Crime prediction in Hungary

Csongor Kapitány
John von Neumann Faculty of Informatics
Óbuda University
Budapest, Hungary
kcsongi@stud.uni-obuda.hu

***Abstract*—Crime is a global issue with a significant impact on people's everyday lives. In Hungary, the number of criminal offenses has shown a declining trend in recent years. However, there is currently no predictive system available to help the public understand crime trends or anticipate potential criminal activities. Addressing this gap, the aim of this article is to describe the development of a web application designed to forecast crimes at the municipal level.**

***Index Terms*—crime prediction, machine learning, web application, public safety, predictive analytics**

I. Introduction

Crime is a significant societal and economic issue. [1] It is a fundamental factor that influences decisions such as whether people choose to move to a particular city or which areas to avoid during their travels. [2]

Currently, there are several online crime maps that can accurately show crime locations and types, broken down by regions, cities, and districts in Budapest. However, despite knowing where and how crimes occur, there is no accurate prediction of future crimes. This article aims to describe in detail how to develop a web application that forecasts crime trends at the municipal level to provide clear and relevant information to the public.

II. System Overview

A web application designed for crime prediction must include fundamental features to ensure successful operation. Data import, processing, modeling, and visualization are crucial for accurate predictions. Data validation guarantees the accuracy of the information, while user interactions enable personalized usage. Comprehensive documentation and support assist users in effectively utilizing the system. Together, these elements ensure the successful implementation of the crime prediction system.

A. Technologies Used

The web application is built using the Flask framework, which provides APIs for various tasks. The backend features a machine learning model developed with scikit-learn to predict crime rates, while pandas and NumPy handle data preprocessing. Visualizations are created using Seaborn, matplotlib, and Chart.js, with map-based visualizations implemented via the OpenStreetMap API. Data is stored in an SQLite database and accessed through the DataAccess Module. The architecture integrates the frontend, backend (including APIs and the machine learning model), and external services like the Map API.

III. Related Work

The related work section explores existing crime prediction systems and the core principles of machine learning, focusing on the steps involved in ML workflow and providing a detailed explanation of each step.

A. Existing Crime Prediction Systems

- **PredPol:** Uses historical data on crime types, locations, and times to predict crimes in 150x150-meter areas [3].
- **HunchLab:** Incorporates weather, geographic, socioeconomic, and crime data for risk modeling [3].
- **IBM SPSS Modeler:** Combines demographic, historical crime, and social media data to identify hotspots and trends [4].
- **PreCobs:** Focuses on residential burglaries, generating heat maps based on past incidents [3].
- **LexisNexis Accurint Crime Analysis:** Provides real-time updates, visualization, and predictive models for law enforcement to identify crime-prone areas effectively [5].

B. Machine Learning (ML) [6]

Machine Learning, a subfield of artificial intelligence, enables systems to learn from data. Key approaches include:

- Supervised Learning: Uses labeled data for classification or prediction tasks.
- Unsupervised Learning: Identifies patterns in unlabeled data.
- Reinforcement Learning: Empowers autonomous agents to make decisions through feedback-based learning.

C. EDA

The Exploratory Data Analysis (EDA) process, essential for understanding the dataset in detail. This process involves creating attributes or features from data that help machine learning models work more effectively and potentially yield better performance. [7]

D. Model Training

Model Training: Once features are engineered, training a model is the next step. This process involves choosing from a variety of supervised machine learning methods based on the task and data characteristics. Machine learning tasks typically fall into two categories:

- Classification: Predicting a categorical class label.
- Regression: Predicting continuous values. [6]

1) Linear Regression [8]: Linear regression is a foundational technique in machine learning that models the relationship between a dependent variable and one or more independent variables. The formula for simple linear regression is:

$$y = C + \beta_x + \epsilon \tag{1}$$

For Multiple Linear Regression (MLR), multiple independent variables are considered:

$$y = C + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \ldots + \beta_n x_n + \epsilon \tag{2}$$

Advantages:

- Easy to interpret and understand.
- Computationally efficient, even with large datasets.

Disadvantages:

- Assumes linearity, which may not always be appropriate for complex data.
- Sensitive to outliers and multicollinearity.

2) Support Vector Machines (SVM) [9]: Support Vector Machines (SVM) are powerful tools for classification and regression tasks. In Support Vector Regression (SVR), SVM attempts to find the best-fitting hyperplane that minimizes the margin between data points, which can also accommodate non-linear data through kernel functions.

Advantages:

- Effective for complex, non-linear datasets.
- Less prone to overfitting compared to other methods.

Disadvantages:

- Computationally expensive, particularly for large datasets.
- Difficult to interpret, especially with non-linear kernels.
- Sensitive to noise in the data.

3) K-Nearest Neighbors (KNN) [10]: The K-Nearest Neighbors (KNN) algorithm is used for both classification and regression. For KNN regression, the predicted value is calculated as the average of the K closest training points.

Advantages:

- Simple to implement and understand.
- No model-building phase, making training time minimal.
- Flexible and adapts well to new data.

Disadvantages:

- Memory-intensive since it requires storing all training data.
- Computationally expensive at prediction time, as distances must be calculated for each query point.
- Prone to overfitting, especially with small values of K.

4) Decision Trees [11]: Decision Trees are widely used for both classification and regression tasks. In Regression Trees, data is recursively split based on feature values to minimize variance in the target variable. The tree leaves contain the predicted values.

Advantages:

- Easy to understand and interpret, which aids decision-making.
- Can handle both categorical and numerical data.
- Non-linear relationships can be effectively modeled without transformation.

Disadvantages:

- Prone to overfitting, especially when trees are deep.
- Sensitive to small variations in the data, leading to unstable models.

E. Ensemble learning

The goal of ensemble learning is to combine multiple models to improve the performance of a machine learning system, enhancing accuracy, robustness, and generalizability. The underlying idea is that multiple weak models can collectively create a strong model that performs better than any individual model.

There are two main types of ensemble learning: bagging and boosting:

Bagging (Bootstrap Aggregation) trains multiple instances of the same algorithm on different subsets of the data and then averages their predictions or makes a final decision based on voting. For example, the Random Forest algorithm trains several decision trees on different subsets of the data and combines their predictions.

Boosting is a sequential method where each new model attempts to correct the errors of the previous ones. A popular boosting algorithm, Gradient Boosting, trains each new model on the errors of the previous model, leading to progressively more accurate predictions. This method significantly improves the final ensemble model. There are various ensemble learning algorithms, such as AdaBoost, XGBoost, and LightGBM, which have proven effective in various tasks. These algorithms have become valuable tools as they combine the advantages of different models, improving prediction accuracy and model stability. [12]

1) Random Forest [13]: The Random Forest is a widely used machine learning algorithm developed by Leo Breiman and Adele Cutler. It combines the outputs of multiple decision trees to generate a final result. Its flexibility and ease of use have made it popular for both classification and regression tasks. The Random Forest algorithm consists of a collection of decision trees, where each tree is trained on a bootstrap sample of the data. One-third of the training data is set aside as a test set, known as the out-of-bag (OOB) sample. The method increases the diversity of the dataset and reduces the correlation between the trees through feature bagging, which adds randomness to the model. The prediction method depends on the task type: for regression tasks, the predictions of the individual trees are averaged, and for classification tasks, the majority vote determines the predicted class. The OOB sample is used as a form of cross-validation to refine the final prediction.

Advantages:

- Reduces overfitting risk: Decision trees can easily overfit, but the ensemble approach in Random Forest mitigates this by averaging predictions, which reduces variance and errors.
- Flexibility: Random Forest is a powerful method for both regression and classification tasks. It also handles missing values efficiently and maintains accuracy even when parts of the data are missing.
- Easy feature importance measurement: It is easy to determine the importance of individual features based on the decrease in accuracy (Mean Decrease Accuracy).

Disadvantages:

- Time-consuming: The algorithm can be slow when handling large datasets since it needs to process each tree separately.
- Resource-intensive: Handling large datasets requires substantial storage and computational resources.
- Complexity: While a decision tree's predictions are straightforward, a model built from many trees is more complex and harder to interpret.

2) XGBoost [14]: XGBoost (Extreme Gradient Boosting) is an open-source machine learning library that uses gradient boosting with decision trees. It is known for its speed, efficiency, and scalability on large datasets. Gradient boosting refers to the process of iteratively combining weak models to create a strong learning model. The gradient descent algorithm is applied to minimize errors by adjusting the model parameters based on the gradients of the error function. XGBoost optimizes this process by parallelizing tree construction, thus significantly speeding up training.

Comparison of Boosting Algorithms:

- XGBoost vs. AdaBoost: AdaBoost, developed by Yoav Freund and Robert Schapire in 1995, focuses on emphasizing difficult-to-predict data points through a weighting system. Weights are adjusted based on the correctness of prior predictions. XGBoost is faster and more accurate compared to AdaBoost, making it a more preferred choice.
- XGBoost vs. LightGBM: LightGBM, developed by Microsoft, uses a leaf-wise tree construction strategy as opposed to the depth-wise approach in most decision tree algorithms. While both XGBoost and LightGBM show similar speed and accuracy, LightGBM performs better on large datasets.

F. Model Validation

Validation ensures that a model functions as expected and can generalize to unseen data. Without proper validation, we cannot be confident that the model will perform well on new, unseen data. Validation helps determine the best model, parameters, and performance metrics for a given task, and aids in identifying potential issues before they escalate. [15]

Underfitting and Overfitting:

- Underfitting occurs when the model is too simple to capture the complexity of the data, resulting in poor performance on both training and test data. This is typically addressed by using more complex models or improved feature representation. [16]
- Overfitting occurs when a model learns not only the underlying patterns but also the noise in the training data, making it fail to generalize well on new data. This is a fundamental challenge in supervised learning and can be mitigated through methods such as cross-validation, regularization, and choosing simpler models. [17]

Performance Evaluation Metrics:

- R^2: Measures the fit of a regression model to the actual data. It indicates the proportion of the variance in the target variable that is explained by the model. [18]
- Adjusted R^2: Corrects for overfitting by adjusting the R^2 value based on the number of predictors in the model. [19]
- Mean Squared Error (MSE): Measures the average squared differences between predicted and actual values, emphasizing larger errors. [20]
- Root Mean Squared Error (RMSE): The square root of MSE, providing the error in the same units as the data, making it easier to interpret. [21]
- Mean Absolute Error (MAE): Measures the average absolute differences between predicted and actual values, less sensitive to outliers than MSE. [21]

G. Model Deployment [22]

When a machine learning model is deployed, it is integrated into a real-world system or business application. This can be achieved by making the model available through APIs (e.g., REST endpoints), enabling the application to request predictions dynamically. After deployment, the model's performance should be continuously monitored to ensure that it remains accurate and effective in real-world conditions.

IV. Methodology

The development of the web application was completed in four steps. First, the police-provided database was preprocessed using the EDA methodology with various statistical and visualization techniques. Next, the cleaned and transformed data were used to train a model using the Scikit-learn library. The model was then validated on both test and training data, and the final version was saved. Finally, the optimized model was integrated into a user-friendly Flask web application.

A. Data Collection

The database used for the prediction is based on publicly available data released by the Hungarian police and contains approximately 80,000 records of crimes committed in Hungary between 2009 and 2023.

B. Preprocessing

The data preprocessing was done using Google Colab and various Python libraries like pandas, numpy, seaborn, and matplotlib. Initially, the dataset was cleaned by removing rows with missing or invalid values, and new columns were added, including the population size and city categorization based on the Hungarian Central Statistical Office (KSH). Descriptive statistics were calculated to understand key metrics like mean, median, variance, and distribution shape. Data visualizations such as histograms, heatmaps, and boxplots helped in identifying patterns and detecting outliers. The crime data was then transformed into crime rates based on each city's population. For example in case of a smaller cities the crime rate was calculated by dividing the number of crimes by the population of the city, adjusted per 1,000 people. This approach ensured the crime rates were adjusted to the population of each city, improving the relevance of the data. To enhance the reliability of the dataset, the top and bottom 0.001% of extreme crime rate values were excluded. Label encoding and ordinal encoding were applied to convert categorical data into numerical formats suitable for machine learning. The processed dataset was then ready for use in model training.

C. Model Training

The model training process involved several key steps. First, the dataset was split into training and test data in an 80/20 ratio. Independent variables such as crime location, crime type, year, city category, and population were selected, with the dependent variable being the crime rate to predict. The model was then trained using the Scikit-learn library, which provides efficient implementations of various machine learning algorithms. [24] Additionally, the PyCaret library was used to streamline the process and compare multiple models, including Random Forest, LightGBM, XGBoost, and others. The Random Forest Regressor showed the highest accuracy, with low error rates and a high R^2 value, making it the best choice for reliable crime predictions.

D. Validation

The final Random Forest Regressor model was optimized through hyperparameter tuning using GridSearchCV and RandomizedSearchCV, ensuring the best parameters for performance. It was evaluated on test data, achieving an R^2 score of 84.21%, indicating strong predictive accuracy. Low MAE and RMSLE values confirmed stable predictions, though higher MSE and RMSE revealed occasional larger errors.

Feature importance analysis showed that crime type had the greatest influence (37%) on crime rate, followed by city category and population size. The finalized model was saved in a Pickle file for future use in the web application, ensuring precise and efficient predictions.

E. Integration

The web application was developed using Flask, a popular Python framework that facilitates the creation of dynamic and interactive web applications. [25] It uses an SQLite database that stores information about cities and crime statistics, including population, city categories, and encoded crime data.

Once the user selects a city, crime type, and year, the Flask backend processes the inputs and retrieves additional parameters like population and city type from the database. These parameters are passed to the trained machine learning model to generate predictions for crime rates for the selected year, along with projections for ±5 years surrounding it.

Predicted crime data is displayed on a prediction page, where crime rates are visualized using interactive charts created with Chart.js, offering insights into crime trends. Additionally, the geographical location of each city is displayed on a map using the OpenCage Geocoding API for location data and OpenStreetMap for rendering, allowing users to view a visual representation of the selected cities. This integration ensures a smooth, user-friendly experience for exploring and predicting crime trends across Hungarian cities.

V. Evaluation

The machine learning model demonstrated outstanding performance, with validation using 2024 real crime data confirming its reliability and effectiveness. The model was able to explain over 80% of the data on both the test set and the real 2024 data, while maintaining a low error rate, thus proving a stable and accurate performance.

The developed web application offers an intuitive and user-friendly interface, allowing users to easily input the parameters required for the model. The system automatically processes the data and presents crime predictions for the selected locations not only in text but also through interactive charts and map visualizations. This helps users analyze crime trends and geographic distribution with ease. The application provides fast response times, while the map and graphical displays support better data interpretation.

A. Challenges

During the project, several technical challenges were encountered, particularly in enhancing the accuracy of the predictive model. The initial models did not meet expectations because the input data lacked sufficient detail and contained numerous outliers, which made it difficult for the model to make accurate predictions. To address this problem, significant improvements were made to the data preprocessing pipeline. Including detailed data cleaning, removal of outliers, and optimization of the dataset, ensuring higher-quality input for the model. In addition to these improvements, the model was fine-tuned extensively. Hyperparameters were adjusted, and various algorithms were tested to achieve a better performance.

VI. Future Work

The web application could be improved by adding two key features: integrating offender-related data for deeper analysis and including infection rate levels for Hungarian cities, based on national and regional averages. These additions would further improve the model's predictions and expand the application's functionality.

- By adding information about the offenders, the application could give better insights into crime trends. This would help to closely analyze crime patterns and could make the crime predictions more accurate.
- Adding a feature that ranks cities based on their crime rates compared to national and regional averages would help users see the bigger picture of crime trends. This could help policymakers and law enforcement focus on high-risk areas and use resources more efficiently.

REFERENCES

[1] A. Bogomolov, N. O. B. Lepri, J. Staiano, F. Pianesi, and A. Pentland, "Once Upon a Crime, Towards Crime Prediction from Demographics and Mobile Data," Nov. 2014. Available at: https://www.staiano.net/pubs/ICMI2014-crimePrediction.pdf [Accessed: Mar. 2024].

[2] B. S. R. Arulanandam and M. Purvis, "Extracting Crime Information from Online Newspaper Articles," 2014. Available at: https://crpit.scem.westernsydney.edu.au/confpapers/CRPITV155Arulanandam.pdf [Accessed: Nov. 2024].

[3] A. Abraham, F. Pedregosa, M. Eickenberg, et al., "Machine learning for neuroimaging with scikit-learn," Frontiers in Neuroinformatics, vol. 8, no. 14, Feb. 2014. Available at: https://www.frontiersin.org/articles/10.3389/fninf.2014.00014/full?ref=https://githubhelp.com. [Accessed: Nov. 2024].

[4] W. Hardyns and A. Rummens, "Predictive Policing as a New Tool for Law Enforcement? Recent Developments and Challenges," European Journal on Criminal Policy and Research, vol. 24, no. 3, pp. 201–218, Sept. 2018. ISSN: 1572-9869. DOI: 10.1007/s10610-017-9361-2.

[5] W. L. Perry, B. McInnis, J. S. Carter, C. Price, and S. C. Smith, PREDICTIVE POLICING: The Role of Crime Forecasting in Law Enforcement Operations (RAND Corporation research report series). Hollywood, 2013.

[6] F. Richárd. „GÉPI TANULÁS A GYAKORLATBAN - SZEGEDI TUDOMÁNY- EGYETEM, INFORMATIKAI KARÁNAK KURZUSA". (2016), https: / / www . inf . u - szeged . hu / rfarkas / ML20 / index . html [Accessed: Apr. 2024.].

[7] A. Ghosh, M. Nashaat, J. Miller, S. Quader és C. Marston, „A comprehensive re- view of tools for exploratory analysis of tabular industrial datasets", Visual In- formatics, 2. vol., 4. sz., 235–253. old., 2018, ISSN: 2468-502X. DOI: https : //doi.org/10.1016/j.visinf.2018.12.004.

[8] T. M. Hope, "Chapter 4 - Linear Regression", Machine Learning, A. Mechelli and S. Vieira, Eds., Academic Press, 2020, pp. 67–81, ISBN: 978-0-12-815739-8. DOI: https://doi.org/10.1016/B978-0-12-815739-8.00004-3.

[9] ibm, "What Is Support Vector Machine?" (Apr. 2024), title: https://www.ibm.com/topics/support-vector-machine [Accessed: Mar. 2023].

[10] ibm, "What is the k-nearest neighbors (KNN) algorithm?" (Mar. 2023), title: https://www.ibm.com/topics/knn [Accessed: Apr. 2024].

[11] ibm, "What is a decision tree?" (Aug. 2024), title: https://www.ibm.com/topics/decision-trees [Accessed: Oct. 2024].

[12] E. I. T. H. Academy, "What is Ensemble Learning?" (Mar. 2024), title: https://hu.eitca.org/mesterséges-intelligencia/eitc-ai-gcml-google-felh

[13] ibm, "What is random forest?" (May 2023), title: https://www.ibm.com/topics/random-forest [Accessed: Apr. 2024].

[14] ibm, "What is XGBoost?" (May 2024), title: https://www.ibm.com/topics/xgboost [Accessed: Sept. 2024].

[15] R. Koch, "How to Validate Machine Learning Models: A Comprehensive Guide," (Feb. 2023), title: https://www.clickworker.com/customer-blog/how-to-validate-machine-learning-models/.

[16] ibm, "What is underfitting?" (Apr. 2024), title: https://www.ibm.com/topics/underfitting [Accessed: May 2024].

[17] ibm, "What is overfitting?" (Apr. 2024), title: https://www.ibm.com/topics/overfitting [Accessed: May 2024].

[18] ibm, "R2," (Aug. 2021), title: https://www.ibm.com/docs/hu/cognos-analytics/11.1.0?topic=terms-r2 [Accessed: May 2024].

[19] ibm, "Adjusted R-squared," (Aug. 2021), title: https://www.ibm.com/docs/hu/cognos-analytics/11.1.0?topic=terms-adjusted-r-squared [Accessed: May 2024].

[20] Encord, "Mean Square Error (MSE)," available at: https://encord.com/glossary/mean-square-error-mse/ [Accessed: May 2024].

[21] ibm, "Mean Square Error (MSE)," (Jul. 2024), title: https://www.ibm.com/think/topics/loss-function [Accessed: Sept. 2024].

[22] P. Singh, *Deploy Machine Learning Models to Production: With Flask, Streamlit, Docker, and Kubernetes on Google Cloud Platform* (Professional and Applied Computing, Apress Access Books series), 1st ed. Berkeley, CA: Apress, 2020, XIII., 150 pp., 115 b/w illustrations, ISBN: 978-1-4842-6545-1. DOI: 10.1007/978-1-4842-6546-8.

[23] A. Abraham, F. Pedregosa, M. Eickenberg, et al., "Machine learning for neuroimaging with scikit-learn," Frontiers in Neuroinformatics, vol. 8, no. 14, Feb. 2014. Available at: https://www.frontiersin.org/articles/10.3389/fninf.2014.00014/full?ref=https://githubhelp.com. [Accessed: Nov. 2024].

[24] BairesDev Engineering Team. "What is Flask?" BairesDev, Aug. 2023. Available at: https://www.bairesdev.com