
5. Match your writing style

🎓 LEARNING OUTCOMES COVERAGE

How This Project Demonstrates All 5 Outcomes:

1. Professional Standard

- ✅ IBM Data Science Methodology
- ✅ Industry-standard dataset (NSL-KDD)
- ✅ Stakeholder focus (SOC analysts)
- ✅ Ethical considerations (privacy, surveillance)
- ✅ Professional documentation

2. Personal Leadership

- ✅ Learning new domain (cybersecurity)

- ✅ Career alignment (specializing in security)
- ✅ Independent research (attack taxonomy, NIST/MITRE)
- ✅ Decision-making (dataset choice, algorithm selection)

3. Explainable AI (THE KEY DIFFERENTIATOR)

- ✅ Transparency: Reproducible pipeline, documented decisions
- ✅ Interpretability: Features → security concepts, domain baselines
- ✅ Explainability: SHAP, feature importance, decision rules
- ✅ Multiple approaches: White-box (Decision Tree), Grey-box (SHAP for Neural Net)

4. Data Preparation & Analysis

- ✅ Professional data pipeline (download, validate, encode)
- ✅ Handling imbalanced classes (SMOTE)
- ✅ Feature engineering (categorical encoding)
- ✅ Domain-specific analysis (attack patterns)

5. Model Engineering

- ✅ Security-specific metrics (detection rate, false positive rate)
- ✅ Per-attack-type evaluation (not just overall accuracy)
- ✅ Algorithm comparison (Decision Tree → Random Forest → Neural Net → XGBoost)
- ✅ Hyperparameter tuning (GridSearchCV)

🚨 COMMON PITFALLS TO AVOID

1. Class Imbalance Mistake

❌ **DON'T:** Train model directly on imbalanced data ✅ **DO:** Use SMOTE in Iteration 2 + class weights

2. Evaluation Metric Mistake

❌ **DON'T:** Only report overall accuracy (misleading with imbalance) ✅ **DO:** Report per-attack F1-scores, detection rate, false positive rate

3. Feature Confusion

❌ **DON'T:** Think 5 attack types = 5 features ✅ **DO:** Understand: 41 features (inputs) → predict 5 categories

(output)

4. Dataset Confusion

❌ **DON'T:** Use random Kaggle uploads ✅ **DO:** Use official UNB source (GitHub mirror is fine)

5. Explainability Mistake

❌ **DON'T:** Just use black-box models without explanation ✅ **DO:** Start with Decision Tree (white-box), add SHAP for complex models

6. Visualization Overload

❌ **DON'T:** Create 20+ visualizations ✅ **DO:** Focus on 4-5 essential, high-quality visualizations

---

📊 EXPECTED RESULTS (Benchmarks)

Based on academic literature using NSL-KDD:

Overall Accuracy:

- **Iteration Zero (Decision Tree):** ~75-80%
- **Iteration 1 (Random Forest/XGBoost):** ~85-90%
- **Iteration 2 (Optimized):** ~90-95%

Per-Attack Performance (F1-Scores):

- **Normal:** 95-98% (easy, lots of samples)
- **DoS:** 90-95% (clear patterns)
- **Probe:** 80-90% (moderate difficulty)
- **R2L:** 60-70% (challenging, rare)
- **U2R:** 40-60% (very challenging, extremely rare)

Key Challenge:

U2R with only 52 samples is VERY hard to detect

- This is realistic - privilege escalation is rare and hard to catch
- SMOTE will help but won't solve completely
- Focus on improving recall (catch attacks) even if precision drops

---

🎯 SUCCESS CRITERIA

Technical Benchmarks:

- ✅ Overall accuracy > 75%
- ✅ DoS F1-score > 90%
- ✅ Probe F1-score > 80%
- ✅ R2L F1-score > 50%
- ✅ U2R F1-score > 40%
- ✅ False positive rate < 5%

Learning Outcome Criteria:

- ✅ Transparency: Complete pipeline documentation
- ✅ Interpretability: Features connected to security concepts
- ✅ Explainability: SHAP + feature importance + decision rules
- ✅ Professional: Industry-standard practices throughout
- ✅ Personal: Demonstrates cybersecurity domain learning

Project Completion Criteria:

- ✅ Working Iteration Zero (Decision Tree baseline)
- ✅ 3+ algorithm comparison
- ✅ Per-attack evaluation with confusion matrix
- ✅ Explainability demonstrations
- ✅ Complete documentation

📚 RESOURCES & REFERENCES

Dataset:

- **Primary:** https://www.unb.ca/cic/datasets/nsl.html
- **GitHub Mirror:** https://github.com/defcom17/NSL_KDD

Academic Papers:

- Tavallaee et al. (2009) - NSL-KDD methodology
- Buczak & Guven (2016) - IDS survey

Frameworks:

- NIST Cybersecurity Framework
- MITRE ATT&CK

Tools:

- scikit-learn documentation
- SHAP documentation
- imbalanced-learn documentation

✅ FINAL CHECKLIST

Documentation:

- [x] Full Proposal
- [x] ML Document
- [x] Quick Reference
- [x] Dataset Proof
- [x] Implementation Handoff
- [x] Progress Handoff (this document)

Notebooks:

- [x] Data Provisioning started
- [x] Setup/Data Collection complete
- [ ] Data Understanding (25% - need 3-4 more viz)
- [ ] Data Preparation
- [ ] Data Provisioning (modeling)
- [ ] Preprocessing
- [ ] Iteration Zero
- [ ] Iteration 1
- [ ] Iteration 2
- [ ] Explainability

- [ ] Save Model

Visualizations:

- [x] Attack distribution (1/4)
- [ ] Feature patterns (2/4)
- [ ] Confusion matrix (3/4)
- [ ] Feature importance (4/4)
- [ ] Optional: Algorithm comparison (5/5)

END OF HANDOFF DOCUMENT

Ready to continue in next chat! 🚀