Predicting Crime Risk in Toronto Using Bayesian Neural Networks: A Comparative Analysis with Decision Trees and KDE

Jun Hee Lee, Yuxin Fan, Hyun Jo Jang

Abstract

For emergency services, whether a reported crime is of high or low risk is a critical piece of information which determines response priority and scale. However, such information may be missing at the time of the initial report. We provide a comparison analysis of Decision Trees, Naive Bayes estimation, and Bayesian Neural Networks (BNN). Results show that the BNN achieves similar performance to Decision Trees and Naive Bayes KDE. Despite the lack of performance uplifts, we suggest that a BNN is suitable for our application as it provides a built-in confidence metric alongside its predictions, as well as being robust to noisy, imbalanced data.

1 Introduction

When emergency services, particularly the police, receive a crime report the initial information provided is often insufficient to create a complete picture. A detail like the type of crime and its risk factor is essential for determining appropriate response measures, yet reports may lack this detail or arrive late. To address this, we aim to uncover patterns and relationships between key features of crime reports in Toronto — the time, date, and location of the call, to assist in filling such a gap. Our objective is to develop a predictive model that uses these features to accurately determine the risk factor of the crime, whether it is high-risk or low-risk, enabling more informed and timely responses by the police. (All group members have completed the course evaluation.)

2 Literature Review

Crime prediction problem has been a popular topic in the field of machine learning [5] [3] as more and more researchers achieve good crime classifications and predictions by using various learning algorithms such as Decision Tree and Bayes related models [5] [1] [3] [4].

Mohammadi et al. [2] used kernel density estimation (KDE) and geographically weighted regression (GWR) to study the correlation between spatial factor and homicide rates across Toronto neighborhoods. They found homicide rate clustered significantly in downtown Toronto and commercial areas, suggesting that homicide tends to happen in certain spaces repeatedly.

Zhang et al. [5] compared the performance of six different learning algorithms in the context of crime hotspot prediction. Decision tree with random forest and naive bayes classifier was explored by using the same dataset. They found decision tree outperformed naive bayes model most of the time, but LSTM always performs better than both of the models. Similarly, Khan et al. reached the same conclusion in their study by using San Francisco crime data.

Furthermore, Nitta et al. [3] combined naive bayes classifier with LASSO feature selection and acquired relatively high accuracy compared with other models such as SVM and KDE. While Sharma et al. [4] focused on PCA applied on random forest classifier and improved the model accuracy, compared with simple decision tree model.

3 Problem Formulation

3.1 Dataset

We use two datasets: the Major Crime Indicators Open Data from the Toronto Police Service (2024), containing crime details like date, time, location, and type, and the Neighbourhood Profiles from Open Data - City of Toronto (2024), providing demographic and geographic information. These datasets were integrated using 158 neighbourhood codes, enabling precise mapping of crime data to characteristics such as population, average age, income, and employment rate. With neighbourhood data from 2021, we use 124,470 police reports from 2021–2023 for consistency. The 51 offense types in the police reports were categorized as low-risk or high-risk. High-risk crimes involve weapons, are labeled 'Aggravated,' or indicate potential 'Bodily Harm.' This classification helps prioritize police resources and tailor responses to crime severity.

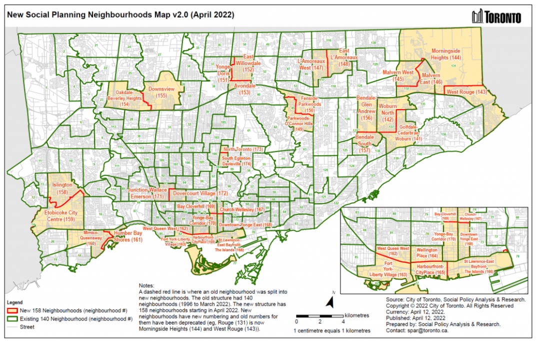

Figure 1: Mapping of 158 neighbourhoods in Toronto

3.2 Decision Tree & Random Forest

Our first two approaches were implementing decision trees and random forests as the baselines. Albeit of a simple nature, existing works on crime prediction [1] have demonstrated good results. Therefore, such models serve as our control. Performance evaluations for models introduced later in this section will be made against this baseline.

Both a single decision tree and a random forest are implemented, with hyperparameters tuned through grid-searching. The decision tree provides an easy-to-interpret model for manual inspection, while the random forest attempts to achieve the best performance. Details on their feature splits and performance will be discussed in [**Results**].

3.3 Naive Bayes Classifier

Naïve Bayes is a probabilistic classifier based on the Bayes' Rule that classifies data points by conditional probability, assuming that every feature is conditionally independent given the class label.

In the context of crime-type prediction, although the assumption of Naive Bayes hardly held in real-world cases, it provides a cheap and simple approach to infer the classification. The algorithm calculates the probability of a sample belonging to the binary classes and then selects the class with the highest probability as the final decision. By Bayes' Rule, For $k \in \{0, 1\}$ where C_0 indicates low-risk and C_1 is high-risk crimes classes,

$$P(C_k \mid X) = \frac{P(C_k)P(X \mid C_k)}{P(X)} \tag{1}$$

$$\propto P(C_k)P(X \mid C_k), \tag{2}$$

where

- $P(C_k \mid X)$ is the posterior probability of class C_k given the features X,
- $P(C_k)$ is the prior probability of class C_k,
- $P(X \mid C_k)$ is the likelihood, i.e., the probability of observing the feature set X given the class C_k, and
- $P(X)$ is the observed evidence.

We compute the prior $P(C_k)$ as the proportion of data points belonging to class C_k using the training set. Calculating the likelihood $P(X \mid C_k)$ directly from the observations can be complex and biased because of making assumptions, especially for high dimensional data. To address this, we approximate the likelihood by using Kernel Density Estimation (KDE) to address this challenge.

Kernel Density Estimation

We use Kernel Density Estimation (KDE) to model the likelihood $P(X \mid C_k)$. Compared to traditional parametric assumptions such as Gaussian distribution, KDE offers greater flexibility in estimating the probability density as a non-parametric method. This flexibility is particularly useful in real-world scenarios where the distribution of features can be highly varied and difficult to model with standard assumptions.

Finding the optimized bandwidth parameter h is critical as it determines the smoothness of the resulting density estimate. A smaller h leads to a more detailed (but potentially overfitted) estimate, while a larger h results in a smoother, more generalized curve.

In this project, we use the Gaussian kernel, which is the most widely adopted kernel function due to its smooth, bell-shaped curve. For a Gaussian kernel where $\mathcal{K}(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2}$, leading to the following KDE formula:

$$\hat{f}(x) = \frac{1}{nh\sqrt{2\pi}} \sum_{i=1}^{n} e^{-\frac{1}{2}\left(\frac{x - x_i}{h}\right)^2},$$

We fit KDE models for each feature X_j to estimate the likelihood $P(X_j \mid C_k)$ for each class C_k. Once the KDE models are trained for each feature, we compute the likelihood of observing the feature vector X given class C_k, $\hat{f}(X|C_k)$.

To classify a new data point, use Bayes' Rule to compute the posterior probability for each class. The posterior probability is given by:

$$P(C_k \mid X) \propto P(C_k)\hat{f}(X|C_k).$$

The class with the highest posterior probability is selected as the predicted class label for the new data point.

3.4 Bayesian Neural Networks

Motivation

Traditional Neural Networks are suited to finding patterns in the given data. Yet, they are vulnerable to overfitting when there is noise or anomalies in the training data. Furthermore, we would ideally like to model the uncertainty in our predictions, which is rarely done by models from our [**Literature Review**] (e.g. Decision Trees). Therefore, we experimented with a Bayesian Neural Network (BNN). We deemed it appropriate in our context due to its two useful traits: **(1) its robustness to noisy inputs, and (2) its ability to predict a probability distribution instead of a single value**.

Both advantages stem from the fact that BNN treats model parameters (weights) as probability distributions rather than fixed values. Such distributions also take in a prior, which prevents parameters from deviating too far from the assumed prior distribution. Therefore, anomalies, noise, and imbalanced data of the dataset are handled by the BNN even without extensive data preprocessing beyond normalisation and categorical encoding. Furthermore, the output of a BNN is a distribution, which can be interpreted to determine both the predicted class and confidence in that prediction. This is particularly valuable in emergency response scenarios, where understanding the reliability of a prediction can influence immediate response measures.

Moreover, BNNs can incorporate known patterns in our inputs as priors on weights. However, given the limitations in our data collection process and the need to avoid over-reliance on potentially biased priors, we opted to use non-informative priors.

Bayes' Rule

BNNs involve calculating the posterior distribution of weights $p(w|D)$ using Bayes' Rule:

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

Where D is the training data, w is the weights, $p(D|w)$ is the likelihood, $p(D)$ is the overall probability of the data, and $p(w)$ is the prior distribution (e.g. $\mathcal{N}(0, \sigma^2)$).

Training

The model produces a predictive distribution around unseen data $\hat{x}$ with unknown label $\hat{y}$ by $p(\hat{y}|\hat{x}) = \mathbb{E}_{p(w|D)}[p(\hat{y}|\hat{x}, w)]$. This expectation can be interpreted as combining the many predictions made from each layered network, using the posterior distribution. However, computing this directly is computationally infeasible for large datasets and complex models.

To address this we use Variational Inference (VI), where we approximate the posterior distribution $p(w|D)$ with a variational distribution $q(w)$. The divergence between $q(w)$ and he true posterior $p(w|D)$ is measured using the Kullback-Leibler (KL) divergence, expressed as:

$$\begin{aligned}
&D_{KL}[q(w) \,||\, p(w \mid D)] && (3)\\
&= \int_w q(w) \log\left(\frac{q(w)}{p(w \mid D)}\right) && (4)\\
&= \mathbb{E}_{q(w)}[\log(q(w)) - \log(p(w \mid D))] && (5)\\
&= \mathbb{E}_{q(w)}[\log(q(w)) + \log(p(w, D))] - \log(p(D)) && (6)
\end{aligned}$$

Then we minimize the KL divergence by learning the variational parameters θ which define $q(w)$:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} D_{KL}[q(w)||p(w|D)]$$

In practice, with our approximation of $p(w|D)$, we optimize the model through backpropagation. Our objective/loss function here is:

$$\mathcal{L} = \mathcal{L}_{data} + \lambda \cdot D_{KL}[q(w) \,||\, p(w \mid D)]$$

Where $\mathcal{L}_{data}$ is Binary Cross-Entropy (BCE) loss and λ is a hyperparameter that controls the trade-off between the BCE loss and the KL loss. Here, we use Monte Carlo sampling to estimate $\mathcal{L}_{data}$ by sampling from the distribution of weights $q(w)$ to compute a value of the loss over multiple weight samples.

4 Results

For all models: Decision Tree/Random Forest, Naive Bayes, and Bayesian Neural Network, we have performed an initial grid search with a validation set to identify optimal hyperparameters, then trained with 74,682 test data. Performance comparisons were made with accuracy on the test dataset with 24,894 entries.

4.1 Decision Tree & Random Forest

Hyperparameters and Performance

For the decision tree, three hyperparameters were tuned, to achieve a final accuracy of 82% on the test set:

- Maximum tree depth: tested with [2, 10, 100], best performer = 2
- Minimum samples needed to split: tested with [2, 10, 100], best performer = 2
- Criterion: tested with Criterions = [”gini”, ”entropy”, ”log_loss”], best performer = ”gini”

Splits

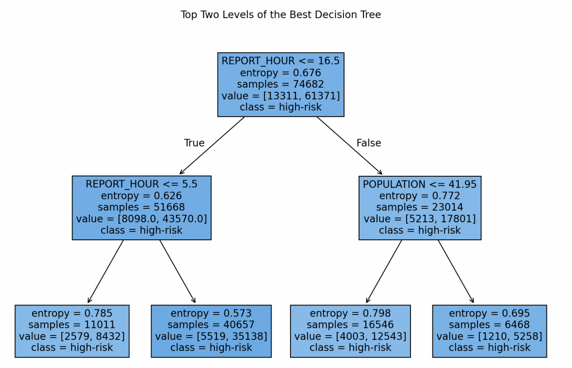

Figure 2: The best-forming decision tree with test accuracy 82%

The best accuracy of 82% was achieved through just two levels. Among the two levels, the time of the report played a significant role, with the population of the neighbourhood following next.

Random Forest

For the random forest, we performed a similar grid search operation but with [10, 20, 40] estimators. Final performance matched that of the best-performing decision tree (82% on the test set) with no additional gains.

4.2 Naive Bayes Classifier

Four Variations

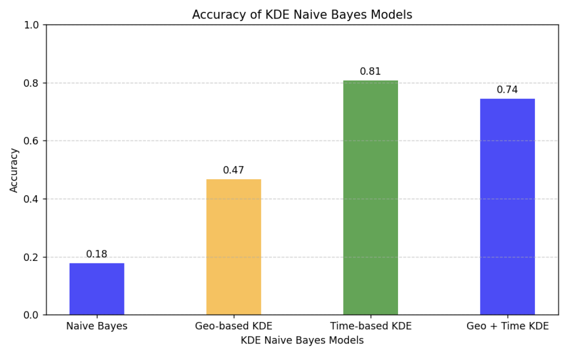

Figure 3: Four Naive Bayes models used for experimentation

Four Naive Bayes models were trained on the same dataset with KDE bandwidths optimized via grid search. The baseline model used standard Naive Bayes, while the “Geo” and “Time” models applied KDE to geographic location and time separately. The “Geo + Time” model combined both features.

Performance

The “Geo + Time” model achieved relatively high accuracy by capturing correlations between these interdependent features. In contrast, the baseline model performed the worst due to Naive Bayes’ assumption of feature independence, which oversimplified the problem. The “Time” model achieved the highest accuracy among all models, suggesting that time plays a crucial role in Toronto crime patterns, while geographic location was less significant, as indicated by the lower accuracy of the “Geo” model.

Efforts to address class imbalance through oversampling and undersampling did not yield improvements in accuracy and, in some cases, led to worse performance. A likely explanation is that balancing the training data distorted the likelihood estimates, causing a shift in the model’s predictions and reducing its ability to generalize effectively on the original imbalanced dataset.

4.3 Bayesian Neural Networks

Hyperparameters

Four hyperparameters were tuned with the validation set, with the best-performers being:

- Weight prior: Normal distribution $\mathcal{N}(\mu = 0, \sigma = 0.15)$
- Learning rate: 0.05
- KL term weight (λ): 0.01
- Optimizer: Adam

Performance

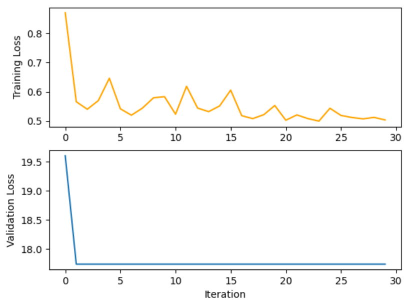

Figure 4: BNN Training and Validation Loss Curve

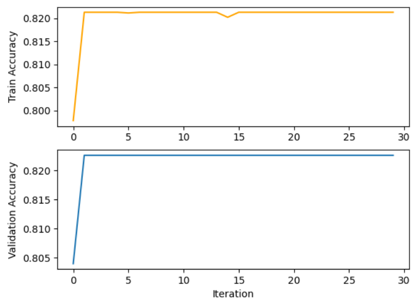

Figure 5: BNN Training and Validation Accuracy Curve

With a final test set accuracy of 82.06%, the BNN did not demonstrate a significant improvement over our decision tree baseline. However, BNN offered unique advantages, as highlighted in [**Problem Formulation**]. Most notably, the BNN provided an explicit measure of confidence in its predictions, which the decision tree and random forest lacked. This was an invaluable metric for our application, as a false-negative prediction can be life-threatening; e.g. first responders may not be adequately prepared for a high-risk crime.

Additionally, the speed and efficiency of the training process were noteworthy. Despite fluctuations in the training loss curve due to random sampling, the overall trend showed a rapid decrease in loss within the first few iterations. The validation loss and accuracy plots further corroborate the effectiveness of the BNN's learning process, demonstrating consistent performance and generalization capability.

5 Conclusion

We have experimented with three main models of different qualities: Decision trees, Naive Bayes with KDE, and finally Bayesian Neural Networks. For our task of predicting the risk factor for a given crime report from imbalanced and noisy data, the hyperparameter-tuned Decision Tree and BNN delivered the best test set accuracy of 82%. The time-based KDE results and the top-levels of decision trees being time features suggest that the time of the report may be a key factor in determining crime risk. However, the time-based KDE is less flexible than other models, as it relies solely on time data and does not integrate demographic or location information, limiting its potential for future extensions. In addition, we have identified a tradeoff relationship between the two best-performers, where BNN sacrifices some computational efficiency for native confidence values alongside its predictions, while decision trees provide fast predictions but with no indication of confidence. We suggest that the BNN is better suited for this task, as the confidence metric it provides is potentially critical for effectively responding to crime reports.

Future studies could explore a generative approach using BNNs to handle incomplete datasets by training the model on partially observed data. The posterior distributions learned by the BNN can be utilized to infer missing values, not just for the crime type but also for other critical features, such as the location coordinates of a crime. By systematically masking features during training, the BNN could develop a deeper understanding of the interdependencies among features, enabling it to make robust predictions in scenarios with missing data. This approach has the potential to enhance the generalizability of the model and offer a practical solution to real-world challenges where incomplete datasets are common. By addressing such gaps, BNNs could play a crucial role in improving public safety decision-making and optimizing resource allocation for crime prevention and intervention.

References

[1] Muzammil Khan, Azmat Ali, and Yasser Alharbi. “Predicting and preventing crime: a crime prediction model using san francisco crime data by classification techniques”. In: *Complexity* 2022.1 (2022), p. 4830411.

[2] Alireza Mohammadi et al. “Homicide rates are spatially associated with built environment and socio-economic factors: a study in the neighbourhoods of Toronto, Canada”. In: *BMC public health* 22.1 (2022), p. 1482.

[3] Gnaneswara Rao Nitta et al. “LASSO-based feature selection and naıve Bayes classifier for crime prediction and its type”. In: *Service Oriented Computing and Applications* 13 (2019), pp. 187–197.

[4] Hitesh Kumar Sharma, Tanupriya Choudhury, and Adarsh Kandwal. “Machine learning based analytical approach for geographical analysis and prediction of Boston City crime using geospatial dataset”. In: *GeoJournal* 88.Suppl 1 (2023), pp. 15–27.

[5] Xu Zhang et al. “Comparison of machine learning algorithms for predicting crime hotspots”. In: *IEEE access* 8 (2020), pp. 181302–181310.