Predicting Crime Incident Locations Using Machine Learning

An Analysis of Dallas Police Open Data (2014–2024)

By Lazaro Martull

Project Overview

This project focused on building a predictive model to classify the location of crimes using open police data from the Dallas Open Data Portal. Using the Socrata Open Data API, we retrieved and analyzed crime records from 2014 to 2024. The goal was to determine the most effective machine learning model to classify crime locations based on the type of crime and its category.

Tools and Technologies Used

Libraries and Packages

Sodapy: Python client for the Socrata Open Data API

Pandas: Data handling and preprocessing

Numpy: Numerical operations and reproducibility

Scikit-learn:

Preprocessing: ColumnTransformer, OneHotEncoder

Modeling: FeedForwardNeuralNetwork, KNeighborsClassifier, RandomForestClassifier

Evaluation: classification_report, accuracy_score, roc_auc_score, roc_curve, auc, label_binarize

Utilities: Pipeline, cross_val_score, StratifiedKFold

Matplotlib: Data visualization (bar charts, confusion matrix, ROC curves)

Data Collection and Preparation

We used the Socrata API to retrieve 10,000 crime records between 2014 and 2024 from dataset ID qv6i-rri7.

After retrieving the data, we discovered approximately 2,000 missing values, primarily in the “nibrs_crime_category” field, with most gaps occurring between 2014 and 2016. We exported the dataset to Excel and manually filled in many of the missing entries by

referencing consistent values from later years. Following this cleanup process, we increased the number of non-null rows to 9,790. To maintain data integrity, we then chose to remove 210 rows that still contained uncertain or incomplete information.

The relevant fields extracted and renamed were:

servyr → year

offincident → crime

nibrs_crime_category → crime_category

premise → location

division → division

The dataset was filtered to remove nulls and ensure the year column was converted to integer format. We decided to not utilize the division field because it did not help our model perform better.

Objective

Our objective was to build predictive models that classify the location of crime incidents based on crime type and incident description. The models evaluated were:

Feed Forward Neural Network (FNN)

K-Nearest Neighbors (KNN)

Random Forest (RF)

Independent Variables – Crime Type and Crime Description

Dependent Variable – Location (Premises)

We trained the models using data from 2014–2023 and evaluated their performance on the 2024 holdout set. We used cross-validation (3, 5, 7, and 9 folds) to assess training performance. We eventually chose 3 folds due to our model not performing much better with more folds.

We also tested our models on a filtered version of the dataset that included only the top 5 most frequent location classes and compared it to the results of the original model that used all locations. Under this setup, model accuracy improved significantly, often scoring

in the 0.50s or higher, which demonstrates the value of focusing on the most common and well-represented classes when predicting crime locations.

Binarizing Labels

To evaluate the models using AUC, we needed to binarize the target labels because our classification task involved multiple location classes. AUC is designed for binary classification, so applying it to multi-class problems requires transforming the target variable into a one-vs-rest format.

We used label_binarize from Scikit-learn to convert the categorical location labels into a binary one-hot encoded format. This step allowed us to compute an ROC curve and AUC score for each class individually and then average them to produce a meaningful overall score. Without binarizing the labels, the model would not be able to distinguish positive vs. negative classes for each ROC calculation, making the AUC metric not applicable.

This transformation was essential for interpreting the model's ability to rank instances correctly across multiple classes and was a critical part of evaluating performance beyond accuracy and F1 scores.

Using Cross Validation CV Accuracy

To evaluate how well our models generalize to unseen data, we used cross-validation (CV) — this technique splits the training data into multiple folds and rotates which fold is used for validation. In our case, we applied 3-fold cross-validation, which divides the data into 3 parts: 2 folds for training and 1 for validation, repeated three times.

This approach helped us simulate how the model would perform on different subsets of the data without touching the 2024 holdout set. It gave us a more robust estimate of model performance than a single train/test split would.

We initially tested with 3, 5, 7, and 9 folds. However, higher-fold configurations did not improve model performance meaningfully and increased runtime. As a result, we chose to use 3-fold CV for all final training evaluations.

In our dataset, which contains imbalanced and sparse class distributions especially in rare crime locations, CV accuracy helped us understand how stable and reliable each model was across different subsets of the training data. Although the CV accuracies were

relatively low, they aligned with the overall difficulty of the classification task and helped justify the use of filtered Top 5 classes to improve model reliability.

Model Evaluation and Results

Each model was evaluated using key classification metrics:

Accuracy: The proportion of total predictions the model got correct. It measures the model's overall effectiveness but may be misleading in imbalanced datasets.

Precision: The proportion of predicted positives that are truly positive. High precision indicates a low false positive rate.

Recall (Sensitivity): The proportion of actual positives that were identified correctly. High recall indicates a low false negative rate.

F1-Score: The harmonic mean of precision and recall. This balances both metrics, making it useful when there is an uneven class distribution or when both false positives and false negatives are important.

Classification Report: Provides all the above metrics (precision, recall, F1-score) for each class. It is especially useful for multi-class evaluation.

AUC (Area Under the ROC Curve): Measures the model's ability to distinguish between classes. A higher AUC indicates better model performance in ranking positive instances higher than negative ones.

Deep Learning (DL) Results

Cross-Validation Accuracy was not used for the deep learning model due to very long wait times. The computational cost is not worth it.

Holdout Accuracy (2024): 0.24

AUC (2024): 0.4983

The Deep Learning model received the highest accuracy out of all models yet the lowest AUC, meaning it was the worst at ranking classes.

K-Nearest Neighbors (KNN) Results

Cross-Validation Accuracy (3-Fold): 0.0514

Holdout Accuracy (2024): 0.1867

AUC (2024): 0.6107

KNN achieved relatively decent accuracy on the holdout data, though its AUC was lower than that of Random Forest. It performed well on rare classes like “Retail Store” but not on others.

Random Forest (RF) Results

Cross-Validation Accuracy (3-Fold): 0.0845

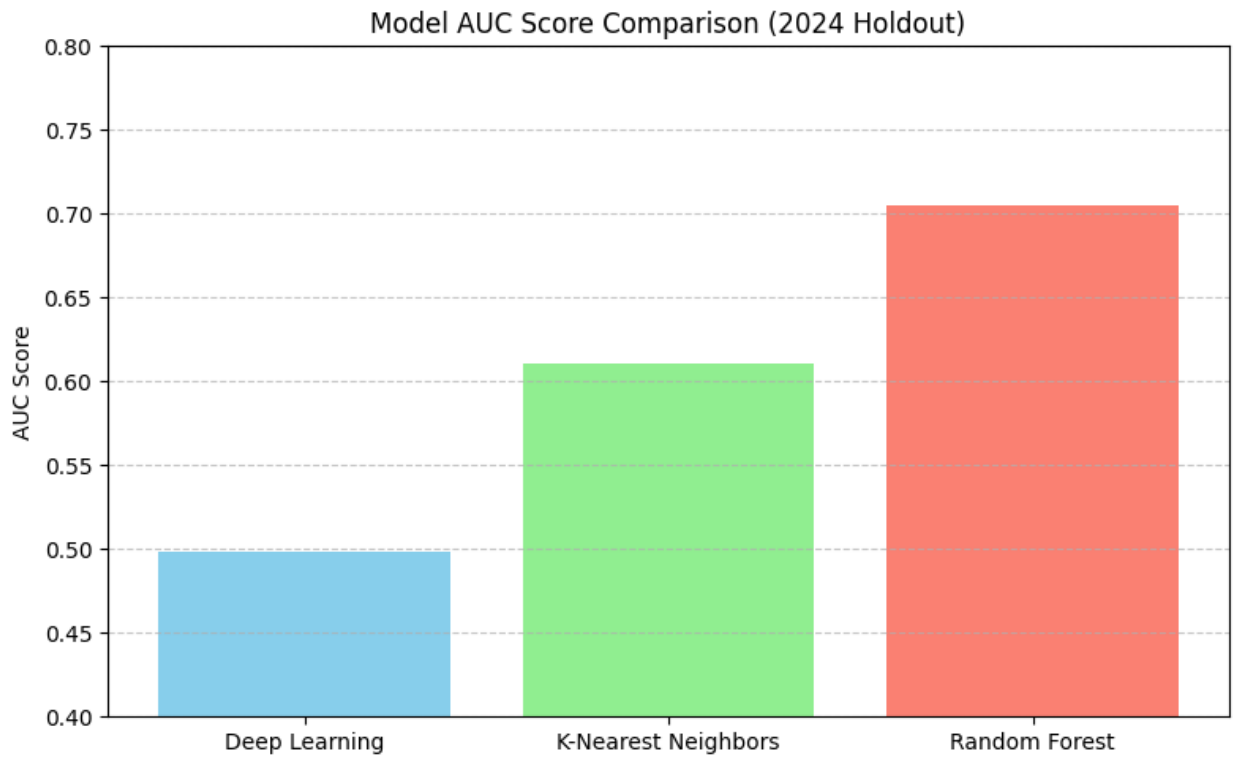
Holdout Accuracy (2024): 0.0400

AUC (2024): 0.7054

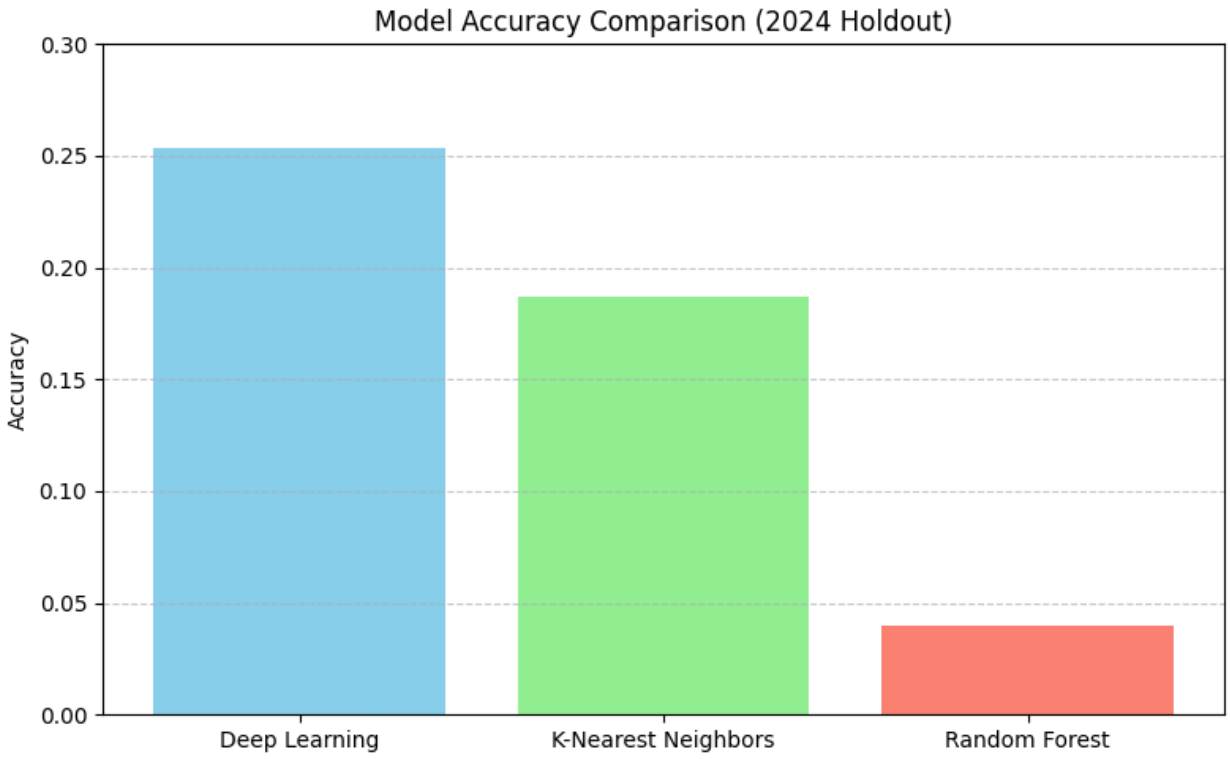
Despite the best AUC, Random Forest performed the worst in terms of accuracy. It struggled to generalize and showed poor results across most classes.

Conclusion and Recommendations:

Random Forest had the best AUC score, meaning it ranked class probabilities better than others, but it suffered in direct prediction accuracy.



Random Forest was the most accurate on the holdout set, but likely overfit to rare classes.



Deep Learning Accuracy: 0.2533

K-Nearest Neighbors Accuracy: 0.1867

Random Forest Accuracy: 0.0400

Random Forest underperformed in accuracy, suggesting its structure was not suited to this problem.

Metrics Summary: Precision, Recall, F1-Score

Weighted Average calculates the metric for each class, then takes the average, but with each class's metric weighted by its support (i.e., the number of true instances in that class).

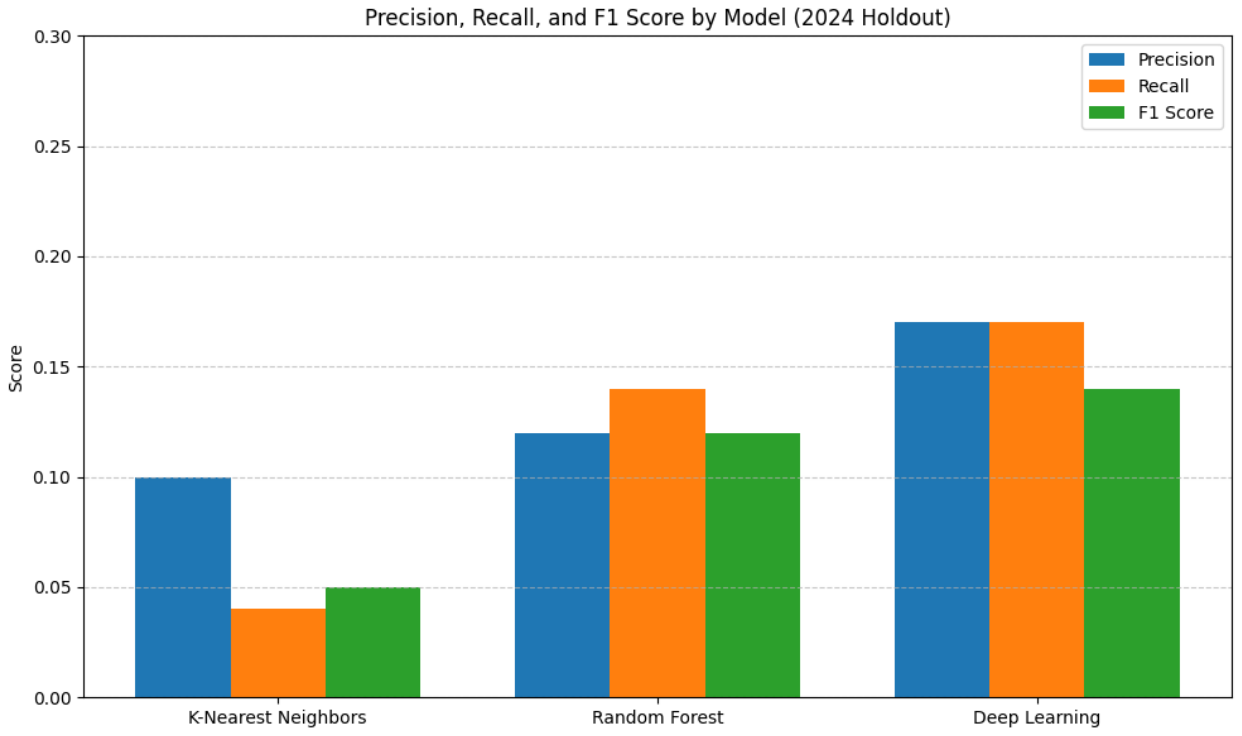
This method gives more importance to classes with more instances, which is useful when you want to consider the class distribution in your performance evaluation. We are relying more on the weighted average because of our imbalanced data set.

Weighted-Averaged Metrics (2024 Holdout Set):

Model	Precision	Recall	F1 Score
Deep Learning	0.26	0.25	0.21
K-Nearest Neighbors	0.14	0.19	0.15
Random Forest	0.19	0.04	0.07

Macro-Averaged Metrics (2024 Holdout Set):

Model	Precision	Recall	F1 Score
Deep Learning	0.17	0.17	0.14
K-Nearest Neighbors	0.12	0.14	0.12
Random Forest	0.06	0.01	0.02



Insights:

- This chart visualizes each model's ability to balance true positives and precision.
- It clearly highlights that the Deep Learning model performs slightly better than the others, especially in precision and recall.

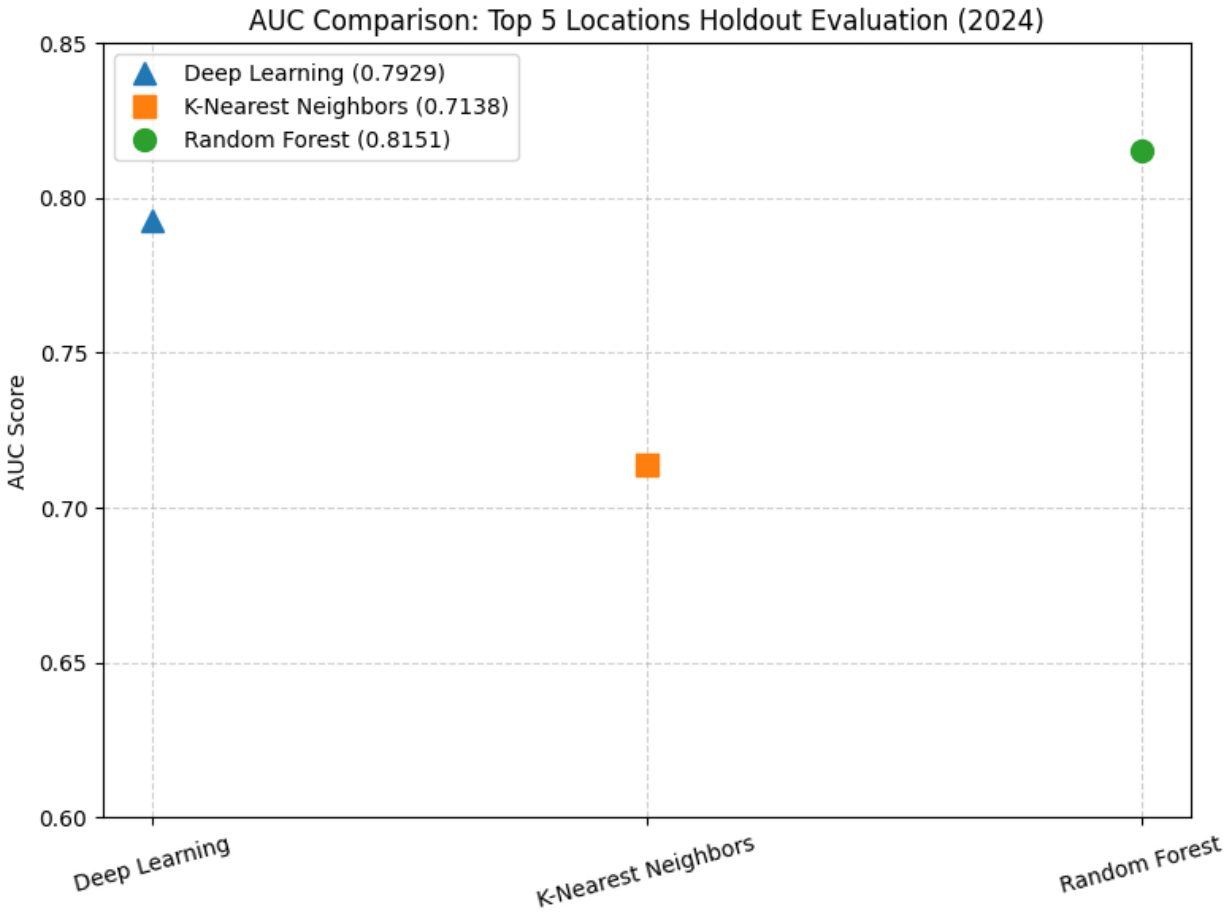
Top 5 Visualizations

Weighted-Averaged Metrics (2024 Holdout Set):

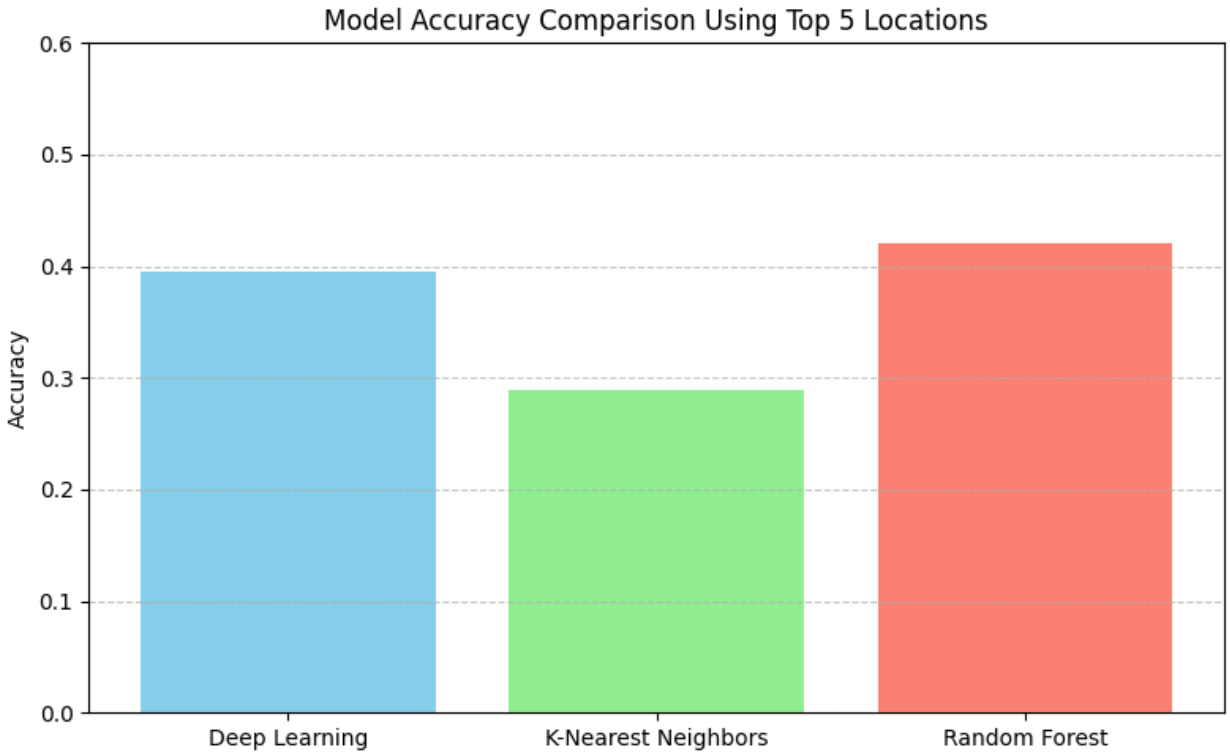
Model	Precision	Recall	F1 Score
Deep Learning	0.39	0.42	0.37
K-Nearest Neighbors	0.44	0.29	0.30
Random Forest	0.54	0.42	0.41

Macro-Averaged Metrics (2024 Holdout Set):

Model	Precision	Recall	F1 Score
Deep Learning	0.35	0.45	0.37
K-Nearest Neighbors	0.36	0.32	0.29
Random Forest	0.45	0.43	0.38



Random forest was the most accurate on the holdout set with Deep Learning Just behind it. However, we can see an increase in the AUC scores after using the top 5 locations.

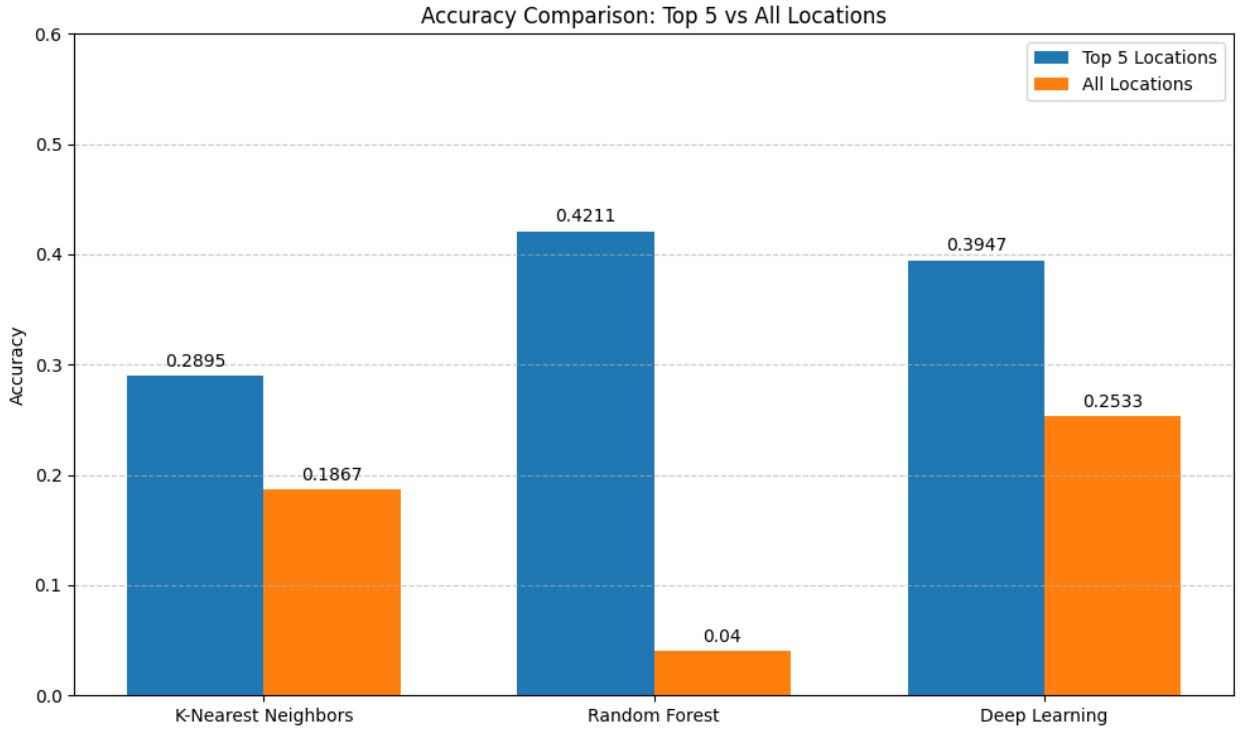


Deep Learning Accuracy: 0.3947

K-Nearest Neighbors Accuracy: 0.2895

Random Forest Accuracy: 0.4211

K-Nearest Neighbors underperformed in both the AUC and Accuracy scores. This means it was not suitable for this problem despite the modifications made to the data.



As we can see from the bar chart above. All three models perform better when trained and evaluated on the top 5 locations. Random forest achieved the highest accuracy of all models followed by Deep Learning and then K-Nearest Neighbors. The drop in model performance when using all locations could be due to class imbalance and complexity when trying to predict many location classes. However, the Deep Learning model seems to outperform both Random Forest and KNN in this challenging setting for all locations.

Conclusion

Despite significant effort, the overall classification task remains challenging due to the large number of unique output classes (i.e., many distinct crime locations). However, focusing on a smaller subset of common categories proves to be a more practical approach for deployment. Based on the results, deep learning is recommended as the best approach for future crime location prediction tasks.