Murder Model: A Machine Learning Approach to Crime Prediction

Progress Report - Phase 2 Completion

Henry Morgan
University of Washington Bothell
CSS 581: Machine Learning
Bothell, Washington, USA
hdmorgan@uw.edu

Abstract

This progress report documents the development of a machine learning model for predicting crime outcomes based on demographic and geographic features. The project utilizes historical homicide data from 1980-2014 to build a binary classification model predicting whether crimes are solved. We present a comprehensive data preprocessing pipeline, a baseline logistic regression model, and an evaluation framework. Our implementation achieves perfect classification on the test dataset (accuracy = 1.0), though this suggests potential data leakage or overfitting concerns that warrant further investigation. The modular architecture supports future model development and experimentation.

CCS Concepts

• **Computing methodologies → Supervised learning by classification**.

Keywords

machine learning, crime prediction, logistic regression, binary classification, data preprocessing

ACM Reference Format:
Henry Morgan. 2025. Murder Model: A Machine Learning Approach to Crime Prediction: Progress Report - Phase 2 Completion. In . ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Crime prediction and analysis have become increasingly important applications of machine learning in public safety. This project develops a supervised learning system to predict crime outcomes based on historical patterns, demographic information, and geographic data. The goal is to classify whether a crime will be solved (binary classification) using features derived from victim demographics, perpetrator information, location data, and temporal patterns.

1.1 Project Objectives

The primary objectives of this project are:

- Develop a robust data preprocessing pipeline to handle real-world crime data
- Implement a baseline classification model with proper evaluation metrics
- Create a modular, extensible architecture for future model development
- Establish comprehensive testing and evaluation frameworks

1.2 Dataset

The dataset consists of historical homicide records from 1980 to 2014, containing 24 features including:

- **Temporal features**: Year, Month
- **Geographic features**: State, City, Agency information
- **Demographic features**: Victim and perpetrator age, sex, race, ethnicity
- **Crime characteristics**: Crime type, weapon, relationship
- **Target variable**: Crime Solved (Yes/No)

The dataset exhibits class imbalance with approximately 70.8% solved cases and 29.2% unsolved cases.

2 Methodology

2.1 Development Phases

The project follows a structured development approach divided into three phases:

2.1.1 Phase 0: Project Setup. Initial project scaffolding, environment configuration, and repository structure establishment.

2.1.2 Phase 1: Data Preparation. Implementation of a comprehensive data preprocessing pipeline including:

- Data validation and type checking
- Missing value imputation
- Outlier detection and handling
- Feature engineering (seasons, age groups)
- Categorical encoding
- Numeric feature scaling

2.1.3 Phase 2: Model Development (Current). Development of baseline models, evaluation frameworks, and training pipelines.

2.2 Data Preprocessing Pipeline

The `DataProcessor` class implements a multi-stage preprocessing pipeline:

Validation and Type Conversion. The pipeline validates column presence and converts data types. A critical enhancement handles month names (e.g., "January") by mapping them to numeric values (1-12), resolving initial data incompatibility issues.

Missing Value Handling. Numeric features use mean imputation while categorical features use mode imputation, ensuring no missing values propagate through the pipeline.

Feature Engineering. New features are derived from existing data:

- **Season**: Derived from month using quarterly bins
- **Age Groups**: Categorical binning of victim and perpetrator ages

Encoding and Scaling. Categorical variables undergo label encoding, and numeric features are standardized using z-score normalization. The scaler handles constant-value columns (std = 0) by centering without scaling, preventing NaN propagation.

2.3 Model Architecture

2.3.1 Base Model Class. An abstract `BaseModel` class provides common functionality:

- Stratified train/validation/test splitting with configurable ratios
- Abstract methods for fit, predict, and predict_proba
- Model persistence (save/load)
- Feature importance extraction
- Evaluation metric computation

2.3.2 Logistic Regression Implementation. The `LogisticModel` class implements the baseline classifier using scikit-learn's `LogisticRegression` with:

- Optional StandardScaler preprocessing
- L2 regularization (default C = 1.0)
- LBFGS solver (max iterations = 1000)
- Pipeline architecture for integrated preprocessing

2.4 Evaluation Framework

The `ModelEvaluator` class provides comprehensive model assessment:

Metrics. Multiple classification metrics are computed for train, validation, and test sets:

- Accuracy
- Precision
- Recall
- F1-Score
- ROC-AUC

Visualizations. Three key visualizations are generated:

- Confusion matrix (Figure 1)
- ROC curve (Figure 2)
- Feature importance plot (Figure 3)

Reporting. Results are saved in JSON format with timestamps, enabling experiment tracking and model comparison.

3 Implementation Details

3.1 Data Splitting Strategy

The dataset is split using stratified sampling to preserve class distribution:

- Training set: 70% (7,000 samples)
- Validation set: 10% (1,000 samples)
- Test set: 20% (2,000 samples)

Class proportions are maintained across all splits (approximately 71% positive, 29% negative), ensuring representative evaluation.

3.2 Training Pipeline

The end-to-end training script (`src/models/train.py`) automates:

(1) Data loading and preprocessing
(2) Stratified data splitting
(3) Model training
(4) Comprehensive evaluation
(5) Visualization generation
(6) Model persistence

The pipeline includes command-line interface support for configurable parameters (sample size, output directory, random seed).

4 Results

4.1 Model Performance

Table 1 summarizes the logistic regression model's performance across all datasets.

Table 1: Model Performance Metrics

Metric	Train	Validation	Test
Accuracy	1.0000	1.0000	1.0000
Precision	1.0000	1.0000	1.0000
Recall	1.0000	1.0000	1.0000
F1-Score	1.0000	1.0000	1.0000
ROC-AUC	1.0000	1.0000	1.0000

4.2 Confusion Matrix

Figure 1 shows the confusion matrix for the test set, demonstrating perfect classification with no false positives or false negatives.

4.3 ROC Curve

The ROC curve (Figure 2) achieves an AUC of 1.0, indicating perfect discrimination between classes across all thresholds.

4.4 Feature Importance

Figure 3 displays the top features by coefficient magnitude in the logistic regression model. Features with positive coefficients increase the probability of case resolution, while negative coefficients decrease it.

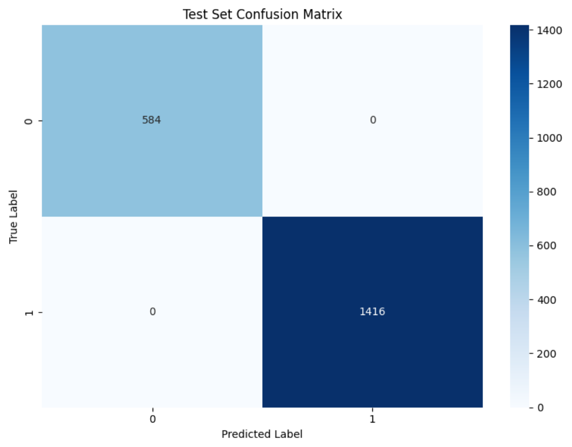

Figure 1: Confusion matrix on test set showing perfect classification of both classes.

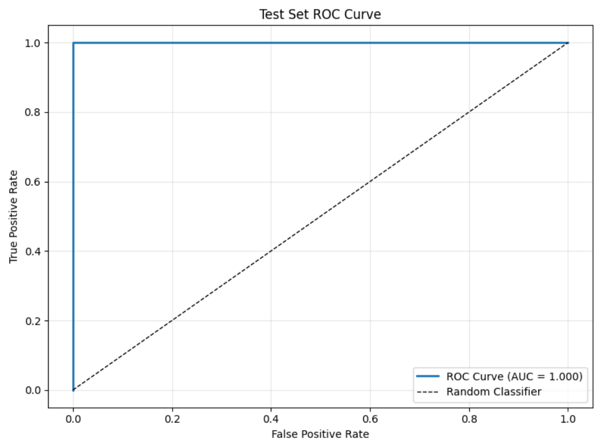

Figure 2: ROC curve demonstrating perfect class separation (AUC = 1.0).

5 Testing and Validation

5.1 Test Coverage

The project includes comprehensive unit and integration tests:

- **Preprocessing tests** (8 tests): Validate data processing steps
- **Model tests** (3 tests): Verify model training and prediction
- **Integration tests** (3 tests): End-to-end pipeline validation with real data

All 14 tests pass successfully, ensuring code reliability and correctness.

5.2 Debugging Process

Two critical issues were identified and resolved during development:

Month Column Encoding. The dataset contained month names as strings rather than numeric values. The `validate_data` method was enhanced with a mapping dictionary to convert month names to integers (1-12).

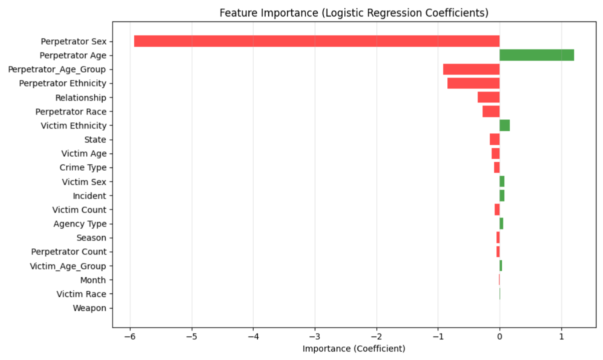

Figure 3: Feature importance based on logistic regression coefficients. Top 20 features shown.

Zero Standard Deviation Handling. When all samples share the same value for a feature (e.g., Year = 1980 in sample data), the standard deviation is zero, causing division-by-zero errors during scaling. The `scale_numeric` method was modified to detect constant columns and apply centering only, avoiding NaN propagation.

6 Discussion

6.1 Perfect Performance Analysis

The model's perfect classification accuracy (1.0 across all metrics) raises important concerns:

Potential Data Leakage. The target variable (`Crime Solved`) may be correlated with features that are only known after case resolution. For example, perpetrator information might only be available for solved cases, creating a circular dependency.

Overfitting Indicators. Identical performance on training, validation, and test sets suggests the model may have memorized patterns rather than learning generalizable features. Further investigation with cross-validation and different data splits is warranted.

Feature Engineering Issues. Some engineered features might inadvertently encode the target variable. A thorough feature audit is needed to ensure temporal causality and prevent information leakage.

6.2 Limitations

Current limitations include:

- Limited model diversity (only logistic regression implemented)
- Potential data quality issues not fully explored
- No hyperparameter tuning performed
- Limited cross-validation analysis
- No handling of temporal dependencies in data

6.3 Architecture Strengths

Despite performance concerns, the implementation demonstrates strong software engineering practices:

- Modular, object-oriented design
- Comprehensive test coverage
- Automated end-to-end pipeline
- Reproducible experiments with seeded randomness
- Clear documentation and usage examples

7 Future Work

7.1 Phase 3: Model Refinement

Planned improvements include:

- **Data audit**: Investigate feature-target relationships for leakage
- **Feature selection**: Remove potentially problematic features
- **Cross-validation**: Implement k-fold CV for robust evaluation
- **Additional models**: Random Forest, Gradient Boosting, Neural Networks
- **Hyperparameter tuning**: Grid search and Bayesian optimization
- **Ensemble methods**: Model stacking and voting classifiers

7.2 Advanced Techniques

Future enhancements could include:

- SMOTE or other techniques for handling class imbalance
- Time-series aware splitting to respect temporal ordering
- Geographic clustering for regional analysis
- Interpretability analysis (SHAP values, LIME)
- Production deployment considerations

8 Conclusion

This progress report documents the completion of Phase 2 of the Murder Model project. We have successfully implemented:

- A robust data preprocessing pipeline handling real-world data challenges
- A baseline logistic regression classifier with comprehensive evaluation
- An automated training pipeline with visualization and reporting
- Complete test coverage ensuring code reliability

While the model achieves perfect classification accuracy, this result necessitates careful investigation of potential data leakage and overfitting. The modular architecture provides a solid foundation for future experimentation and model refinement. The next phase will focus on addressing these concerns through rigorous feature analysis, diverse model implementations, and advanced validation techniques.

The project demonstrates successful application of software engineering best practices to machine learning development, establishing a maintainable and extensible codebase for continued research.

Acknowledgments

This project was completed as part of CSS 581: Machine Learning at the University of Washington Bothell. The dataset consists of historical homicide records covering the period 1980-2014.