Project ChurnBot: Full Strategic & Technical Report

Author: Phillip Harris
Tech Stack: 🗄️ SQLite, 📊 Jupyter, 🐍 Python, 🔥 PyTorch, 💻 C++, 🔧 MLOps, 💻 TypeScript, 🐳 Docker, ⚛️ React, 🌐 Node.js

---

1. Executive Summary

ChurnBot is a **domain-specific AI assistant for telecom churn**. Unlike generic models, it detects telecom-specific behaviors—call patterns, service degradation, subscription anomalies—providing actionable insights and reducing churn-related losses.

Key Differentiators:

- Specialized smaller models outperform massive general-purpose LLMs for telecom tasks.
- Entirely **local-first**: no cloud, no external data transfer.
- Dual interfaces: Terminal (light) & Dashboard (rich visualization).
- Modular architecture allows integration of IT and security monitoring pipelines later.

---

2. Problem Statement

Traditional AI approaches often miss critical telecom-specific signals:

- Call patterns & usage anomalies

- Billing disputes & payment behavior
- Service degradation indicators
- Subscription plan changes

Impact: High false positives/negatives → wasted marketing spend, preventable churn, loss of revenue.

Solution: ChurnBot's three-stage cascade detects these patterns using specialized models: **Random Forest** → **ANN** → **RNN**.

---

3. Core Thesis

Specialized Smaller Models > Generic Larger Models

Research Hypothesis: Domain-specific smaller models outperform massive general-purpose models in precision, recall, and computational efficiency for telecom churn prediction.

Key Arguments:

- 🎯 Focused architecture captures subtle patterns.
- ⚡ Lower computational overhead; high performance on local machines.
- 🔍 Signal-to-noise optimization avoids generalization dilution.
- 💡 Feature engineering + ensembles outperform brute-force parameter scaling.

---

4. Churn Model Pipeline

Three-Stage Cascade:

1. **Random Forest (RF):** Quick baseline classification.

2. **Artificial Neural Network (ANN):** Models complex relationships.

3. **Recurrent Neural Network (RNN):** Captures temporal sequences (call/data patterns).

Pipeline Architecture:

`data_loader → preprocessor → feature_engineer → leakage_monitor → cascade_model → experiment_runner`

---

5. IT & Security Pipelines

Goal: Provide modular IT/security monitoring while remaining **free and local-first**, with monetization options later.

5.1 Anomaly / Intrusion Detection

- **Dataset:** `flows.csv`
- **Model:** IsolationForest (unsupervised), RandomForest (supervised)
- **Free Approach:** Synthetic flows + local logging
- **Future Paid:** Integrate real network telemetry APIs

5.2 Authentication / Account Abuse

- **Dataset:** `auth_logs.csv`
- **Model:** XGBoost / LightGBM
- **Free Approach:** Simulated login patterns
- **Future Paid:** Real user logs via secure API with key

5.3 Ticket Classification & Routing

- **Dataset:** `tickets.csv`
- **Model:** TF-IDF + LogisticRegression (baseline), small transformer (advanced)
- **Free Approach:** Local mock ServiceNow stub
- **Future Paid:** Real ServiceNow API integration with user API key

Notes:

- Users provide API keys for monetized integrations.
- Local-first mode preserves data privacy & reduces liability.
- Modular design allows incremental upgrades for subscription features.

---

6. C++ Optimization

Goal: Maximize inference speed and memory efficiency.

Implementation:

- RF, ANN, and RNN C++ implementations in `cpp_models/`
- pybind11 bindings to Python interface
- Shared optimizations:
    - Branch & bound
    - SIMD matrix ops
    - Cache-friendly data structures

- Custom memory allocators

Benchmarking: Compare Python vs. C++ pipelines using `cpp_benchmarking_lab.ipynb`.

---

7. Privacy & Security Philosophy

- **Local execution only** (zero cloud dependencies)
- No external data transfers
- Data sovereignty → regulatory compliance
- Optional API integrations require user-provided keys
- Security/IT pipelines can run in local VM / sandbox for testing

---

8. Project Structure Highlights

```
prototype/
├── churn_pipeline/
├── chatbot_pipeline/
├── security_pipeline/
├── it_pipeline/
├── cpp_models/
├── interfaces/
├── utils/
├── notebooks/
├── BasePipeline.py
├── requirements.txt
└── README.md
```

Key Features:

- Modular architecture for easy integration of pipelines
- Dual-mode interfaces (Terminal + Dashboard)
- Benchmarking & experimentation ready

---

9. Requirements & Setup

System Requirements: Python 3.8+, Node.js 16+, 8GB RAM (16GB recommended)

Dependencies: PyTorch, scikit-learn, pandas, numpy, FastAPI, React/TypeScript

Setup Commands:

```
# Backend
cd backend
pip install -r requirements.txt
uvicorn app.main:app --reload

# Frontend
cd ../frontend
npm install
npm start

# Terminal Version
python BasePipeline.py --mode terminal

# Dashboard Version
python BasePipeline.py --mode dashboard
```

---

10. Benchmarking & Testing

- Baseline tests: RandomForest → ANN → RNN
- Metrics: Precision, Recall, F1, PR-AUC, Inference Time
- C++ benchmarking for speed & memory optimization
- Synthetic datasets used for free development

---

11. Monetization Strategy

1. Core telecom analytics free & local-first
2. Optional paid modules:
    - ServiceNow API integration
    - Geo-tracking API
    - Advanced anomaly detection on live telemetry
3. Users supply API keys; ChurnBot never stores sensitive credentials

---

12. Roadmap & Next Steps

Phase 1 (Immediate):

- Implement basic churn pipeline (RF → ANN → RNN)
- Build synthetic datasets for IT/security modules
- Configure Terminal + Dashboard UI
- Begin C++ implementations & benchmarking

Phase 2 (Intermediate):

- Implement IT/security pipelines (local-only)
- Wire mock ServiceNow + GeoIP offline integration
- Add notebook experiments & reproducible tests

Phase 3 (Future / Monetized):

- Enable real API integrations with user-supplied keys
- Expand anomaly detection to live system telemetry
- Add subscription tiers for advanced features

---

13. Key Advantages

- 🔒 Privacy-first, local execution
- 🎯 Domain-specific intelligence
- ⚡ Performance-optimized via C++
- 💼 Enterprise-ready modular architecture
- 💰 Monetization-ready with minimal liability

---

14. References / Inspiration

- Telecom churn datasets: `WA_Fn-UseC_-Telco-Customer-Churn.csv`
- IT/Security synthetic datasets: `auth_logs.csv`, `flows.csv`, `tickets.csv`

- Machine learning models: RandomForest, ANN, RNN, IsolationForest, XGBoost, LightGBM

- C++ performance optimization: pybind11 bindings, SIMD, cache-aware structures

---

Summary

Project ChurnBot is **a research-grade, enterprise-ready, local-first AI platform** for telecom churn, IT monitoring, and security analysis. It balances **privacy, performance, and future monetization**, and provides a **clear path from prototype to production**.

The **core value proposition**: *turn churn from guesswork into intelligence while keeping sensitive data safe and processing fully local*.