Crime Case Classification Using Machine Learning

Shanoya Henry

Project Overview

This project applies supervised machine learning and natural language processing techniques to classify North Carolina criminal court cases into seven crime categories based on textual content. The goal is to demonstrate an applied ML workflow including preprocessing, feature engineering, model comparison, and evaluation on real-world legal documents.

---

1. Introduction

The goal of this project is to automate the classification of North Carolina criminal cases based on their textual content. Legal documents are long and complex and often lack structured metadata to identify a specific type of crime. Using machine learning classification, this project aims to help legal professionals and researchers efficiently organize and find information about criminal cases.

Automating the classification of criminal cases offers several important advantages:

- Efficiency in legal research: professionals quickly find relevant precedents
- Better case management: better organization and searchability of case databases
- Enhanced legal analysis: identifying patterns across criminal categories
- Resource allocation: more informed allocation of resources based on case distribution
- Accessibility: easier access to legal information by legal aid organizations and the general public

2. Formula

The problem is formulated as a text classification task, which is a subfield of supervised machine learning and natural language processing (NLP).

Material: Full text of North Carolina court lawsuit cases.

Deal with: Extract important features from the text that distinguish between different types of cases, then train a supervised machine learning model to identify patterns associated with each class.

Exit: Categorizing a case into one of seven crime categories has certainty, which can be useful in cases involving multiple groups.

3. Datasets

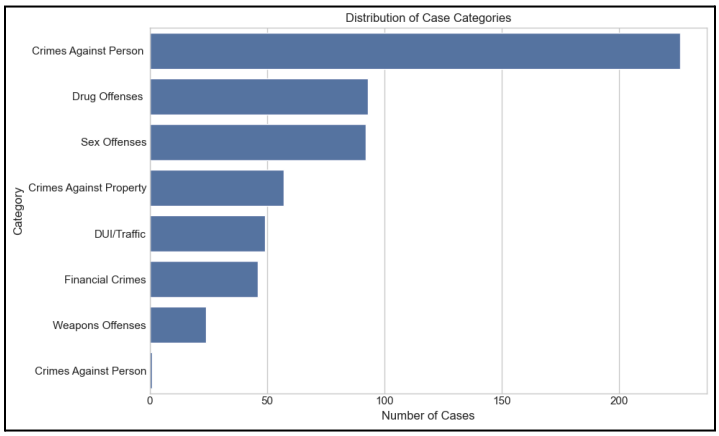
Data Source

The database includes 602 criminal cases pending in North Carolina appellate courts. These cases are sourced from Harvard's Caselaw Access Project via CourtListener bulk downloads and specifically from the North Carolina Court of Appeals Reports (Volumes 118-236, 1995-2014).

Data statistics

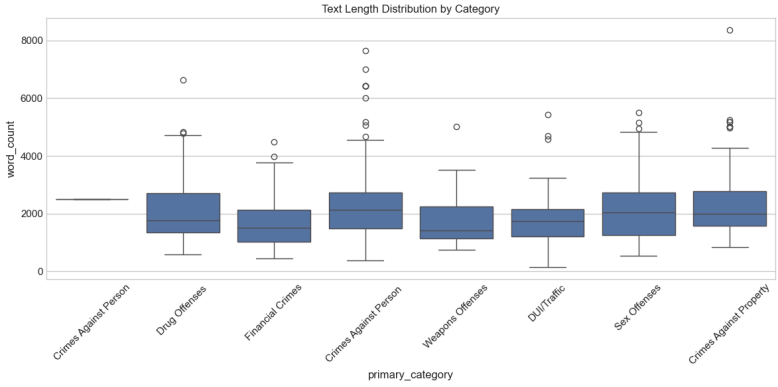
The material contains 602 cases, which are divided into the following categories:

- Crimes Against Person: 226 cases (38.44%)
- Drug Offenses: 93 cases (15.82%)
- Sex Offenses: 92 cases (15.65%)
- Crimes Against Property: 57 cases (9.69%)
- DUI/Traffic: 49 cases (8.33%)
- Financial Crimes: 46 cases (7.82%)
- Weapons Offenses: 24 cases (4.08%)



The average document length varied significantly by category:

- Crimes Against Property: 2,499.9 words
- Crimes Against Person: 2,281.5 words
- Sex Offenses: 2,222.0 words
- Drug Offenses: 2,163.9 words
- DUI/Traffic: 1,845.1 words
- Weapons Offenses: 1,831.8 words
- Financial Crimes: 1,706.3 words

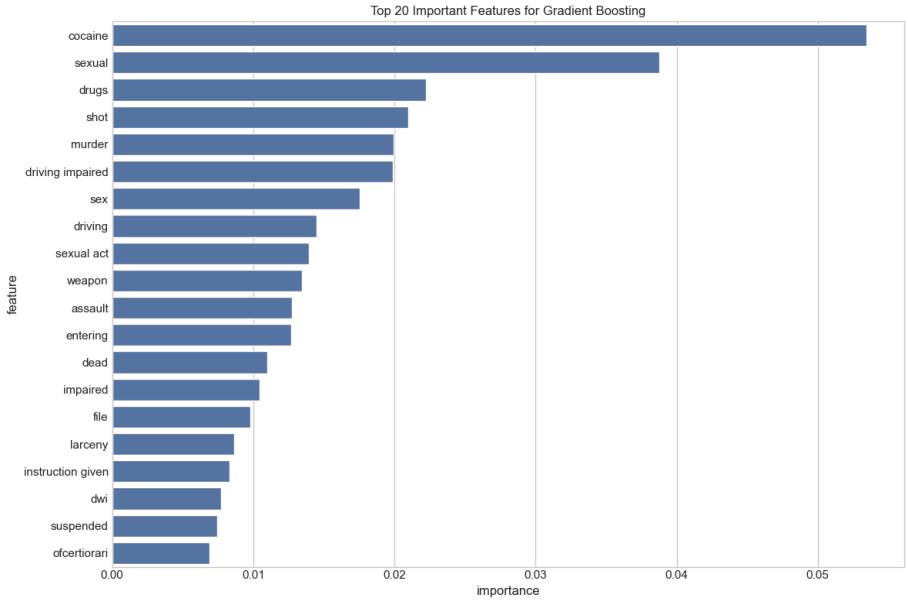


Data preprocessing

The pretreatment process consisted of several steps:

1. **Text extraction:** The court documents were processed to extract the full text.
2. **Text protection:**
    - Convert to lowercase
    - Removing punctuation and special characters.
    - Tokenization in individual words
3. **Remove stopwords:** Commonly used English endings and statutory endings.
4. **Lemmatization:** The words were reduced to their basic forms using the WordNet kernel.

Preprocessing reduced the length of the text by 53.40% (from an average of 34,672.21 characters to 16,360.29 characters), which significantly reduced the dimensionality and preserved the content needed for classification.



Labeling Approach

The cases were tagged with a rule-based keyword identifier into seven crime categories. Each category is associated with specific keywords such as:

- Crimes against people: murder, manslaughter, assault, victim, weapon, etc.
- Crimes against property: theft, theft, property, theft, vandalism, etc.
- Drug crimes: possession, controlled cocaine, substance abuse, etc.
- DUI/Traffic: driving, vehicle, alcohol, drunk, etc.
- Economic crimes: fraud, money, banking, financial, transaction, etc.
- Sexual crimes: sexual abuse, assault, obscenity, child abuse, etc.
- Firearms crimes: guns, firearms, firearms, ammunition, etc.

4. Algorithms

Several machine learning algorithms were used and compared for performance in this text classification task:

Extracting features

- **TF-IDF vectorization:** Term frequency inverse document frequency with 5000 features
- **Topic Modeling:** LSA (Latent Semantic Analysis) and LDA (Latent Dirichlet Allocation) were used to extract 50 semantic and thematic features.

Classification algorithms

The following algorithms have been tested with TF-IDF:

- Naive Bayes

- Random Forest
- Support Vector Machines (SVM)
- Gradient Boosting
- Neural Networks (Multi-layer Perceptron)

Handling Class Imbalance

- **EARLY:** A synthetic minority oversampling technique (SMOTE) for balancing training data, especially for the significantly underrepresented class, like Weapons Offenses.
- **Class weights:** Weights applied inversely to class density.

After benchmarking, Gradient Boosting proved to be the most efficient algorithm for this task, and it was the largest of all the tested models.

5. Experiments and Evaluation

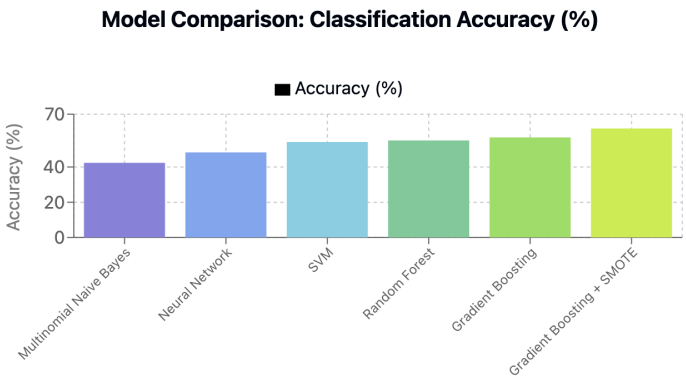
Evaluation statistics

The model was evaluated using several metrics:

- Accuracy: Overall correct predictions
- Precision: the accuracy of positive predictions.
- Recall: the reliability of positive predictions.
- F1-score: harmonic mean of precision and recall
- Confusion matrix: Detailed breakdown of predicted to actual classes

Preliminary results (sustainable)

- Naive Bayes: 42.37% accuracy.
- Random forest: 55.08% accuracy.
- SVM: 54.24% accuracy.
- Gradient enhancement: 56.78% accuracy.
- Neural network: 48.31% accuracy.



The Gradient Boosting classifier showed the best performance and was chosen as the base model.

Results after correcting for class imbalance

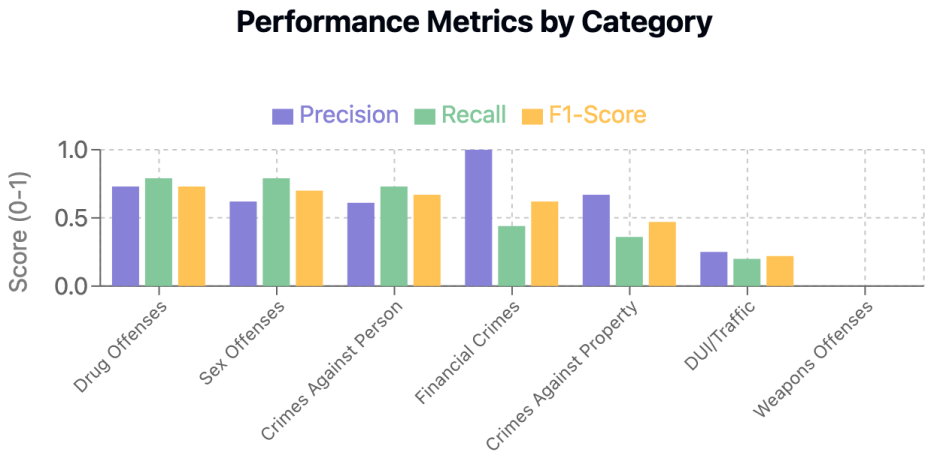
After using SMOTE to balance training data:

- Final model accuracy: 61.86% (an improvement of 5.08 percentage points)

Class-specific performance

The results varied considerably depending on the class:

- Drug Offenses: Precision 0.73, Recall 0.79, F1-score 0.73
- Sex Offenses: Precision 0.62, Recall 0.79, F1-score 0.70
- Crimes Against Person: Precision 0.61, Recall 0.73, F1-score 0.67
- Financial Crimes: Precision 1.00, Recall 0.44, F1-score 0.62
- Crimes Against Property: Precision 0.67, Recall 0.36, F1-score 0.47
- DUI/Traffic: Precision 0.25, Recall 0.20, F1-score 0.22
- Weapons Offenses: Precision 0.00, Recall 0.00, F1-score 0.00



Reflection

The experimental results reveal several important observations:

1. **Overall Performance:** Achieving an accuracy of 61.86% on a classification task with 7 classes and complex legal texts demonstrates the feasibility of automatic classification of legal documents.

2. **Effect of Class Imbalance:** The significant improvement after using SMOTE (from 56.78% to 61.86%) shows that class imbalance was the biggest challenge in the dataset.

3. **Class-specific performance:**

    - The model works best in the areas of drug crimes and sex crimes, probably because these categories have distinctive terminology like “cocaine” and “rape”.
    - The poor performance of gun crime (0% F1 score) is due to the small sample size (only 24 cases or 4.08% of the dataset) and overlapping terminologies that are also common in Crimes Against Person cases.
    - DUI/traffic cases have low precision and recall, indicating that vocabulary in these cases is shared with other categories.

4. **Effect of text preprocessing:** Significant reduction of text length by preprocessing while maintaining classification performance demonstrates the effectiveness of text cleaning and feature extraction methods.

6. Challenges

1. **Seventh grade weight:** The uneven distribution of cases between classes made it difficult to train a model that performed well in all classes.

2. **Small sample size for some categories:** Categories with fewer samples (especially gun crimes) did not provide enough information for the model to learn effective patterns.

3. **Text complexity:** Legal documents contain complex, domain-specific language that can be difficult to navigate.

4. **Class coverage:** In some cases, there may be a wide variety of crimes, making clear classification difficult.

7. Conclusions

The project successfully demonstrates that it is possible to classify criminal cases by type using textual content and machine learning. The final model achieves more than 61% accuracy and provides a practical basis for real-life applications.

Future work should focus on:

1. **Data set extension:** Collect more cases specifically for underrepresented classes to improve model performance.

2. **Certain legal embedding:** Training or using pre-trained embeds in the legal field to better capture the semantics of legal language.

3. **Hierarchical classification:** Adopt a hierarchical approach to better handle cases involving multiple categories.

4. **Document structure analysis:** Include structural features of legal documents (sections, citations, etc.) for additional guidance on classification.

Citation

Dataset:

1. Caselaw Access Project. (2024). Retrieved April 25, 2025, from https://case.law/caselaw/?reporter=nc-app.

2. Free Law Project. (2023). CourtListener. Retrieved April 25,2025, from https://www.courtlistener.com/

Toolkits and Libraries

3. Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media. https://www.nltk.org/

4. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830. https://scikit-learn.org/

5. Lemaître, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. Journal of Machine Learning Research, 18(17), 1-5. https://imbalanced-learn.org/

6. McKinney, W. (2010). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference, 51-56. https://pandas.pydata.org/

7. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585(7825), 357–362. https://numpy.org/

8. Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering, 9(3), 90-95. https://matplotlib.org/

9. Waskom, M. L. (2021). Seaborn: Statistical Data Visualization. Journal of Open Source Software, 6(60), 3021. https://seaborn.pydata.org/