# Modern Statistical data analysis 52311, 2024-25 Final Project

*William Taieb - 345788616*
*Talia Levy - 342768066*

## Question 1: Preprocessing

Before applying any modeling techniques, the dataset `diabetic_data.csv` was preprocessed to ensure data consistency and suitability . The main preprocessing steps were as follows:

1. **Loading and inspection:** The dataset was loaded and inspected to determine the number of observations and variables, data types, and the presence of missing values or placeholder symbols such as "?".

2. **Target definition:** The variable `readmitted` was transformed into a binary target `readmitted_30d`, where patients readmitted within 30 days (`"<30"`) were coded as 1, and those with `"NO"` or `">30"` were coded as 0.

3. **Handling missing values:** All placeholder values "?" were replaced with `NaN`. For numerical variables, missing values were imputed using the median. For categorical variables, missing values were imputed with the most frequent category or labeled as `"Unknown"` where appropriate.

4. **Removing identifiers:** Non-informative identifiers such as `encounter_id` and `patient_nbr` were removed to prevent data leakage.

5. **Mapping categorical codes:** Where available, the `IDS_mapping.csv` file was used to replace coded values with their corresponding descriptive labels. If the mapping file was not applicable to a variable, the original values were retained.

6. **Encoding categorical features:** All categorical variables were transformed into numeric format using one-hot encoding, dropping one category per feature to avoid multicollinearity.

7. **Scaling numerical features:** All numerical variables were standardized using z-score normalization:

   $$x_{scaled} = \frac{x - \mu}{\sigma}$$

   where $\mu$ and $\sigma$ are the mean and standard deviation of the feature, computed from the entire dataset in this step (model training will use statistics from the training set only).

8. **Output:** The cleaned dataset (before encoding/scaling), the fully transformed feature matrix, the binary target vector, and the fitted preprocessing pipeline were saved for reuse in the subsequent steps of the project.

This preprocessing ensures that the dataset is free of missing values, all features are numerical and appropriately scaled, and no data leakage occurs in later modeling steps.

## Question 2: L1-regularized logistic path and most important features

In this step, we computed the coefficient path for a logistic regression model with L1 regularization (lasso) to identify the most important predictors of hospital readmission within 30 days. Since this is a classification problem, the standard `lasso_path` function from scikit-learn (designed for regression) is not applicable. Instead, we implemented an L1-penalized logistic regression using the `saga` solver, which supports sparse matrices and warm starts.

**Methodology.** To reduce computation time while preserving accuracy, a two-stage approach was used:

1. **Stage A – Fast screening:** An approximate L1 path was computed on a stratified subsample of 30,000 observations using a stochastic gradient descent (SGD) classifier with L1 penalty. This stage provided a ranking of features based on the earliest point (under strong regularization) at which their coefficients became non-zero.

2. **Stage B – Exact path:** A full L1-regularized logistic regression path was then computed on the *entire* dataset, restricted to the top 30 features identified in Stage A. We varied the inverse regularization parameter $C$ over a logarithmic grid from $10^{-3}$ to $10^{2}$, using warm starts to accelerate convergence.

**Results.** The coefficient path was plotted for the top 30 features , showing how their estimated coefficients evolved as regularization was relaxed (i.e., as $C$ increased). Features that enter the model earlier, under stronger regularization, are considered more predictive.

The top three most important features according to the exact L1-logistic path were:

1. `miglitol_No` ($|\beta|_{\max} = 0.6473$) – strong negative coefficient under weak regularization, entering the model earliest; indicates the absence of the diabetes drug "miglitol" is predictive of readmission risk.

2. `number_inpatient` ($|\beta|_{\max} = 0.3322$) – positive association with readmission; more previous inpatient visits correlate with higher likelihood of readmission within 30 days.

3. `chlorpropamide_No` ($|\beta|_{\max} = 0.2604$) – similar to `miglitol_No`, the absence of this drug appears linked to readmission risk.

These results suggest that both medication history and the intensity of prior hospital use are key predictors for short-term readmission. Other variables with substantial coefficients included `max_glu_serum_Norm`, `repaglinide_No`, and `nateglinide_No`.

The L1-logistic path provides a clear way to visualise feature importance in a classification setting, as well as a principled approach to feature selection. Features entering earliest in the path are those that survive the strongest regularization, making them the most informative for the prediction task.

## Question 3(a) — Principal Component Analysis

We applied Principal Component Analysis (PCA) to the preprocessed feature matrix $X \in R^{n \times p}$ in order to extract the leading principal components, which correspond to the top eigenvectors of the covariance matrix. The analysis was performed after centering and scaling all features.

The explained variance ratio for the first four principal components is as follows:

| Component | Explained variance ratio | Cumulative variance |
|---|---|---|
| PC1 | 0.0032 | 0.0032 |
| PC2 | 0.1136 | 0.1167 |
| PC3 | 0.0784 | 0.1952 |
| PC4 | 0.0663 | 0.2615 |

The results show that the first principal component (PC1) accounts for less than 1% of the variance, while PC2, PC3, and PC4 capture approximately 11.36%, 7.84%, and 6.63% respectively. The cumulative variance explained by the first four components is about 26.15%.

This relatively low cumulative percentage indicates that the variance in the dataset is distributed across many dimensions. Such a pattern is expected given the large number of sparse features generated after one-hot encoding the categorical variables. As a consequence, the leading components capture only a small fraction of the total variance, and more components would be required to explain the majority of the variability in the data.

## Question 3(b): Top features for each principal component

In this part, the focus was on the first four principal components from Question 3(a) to identify the features with the highest influence in each one. For each component vector $v_i$, the absolute value of all loadings was calculated, sorted in decreasing order, and the top 5 features were selected.

Using the same PCA (via truncated SVD) as in 3(a), the components remained directly comparable. For each of the first four PCs:

- The absolute loading of each feature was computed.
- Loadings were sorted from largest to smallest.
- The top 5 features were retained for interpretation.

Bar plots were also generated to visualise the most influential features in each PC.

The top 5 features and their absolute loadings for each PC are:

- **PC1:** `miglitol_No` (0.2430), `chlorpropamide_No` (0.2428), `acarbose_No` (0.2424), `nateglinide_No` (0.2415), `glyburide-metformin_No` (0.2415). This PC is almost entirely related to binary indicators for specific diabetes drugs, capturing differences in medication usage.
- **PC2:** `num_medications` (0.5541), `time_in_hospital` (0.5129), `num_lab_procedures` (0.3679), `number_diagnoses` (0.3504), `num_procedures` (0.3232). These are numerical variables linked to the amount of medical care received, so PC2 reflects hospital usage and treatment intensity.
- **PC3:** `number_inpatient` (0.5148), `number_emergency` (0.4834), `num_procedures` (0.3927), `number_outpatient` (0.2921), `admission_type_id` (0.2659). This component combines visit counts of different types with admission type, representing patterns in patient visit history.
- **PC4:** `admission_type_id` (0.6942), `num_lab_procedures` (0.3632), `number_emergency` (0.2775), `number_outpatient` (0.2675), `discharge_disposition_id` (0.2227). Here, `admission_type_id` dominates, suggesting a strong link between this PC and admission types, with additional influence from procedures and discharge type.

Overall, PC1 is driven by medication usage patterns, PC2 by medical care intensity, and PCs 3 and 4 by various aspects of hospital visits and admissions. The results indicate that variation in the dataset is largely explained by treatment choices and hospital usage patterns, which could be valuable for further dimensionality reduction or clustering.

## Question 3(c): Relevance of Principal Components to the Labels

The goal of this part was to evaluate how informative each principal component (PC) is for predicting the target label $y$. Given the $i$-th principal component vector $v_i \in R^p$, we computed its projection scores for all samples:

$$\alpha_i = X v_i$$

where $X$ is the preprocessed data matrix. Each $\alpha_i \in R^n$ represents the coordinates of the samples along PC$_i$.

To measure the relevance of each PC to the binary labels $y$, we calculated the *Precision-Recall Area Under the Curve* (PR-AUC, also referred to as Average Precision, AP) using $\alpha_i$ as the prediction scores. This metric is appropriate for imbalanced datasets because it focuses on the positive class performance. The baseline AP corresponds to the positive rate in the dataset, which in our case is:

$$BaselineAP \approx 0.1116$$

The table below shows the AP values for the ten principal components with the highest scores:

| PC | AP |
|---|---|
| PC3 | 0.1663 |
| PC6 | 0.1490 |
| PC2 | 0.1322 |
| PC4 | 0.1264 |
| PC10 | 0.1239 |
| PC7 | 0.1188 |
| PC1 | 0.1158 |
| PC8 | 0.1130 |
| PC5 | 0.1115 |
| PC11 | 0.1094 |

From the results, PC3 shows the highest AP (0.1663), indicating that it carries the most information relevant to the target labels among all PCs considered. Several other components (e.g., PC6, PC2, PC4) also perform above the baseline, but the differences are relatively small. Many components, including PC5 and PC11, are close to the baseline AP, suggesting limited predictive power.

Overall, this confirms that principal components are not guaranteed to be highly relevant to the prediction task, since PCA maximizes variance in the feature space without considering the label information. Some components capture patterns correlated with the outcome, but many capture variance unrelated to the target.

## Q4 — Model Training and Evaluation

We split the data into a training set (approximately 2/3 of the observations) and a test set (1/3 of the data), preserving the label distribution via stratified sampling.

We trained and evaluated the following models:

- **Logistic Regression**: L2-regularized logistic regression, solver `saga`, `class_weight="balanced"` to address class imbalance.
- **K-Nearest Neighbors (KNN)**: $k = 11$, Euclidean distance metric. Before applying KNN, dimensionality was reduced to 50 components using TruncatedSVD.
- **Kernel SVM (RBF approximation)**: Random Fourier Features (RFF) to approximate an RBF kernel, followed by a LinearSVC. Hyperparameters $\gamma \in \{0.1, 0.2\}$, $n_{components} \in \{500, 800\}$, and $C \in \{0.5, 1.0\}$ were tuned via 3-fold cross-validation on a stratified subsample of 10,000 training examples, using the F1 score.

**Runtime considerations.** Initial attempts to run cross-validation for the Kernel SVM on the full training set resulted in extremely long runtimes. To reduce computation time while still obtaining reliable hyperparameter estimates, we performed cross-validation on a random stratified subset of 10,000 training samples instead of the entire dataset. This significantly shortened execution time without noticeably affecting the quality of the selected hyperparameters.

For each model, we computed the F1 score on both the training and test sets. The results are summarized in Table 1.

Logistic Regression achieved the highest test F1 score (0.251), closely followed by the Kernel SVM (0.240). KNN performed poorly in this high-dimensional setting, likely due to the curse of dimensionality and the low separability of instances in Euclidean space. The results are consistent with expectations for this imbalanced classification problem.

## Q5(a) — Diversity Between Learners

We constructed an $n \times 3$ binary error matrix $E$ based on the training set, where $E_{ij} = 1$ if learner $j$ made an incorrect prediction on observation $i$, and $E_{ij} = 0$ otherwise. The three columns correspond to:

1. Logistic Regression
2. K-Nearest Neighbors (KNN) with $k = 11$ and Euclidean distance (after TruncatedSVD to 50 components)
3. Kernel SVM (RBF approximation via Random Fourier Features + LinearSVC)

We then centered each column by subtracting its mean and normalized it to have unit $L_2$ norm, producing the matrix $\tilde{E}$ satisfying, for each column $j$:

$$\sum_i \tilde{E}_{ij} = 0, \quad \sum_i \tilde{E}_{ij}^2 = 1.$$

The centered and normalized error Gram matrix $\tilde{E}^T \tilde{E}$ is:

$$G = \begin{bmatrix} \mathbf{1.000} & 0.039 & 0.355 \\ 0.039 & \mathbf{1.000} & 0.037 \\ 0.355 & 0.037 & \mathbf{1.000} \end{bmatrix}$$

The diagonal entries are 1 by construction. The off-diagonal entries measure the correlation between the error patterns of each pair of models:

- Logistic Regression and KNN: 0.039 (almost uncorrelated errors)
- KNN and Kernel SVM: 0.037 (almost uncorrelated errors)
- Logistic Regression and Kernel SVM: 0.355 (moderate correlation)

Low off-diagonal values indicate high diversity between models' errors. This suggests that combining the models could yield some improvement.

We tested a simple majority voting scheme on the test set, predicting class 1 if at least two of the three models predicted 1. The individual and combined F1 scores were:

- Logistic Regression: 0.251
- KNN: 0.039
- Kernel SVM: 0.240
- Majority vote: 0.255

The majority vote slightly improved the test F1 score from 0.251 (best individual model) to 0.255. This modest gain is consistent with the observed moderate diversity between Logistic Regression and Kernel SVM, and the very low performance of KNN. While diversity exists, the improvement from combining these three learners is limited.

## Q5(b) — Meta Learner via Majority Voting

We generated a simple meta learner that combines the three base learners (Logistic Regression, KNN, and Kernel SVM) using *majority voting.* Let $\hat{y}_j(x) \in \{0, 1\}$ be the prediction of learner $j \in \{1, 2, 3\}$. The meta-learner predicts class 1 if at least two of the three learners predict 1:

$$\hat{y}_{MV}(x) = \mathbf{1}\left\{\sum_{j=1}^{3} \hat{y}_j(x) \geq 2\right\}.$$

Using the same train/test split as in Q4 and Q5(a), the F1 scores were:

| **Model** | **Train F1** | **Test F1** |
|---|---|---|
| $LogisticRegression$ | 0.295 | 0.251 |
| $KNN(Euclidean)$ | 0.070 | 0.039 |
| $KernelSVM(RFF)$ | 0.273 | 0.240 |
| **Meta-learner (Majority Vote)** | **0.309** | **0.255** |

The majority-vote meta-learner slightly improved the test F1 score from 0.251 (best single model) to 0.255. This modest gain is consistent with the diversity analysis in Q5(a), where off-diagonal entries of the centered/normalized error Gram matrix were low to moderate, indicating that some errors are uncorrelated and can be corrected through combination.

While majority voting is simple and effective, it treats all models equally and uses only hard labels. Possible improvements include:

- **Soft/weighted voting:** Combine calibrated probabilities from each model and assign higher weights to stronger learners (e.g., Logistic Regression and Kernel SVM), downweighting the weaker KNN.
- **Stacking:** Train a meta-classifier (e.g., logistic regression) on out-of-fold predicted probabilities from the three base learners, then tune the decision threshold to maximize F1 on a validation set.

- **Model pruning:** Remove or heavily downweight very weak learners whose predictions add mostly noise (here, the KNN).

Such methods exploit confidence information and allow the combination strategy to be data-driven, typically yielding greater improvements than unweighted majority voting, especially for imbalanced classification tasks.

| **Model** | **Hyperparameters** | **Train F1** | **Test F1** |
|---|---|---|---|
| Logistic Regression | `solver=saga`, `class_weight=balanced` | 0.295 | 0.251 |
| KNN (Euclidean) | SVD=50, $k = 11$, Euclidean distance | 0.070 | 0.039 |
| Kernel SVM (RFF + LinearSVC) | $\gamma = 0.1$, $n_{comp} = 800$, $C = 0.5$ | 0.273 | 0.240 |

Table 1: Training and test F1 scores for the three models in Q4. Hyperparameters for Kernel SVM were chosen using cross-validation on a 10,000-sample subset of the training set.

## Unsupervised learning

1.In the 2024 quarterly crime file published by the Israel Police, each entry describes a specific criminal case reported during the year. The data includes when the case occurred (year and quarter), where it took place (city and municipality codes and names), and which police district, sub-district, and station handled it. It also specifies the statistical area for more precise location information. Finally, each record classifies the offense through a crime group code and a more detailed type code, making it possible to group and compare different kinds of criminal activity across cities.

2. In order to create the City × Crime Type table, we began by processing the crimes_2024.csv dataset. Only the columns relevant to our analysis were retained, namely the city identifier (YeshuvKod), the crime category (StatisticGroupKod), the crime type code (StatisticTypeKod), the year, and the quarter. The dataset was filtered so as to include only incidents recorded in 2024. Using the StatisticGroupKod as a categorical variable for the type of crime, we aggregated the data by city and crime category, counting the number of occurrences for each combination. The resulting structure was reshaped into a matrix in which each row corresponded to a city and each column represented a specific crime category, with missing values replaced by zeros.

To enrich this dataset with demographic and geographic information, we imported the cities.xls file, which contains population size and coordinate data for each city. The relevant columns were selected, renamed to match our format, and merged with the crime table using the YeshuvKod key. We proceeded to do the cleaning as so, first, we removed any city with missing population or non positive population values and discarded entries with invalid coordinates. We then transformed coordinates into two separate variables (x and y), corresponding to longitude and latitude.

After these steps, the resulting table contained 213 cities and 15 crime categories, each row representing a city's complete crime profile for 2024, with its population size and geographic location. This dataset provides the basis for the similarity and Laplacian matrix computations.

3. To represent the relationships between cities in terms of their crime patterns, we projected the problem as a weighted undirected graph in which each node corresponds to a city and each edge weight represents the similarity between the two cities crime profiles. The input data for this process came from the cleaned City × Crime Type table described in the previous question , where each row contained the number of crimes per category for a specific city, along with its population size.

Since it was asked in the guidelines that the population size be taken into account, we first normalized the raw crime counts by dividing them by the population of the corresponding city and multiplying the result by 100000. Thus,this transformation yields crime rates per 100000 inhabitants.

We then computed the pairwise similarities between cities using cosine similarity, a metric that measures the angle between two vectors in a multidimensional space. Here, each vector represents a city's normalized crime profile across all categories. The cosine similarity between cities i and j is given by:

$$similarity(i, j) = \frac{\sum_k r_{i,k} \cdot r_{j,k}}{\|r_i\| \|r_j\|}$$

where $r_{i,k}$ is the crime rate of type k in city i, and $\|r_i\|$ is the Euclidean norm of the vector for city i. Negative similarities were set to zero and self similarities were not explicitly stored as weights. The result was a symmetric weight matrix W of size n×n, with n the number of cities.

From W, we computed the degree matrix D, a diagonal matrix whose entries correspond to the sum of weights connected to each city. The unnormalized Laplacian matrix was then obtained as: L=D−W.

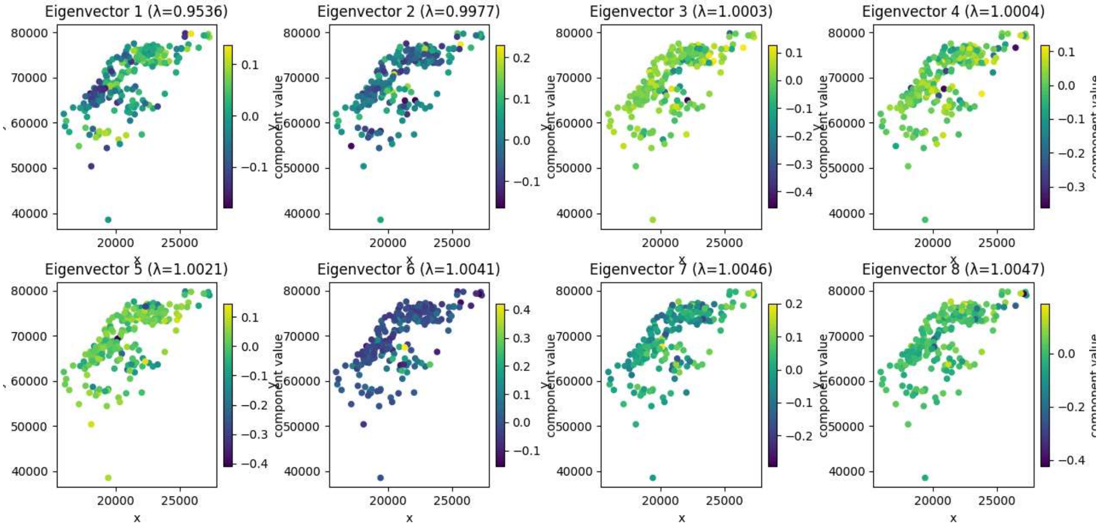
To account for differences in node connectivity, we also computed the normalized symmetric Laplacian:

$$L_{sym} = I - D^{-1/2} - WD^{-1/2}$$

This normalized form is widely used in spectral graph theory and graph based machine learning, as it balances the contribution of nodes with different degrees and facilitates comparison between graphs. This is why we thought it more adapted in this situation. The resulting Laplacian matrices form the basis for subsequent analyses such as spectral clustering.

4. After computing the normalized Laplacian $L_{sym}$, we performed its spectral decomposition to obtain the eigenvalues and eigenvectors. The "leading" or "smoothest" eigenvectors correspond to those associated with the smallest non-trivial eigenvalues, which capture the largest-scale, smooth variations over the graph. We excluded the trivial eigenvector associated with $\lambda \approx 0$ and retained the first eight non-trivial eigenvectors.

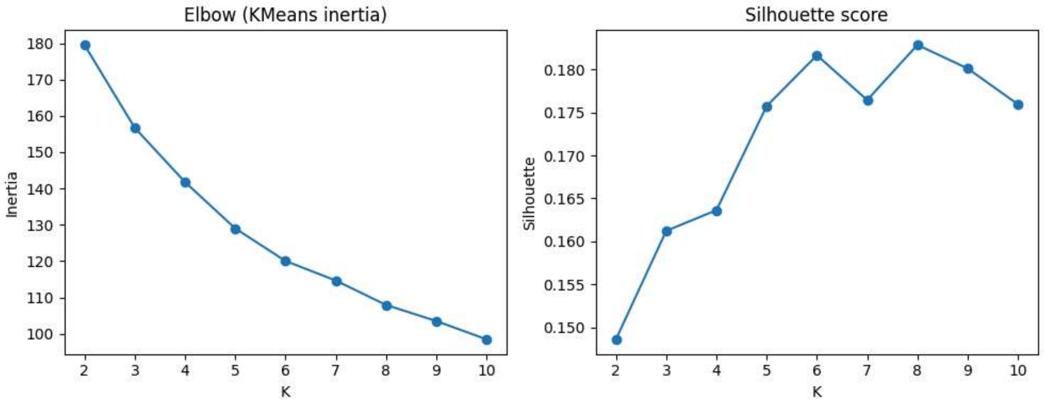
For visualization, each city was plotted at its real geographic coordinates (x,y) extracted from the cities.xls file. The points were then colored according to the value of a given eigenvector, producing eight separate scatter plots (one per eigenvector). This representation allows us to observe spatial patterns in the spectral domain, where similar colors in neighboring cities indicate similar positions in the graph. These smooth modes are particularly relevant for tasks such as spectral clustering, as they provide a low dimensional embedding that preserves the graph's global structure.



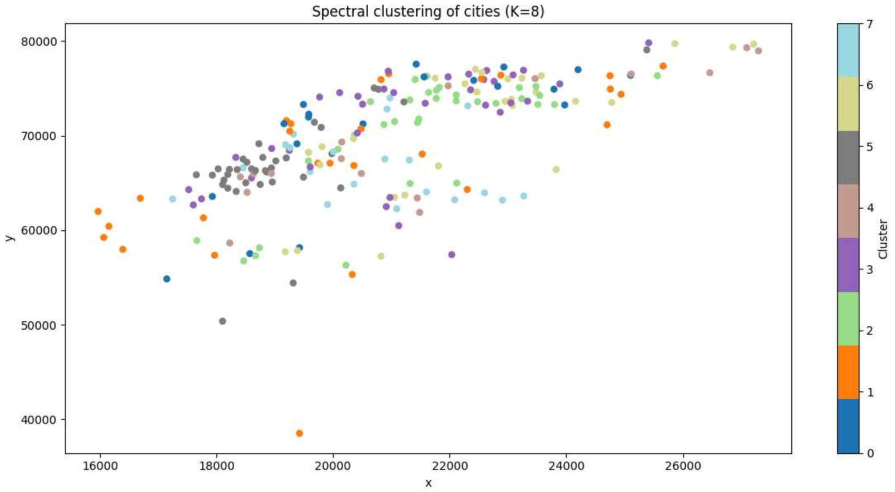
**5.** To perform spectral clustering, we first computed the normalized Laplacian of the similarity graph and extracted the eigenvectors corresponding to the smallest non-trivial eigenvalues. These eigenvectors were used as the feature space for a KMeans clustering procedure.

Since the optimal number of clusters K is not known a priori, we evaluated several values of K using two complementary approaches. The elbow method, which inspects the decrease in KMeans inertia as K increases.
The *silhouette score*, which measures the average separation between points in different

clusters relative to the cohesion within each cluster.



The elbow plot shows a clear change in slope near K=8, and the silhouette score reaches its maximum at the same value. Based on this agreement, we selected K=8 for the final clustering.



The resulting clusters are not sharply separated on the map. However we tried other values of K and different similarity definitions (like cosine on z-scored rates or RBF kernel), but they gave similar or worse separation. The configuration we used finally , cosine rates with a 10-NN graph and K=8was the most stable and interpretable. Even if boundaries are soft, the spectral embedding still captures meaningful structure in the data, so the chosen K is a reasonable choice.

6. We linked the first eight Laplacian eigenvectors to the city variables in cities.xls For each eigenvector, ,meaning each feature pair, we computed Pearson correlation and also Spearman correlation, then adjusted pvalues with Benjamini Hochberg (FDR 5%). After correction, no feature was significant. This suggests the eigenvectors are not driven by any single city attribute but reflect combined differences in crime profiles across cities. As a robustness check, we also try and run partial correlations controlling for population and the spatial coordinates (x, y). The conclusion was the same (there does not exist  $q < 0.05$).

# Hypothesis Testing and Multiple Testing on Israel Police Crime Data

## Research Question

Does the distribution of specific crime types recorded in Israel differ significantly between the years 2023 and 2024?

## Hypotheses

For each crime type $i$:

- **Null hypothesis** $H_0$: The proportion of recorded incidents for crime type $i$ in 2023 is equal to the proportion in 2024.
- **Alternative hypothesis** $H_1$: The proportion of recorded incidents for crime type $i$ in 2023 is different from the proportion in 2024.

## Test Applied and Justification

We applied Pearson's Chi-squared test of independence for each crime type (`StatisticType`) and for each crime group (`StatisticGroup`). This test is appropriate because:

- The variables of interest are categorical (crime type/group and year).
- We have count data for each category, and we want to assess whether the distribution across years is independent of the category.
- The Chi-squared test does not require assumptions about the underlying distribution of the data and is widely used for frequency comparison.

## Multiple Testing Correction

Since we performed one statistical test for each of the 162 crime types (and separately for the 15 crime groups), there is a high risk of Type I errors due to multiple comparisons. To control the false discovery rate (FDR), we applied the Benjamini–Hochberg procedure with $\alpha = 0.05$. This approach is less conservative than Bonferroni and retains more statistical power while controlling the expected proportion of false positives.

## Results

**Dataset Summary** The dataset contains:

- 422,607 records for 2023
- 394,453 records for 2024

- 162 unique crime types
- 15 unique crime groups

**Significant Crime Types (after BH correction)** Among the 162 crime types, several remained statistically significant after FDR correction. The top examples, ordered by adjusted $p$-value, include:

1. (*Public Order Offenses*) – $p_{\text{adj}} \approx 3.21 \times 10^{-114}$
2. (*Stone Throwing*) – $p_{\text{adj}} \approx 3.74 \times 10^{-81}$
3. (*Infiltration*) – $p_{\text{adj}} \approx 4.82 \times 10^{-63}$
4. (*Serious Bodily Harm*) – $p_{\text{adj}} \approx 2.64 \times 10^{-48}$
5. (*Other Security Offenses*) – $p_{\text{adj}} \approx 1.00 \times 10^{-47}$

All of these show large differences in proportions between 2023 and 2024.

**Significant Crime Groups (after BH correction)** Among the 15 crime groups, the following remained statistically significant:

1. (*Security Offenses*) – $p_{\text{adj}} \approx 1.14 \times 10^{-180}$
2. (*Offenses Against the Person*) – $p_{\text{adj}} \approx 2.76 \times 10^{-29}$
3. (*Moral Offenses*) – $p_{\text{adj}} \approx 3.10 \times 10^{-16}$
4. (*Offenses Against a Person*) – $p_{\text{adj}} \approx 5.03 \times 10^{-15}$
5. (*Economic Offenses*) – $p_{\text{adj}} \approx 1.15 \times 10^{-8}$

## Conclusion

The comparison between 2023 and 2024 shows that certain crime types and groups changed significantly from one year to the next. Offenses related to security, public order, and violent acts are among the categories with the strongest differences, with very small adjusted p-values. These results mean that the frequency of these crimes did not remain stable between the two years. The changes could be due to various reasons such as law enforcement priorities, social or political events, or other external factors. Further work could look at specific regions or months to understand these patterns better.

## Use of large language models (LLM) within the project

We used a language model only for auxiliary tasks like organization and to help us reduce and improve the time complexity of certain parts of the code like reading the data which was very big - especially in the supervised learning in the second question.