Who Is The Killer?

Piraeus Vice Pattern Recognition Project

Student 1: Βασίλαινα Μαρία (ID: Π23015)
Student 2: Γρηγοράσκου Τερέζα (ID: Π22037)
Student 3: Λιβέρης Φώτης (ID: Π23104)

Department of Informatics
University of Piraeus

February 19, 2026

Abstract

In this project, we addressed the killer identification problem as a multiclass classification task. We first estimated class-specific means and covariances using Maximum Likelihood Estimation (MLE), and used them to build a Gaussian Bayes classifier. We then implemented a linear classifier, a non-linear SVM (RBF kernel), a two-hidden-layer MLP neural network, and an unsupervised PCA + k-means approach with majority-vote label mapping. All preprocessing (standardization and one-hot encoding) was fitted on the TRAIN split and applied to VAL and TEST to avoid data leakage. PCA was used both for visualization and as a preprocessing step for k-means. On the validation set, non-linear models (SVM and MLP) achieved the best performance, outperforming the Gaussian Bayes and linear classifiers. The PCA + k-means approach achieved lower accuracy, indicating that natural clusters in feature space do not perfectly align with killer identities. Overall, the features that proved to be of the most importance were those related with the victim and the crime scene. Last but not least, modeling nonlinear decision boundaries significantly improved classification performance.

Collectively, in order to classify data and reach our final prediction we followed the following steps.

- MLE (Maximum Likelihood Estimation)
- Gaussian Bayes classifier
- Linear classifier
- SVM (Sypport Vector Machine)
- MLP (Multi-Layer Preceptron)
- PCA (Principal Component Analysis)
- k-means

Contents

1 Introduction

In the current project we were asked to identify the most likely killer for each crime incident in the "Piraeus Vice" data set.

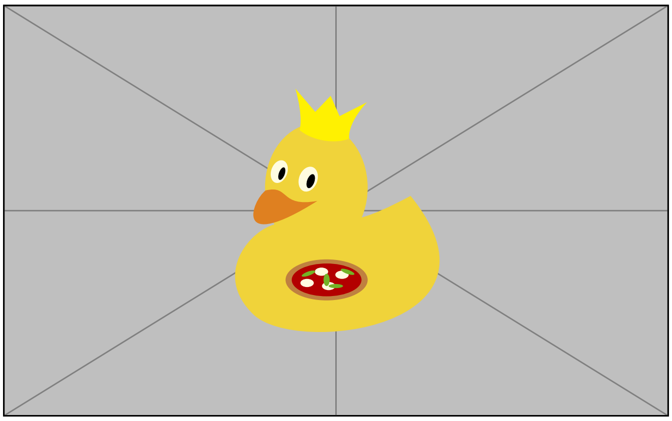


Figure 1: Humorous placeholder: the "prime suspect" before any machine learning. Replace this with a real illustrative figure (e.g., a high-level pipeline diagram of your method).

2 Data description

The csv file we were given consists of a total of 4.800 rows, each one representing one crime incident. Also, it consists of 15 columns, one for each feature. An overview of these features and their corresponding types follows in section 2.1. The dataset contains a predefined split column that is either TRAIN, VAL or TEST. This means that the data is already partitioned for modeling and evaluation. As far as we can tell the dataset is not standardised or normalised.

2.1 Feature overview

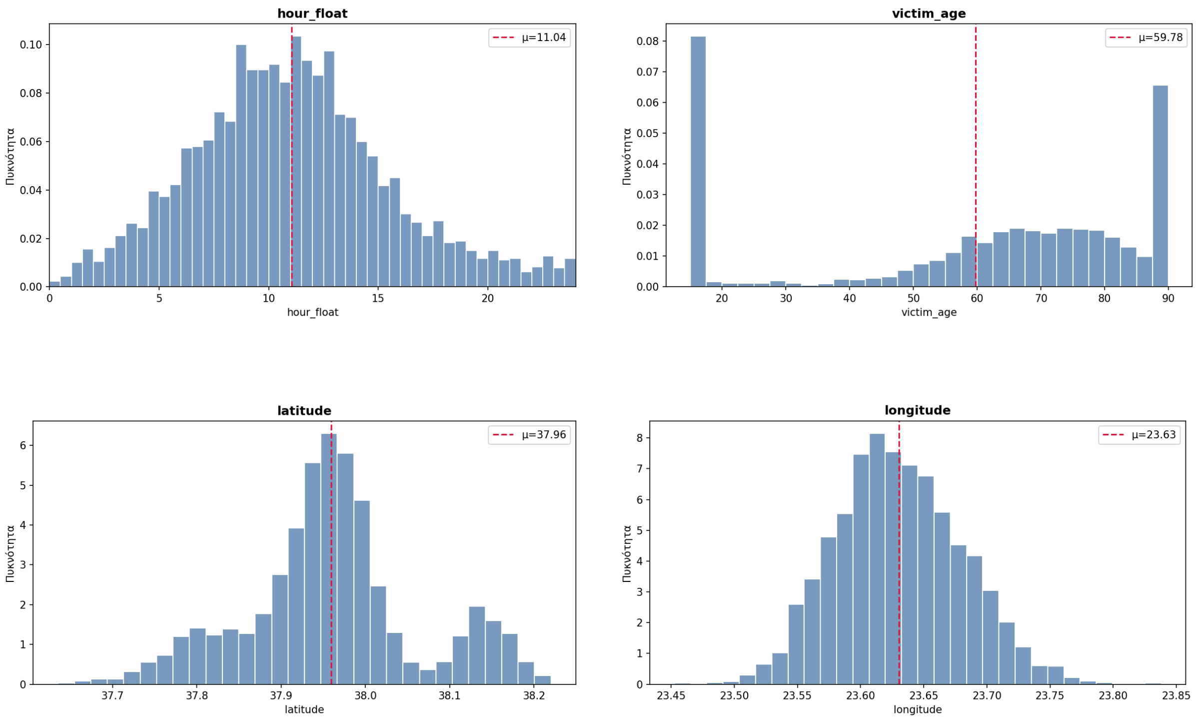
Table 1: Feature summary.

Feature name	Type
incident_id	identifier
hour_float	continuous
latitude	continuous
longitude	continuous
victim_age	continuous
temp_c	continuous
humidity	continuous
dist_precinct_km	continuous
pop_density	continuous
weapon_code	categorical
scene_type	categorical
weather	categorical
vic_gender	categorical (binary)
split	categorical
killer_id	identifier

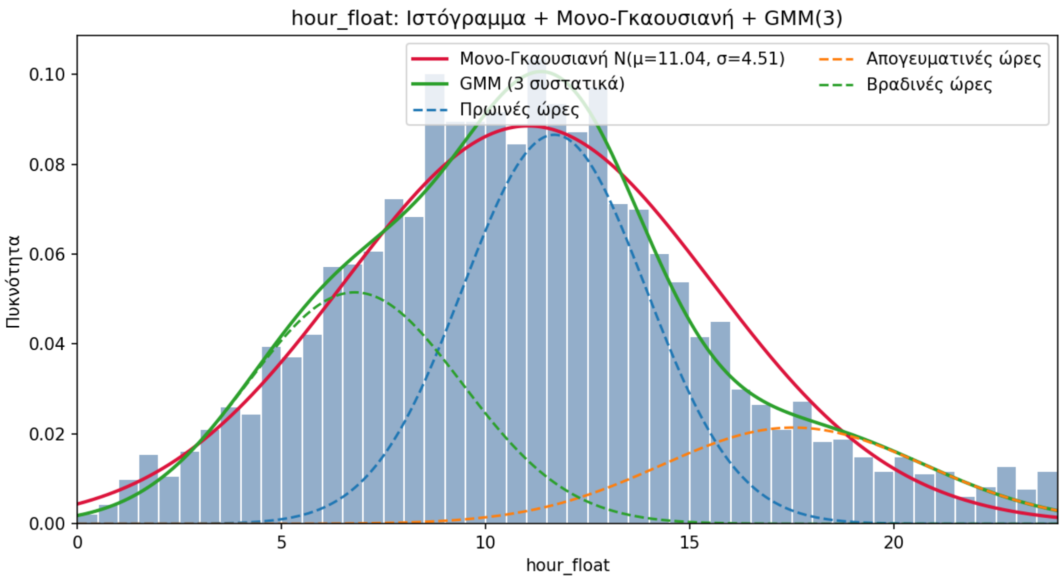
3 Results

Q1: Exploratory analysis

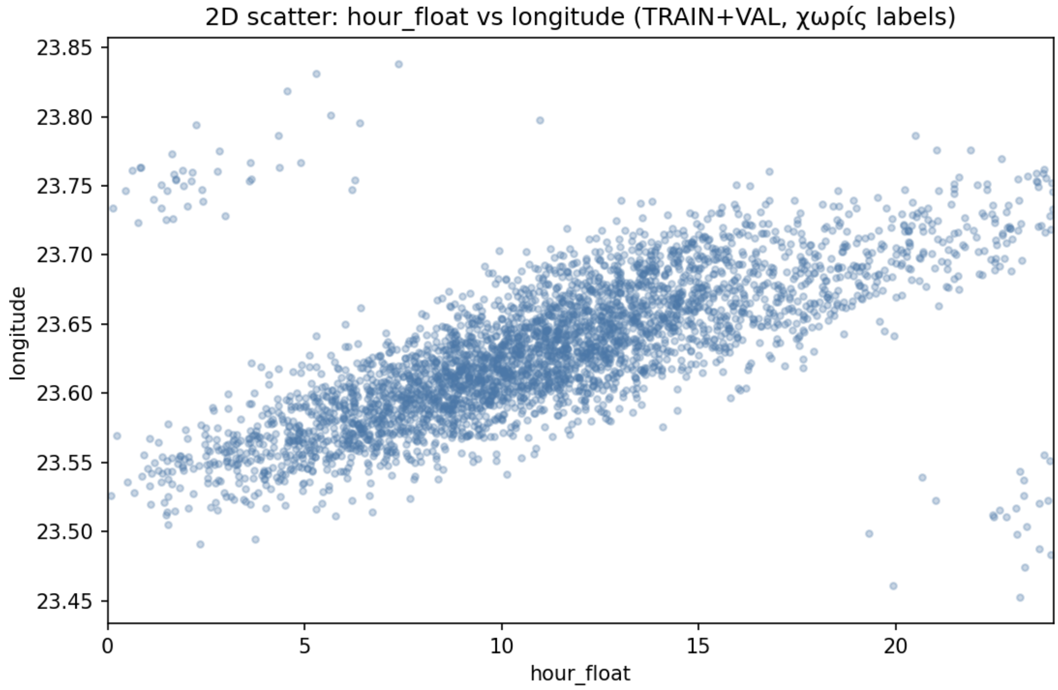
The images that follow are the histograms for the key continuous variables (hour, age, latitude, longitude) that we produced.



Below is the comparison of the single Gaussian fit and a mixture of Gaussians for `hour_float`.



Also, we provide you with the additional two-dimensional plot involving `hour_float` and `longtitude`. using TRAIN+VAL and without using any labels, as requested.



Our observations around the 4 key continuous variables are the following:

- Firstly, about the variable `hour_float` the incidents are at their peak during daytime, which is somewhat unexpected.
- As far as the variable `victim_age` is concerned it can easily be observed that the selected victims were either of a very young age or of a very old. With that said, we can safely come to the conclusion that they were targeted because they belonged to the most vulnerable age groups.
- The histogram of the variable `latitude` presents some fluctuations that are at their core closer to its average value.
- Lastly, concerning the variable `longtitude`, by observing both the corresponding histogram as well as the 2D scatter, it is clear that the incident density is also thicker closer to its average value.

Regarding the single Gaussian, we believe that it is inadequate. We can easily see that it is not symmetric. But it is not solely that. It also ignores some of the peaks. As a result it fails to represent correctly the given data. By splitting it to the 3-component Gaussian Mixture we achieve better interpretation of the times of the day as well as better representation of the data as it focuses on distinct times of the day.

Explanation

We have decided to use the library GaussianMixture that proved to be very usefull. Also, we created the following function in order to be able to create the histograms. As we can see it requires 5 parameters, 3 of which have default values in case they don't receive any. The parameters are:

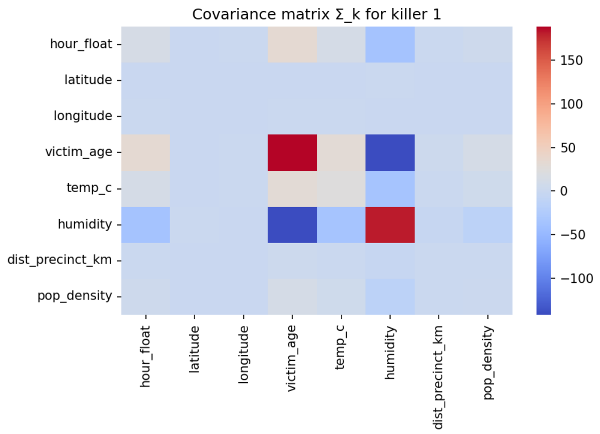
- series: a pandas series containing numerical data,
- title: it describes the column we are focusing on,
- bins: the number or range of lines that the histogram will consist of,
- kde: it controls whether the result will be a histogram or a smooth curve that represents the distribution of the data,
- xlim: it limits the range of the x-axis.

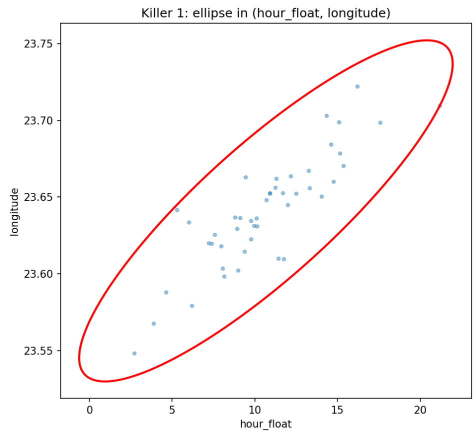
```
def plot_hist_with_stats(series, title, bins=30, kde=False, xlim=None):
    s = series.dropna().values
    fig, ax = plt.subplots(figsize=(8, 4.5))
    sns.histplot(s, bins=bins, stat="density", kde=kde, edgecolor="white", color="#4C78A8", ax=ax)
    if xlim:
        ax.set_xlim(xlim)
    ax.set_title(title, fontweight="bold")
    ax.set_xlabel(title)
    ax.set_ylabel("Π     ")
    mu, sigma = np.mean(s), np.std(s, ddof=1)
    ax.axvline(mu, color="crimson", linestyle="--", label=f" ={mu:.2f}")
    ax.legend()
    plt.tight_layout()
    return fig, ax, mu, sigma
```

Q2: Gaussian MLE per killer

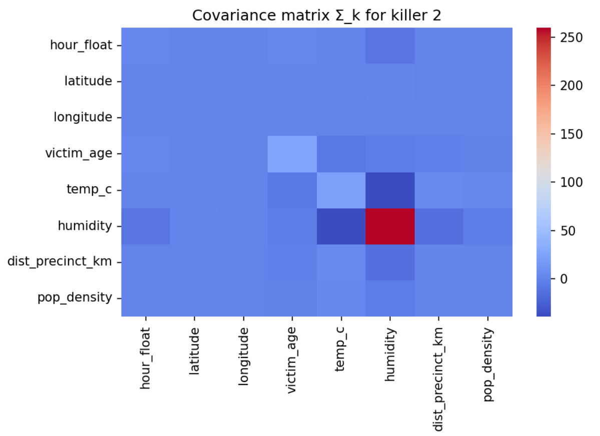
Bellow we present you one covariance heatmap and one 2D projection with ellipses per killer, as requested.

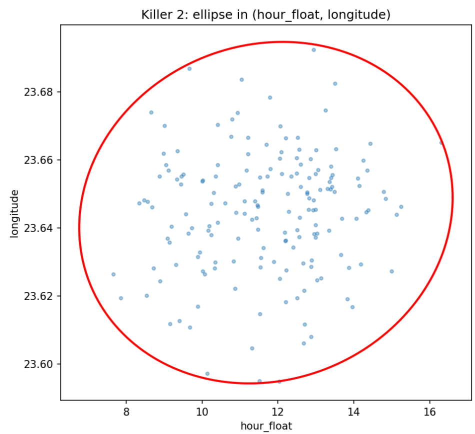
- Killer 1:



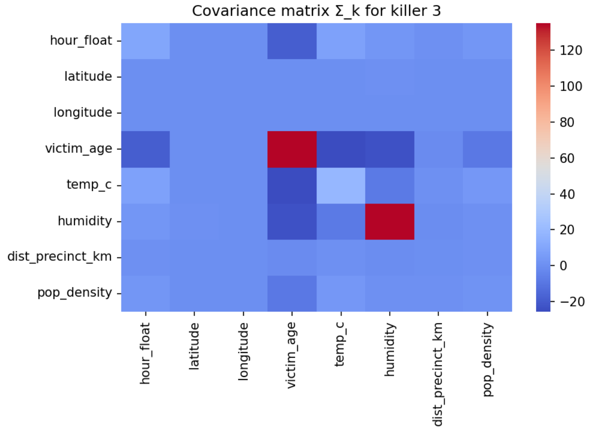


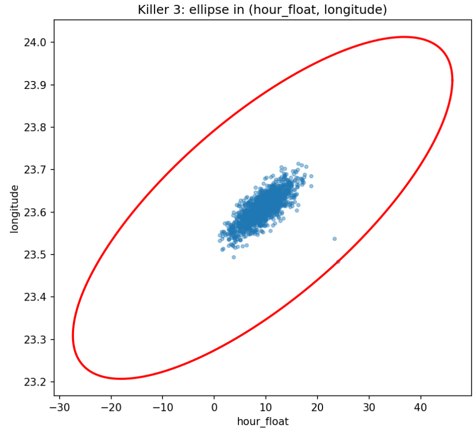
- Killer 2:



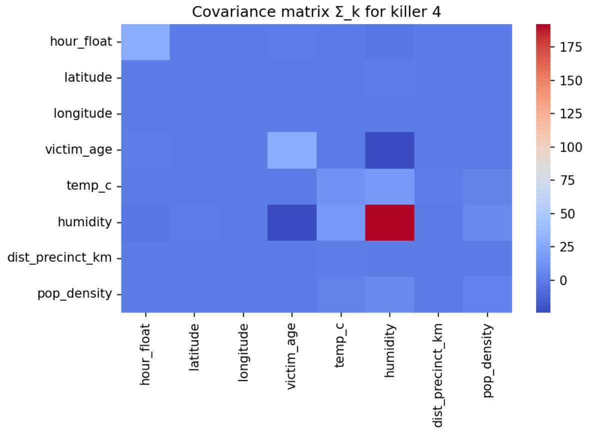


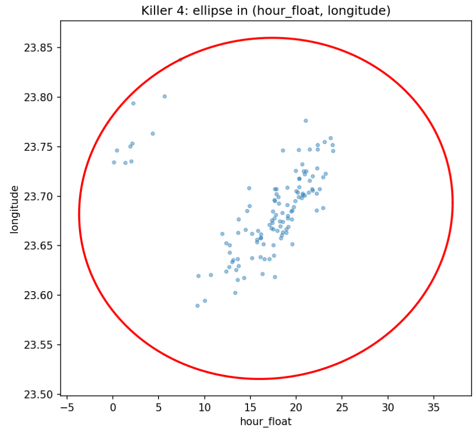
- Killer 3:



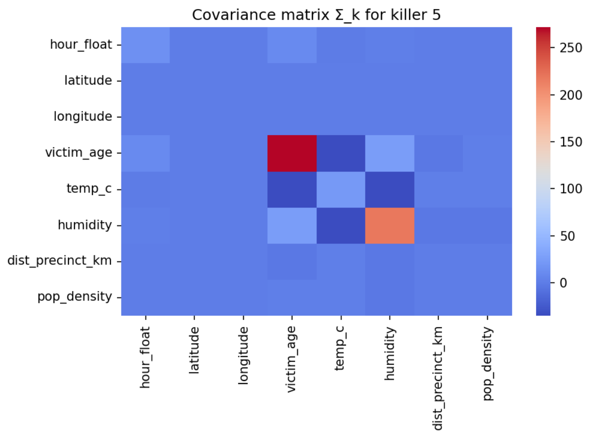


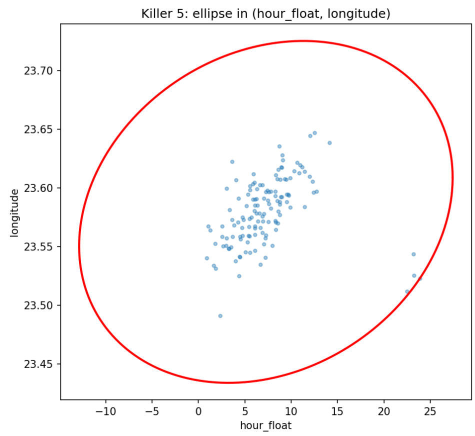
- Killer 4:



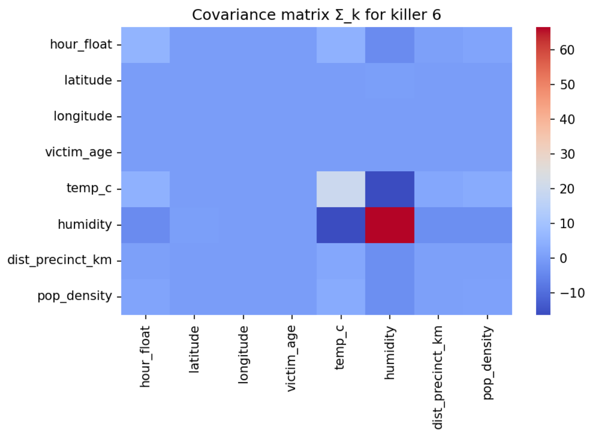


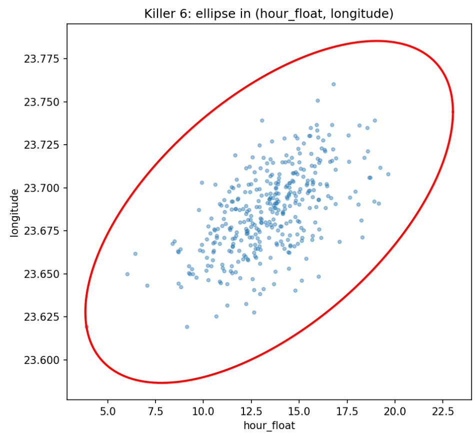
- Killer 5:



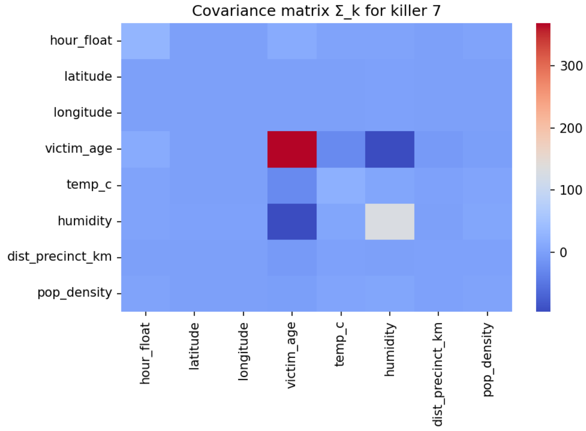


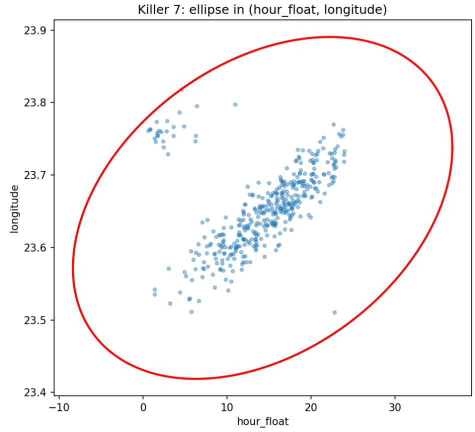
- Killer 6:



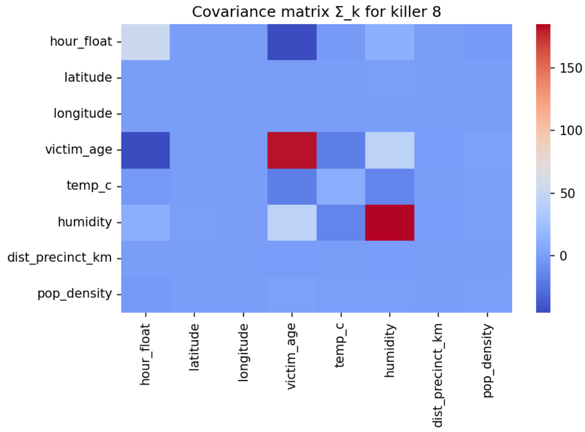


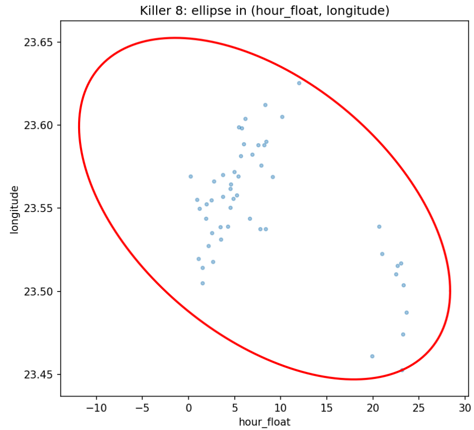
- Killer 7:





- Killer 8:





Explanation

To estimate the mean and covariance of the continuous features for each killer on TRAIN we assumed that, for each killer k, the continuous feature vector follows a multivariate Gaussian distribution. To estimate the parameters we used Maximum Likelihood Estimation (MLE), restricting ourselves only to the TRAIN split, where the killer labels are known. For each killer k, we collected all TRAIN incidents commited by that killer. This way, each killer's profile starts to assemble. Lastly, we visualise our results for better interpretation.

Q3: Multiclass Gaussian Bayes classifier

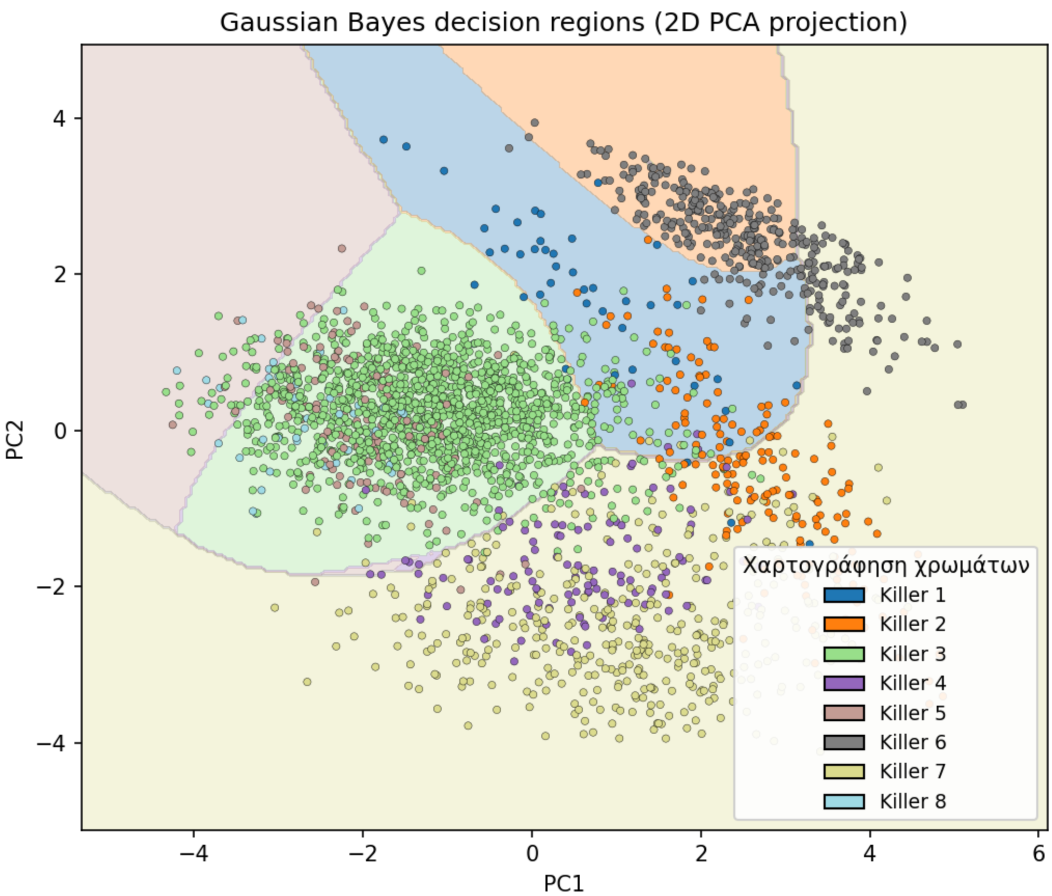
We evaluated the TRAIN accuracy and the VAL accuracy and concluded that they are equal to 89% and 90% respectively. Here is the requested confusion matrix on VAL.

Table 2: Confusion matrix for the Gaussian Bayes classifier on VAL.

True / Pred	1	2	3	4	5	6	7	8
1	13	1	2	0	0	1	0	0
2	1	59	1	0	0	1	0	0
3	3	0	474	0	5	0	7	2
4	0	0	2	38	1	0	4	0
5	0	0	32	1	11	0	1	3
6	0	0	0	0	0	133	0	0
7	1	2	0	3	0	0	137	0
8	0	0	15	0	2	0	0	2

We can see that the matrix prodyced is mostly diagonal and the killers with the highest presence are the killers 3, 6 and 7.

Below follows the two-dimensional projection, visualising the decision regions induced by the Bayes classifier, with TRAIN points coloured by true killer. Each curved boundary seperates the feature regions of each killer.



Explanation

We implemented a multiclass Gaussian Bayes classifier using the parameters estimated in Q2. For each killer k, we modelled the continuous feature vector as a multivariate Gaussian distribution. For each incident x we calculated the long-posterior probability. Lastly, we evaluated the classifier on the VAL split by calculating the overall accuracy and of course prosuced the corresponding confusion matrix.

Q4: Linear classifier

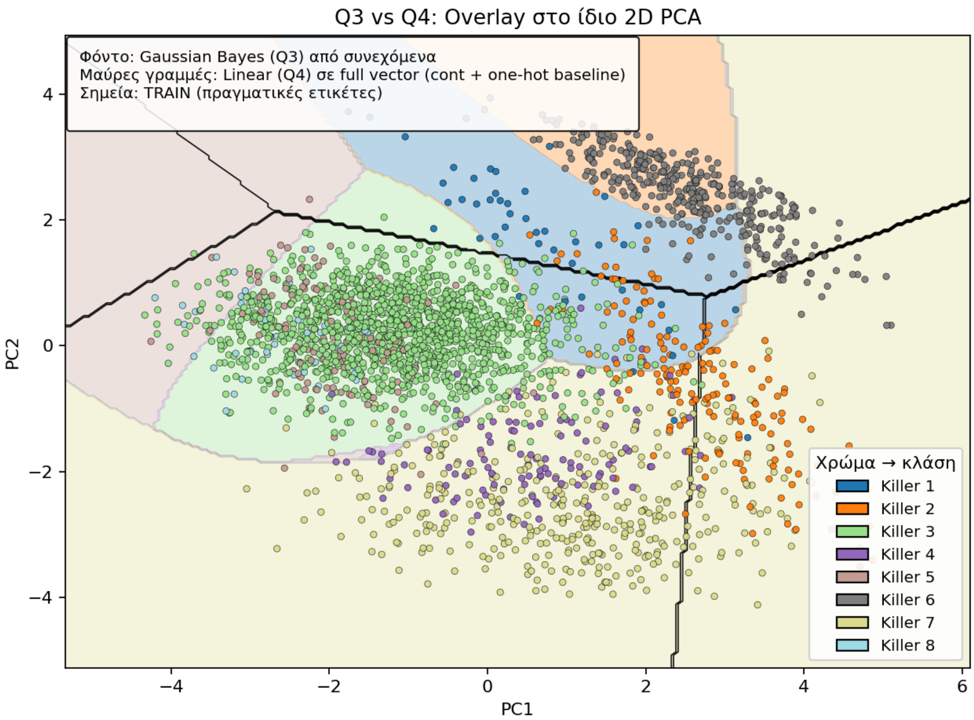
This time, the TRAIN accuracy and the VAL accuracy are equal to 77% and 78% respectively. We need to highlight that these numbers are lower than Q3's. That could mean that the Gaussian assumption is more accurate. Here is the confusion matrix on VAL.

Table 3: Confusion matrix for the Gaussian Bayes classifier on VAL.

True / Pred	1	2	3	4	5	6	7	8
1	0	1	9	0	2	4	1	0
2	1	38	17	0	0	3	3	0
3	0	4	473	0	4	1	9	0
4	1	0	4	0	0	0	40	0
5	0	1	28	1	14	0	4	0
6	0	3	12	0	0	117	1	0
7	3	4	26	0	1	1	108	0
8	0	0	12	0	4	0	3	0

Comparing it to Q3's matrix it obviously does not resemble a diagonal matrix which may mean that we have strayed from our end goal. While some numbers have slightly changed(e.g. 474 -> 473), most of them have completely changed. Also, the last column consists entirely of zeros. However, the killers with the highest presence remain the same.

Also, in the 2D PCA projection used in Q3, we overlayed the approximate linear decision boundaries and the result is the following.



When compared to Q3's RCA projection one can easily see that the regions are seperated by lines and clearly determine which one corresponds to each killer. Some may correspond to more than one killers. This happens because the linear model is not able to fully adapt to the shape of the data the way that the Gaussian Bayes does.

Explanation

We treated the killer identification as a discriminative multiclass classification problem using the full vector consisting of the 8 continuous features (hour_float, latitude, longitude, victim_age, temp_c, humidity, dist_precinct_km and pop_density) and 4 more cateforical features (weapon_code, scene_type, weather, vic_gender). We implemented the linear classifier given. We trained the model on the TRAIN split using MSE (Mean Squared Error loss). Lastly, we valuate the accuracy and the confusion matrix and produce the 2D PCA overlaying the one from Q3.

Note to teacher: As you may have noticed if you have run our code, on the terminal also appear the Q3's results. Everything on terminal is exclusively for our convenience in order to make sure all things run smoothly and are executed correctly. It should not consern you as we have provided you with Q4's results in the previous section.

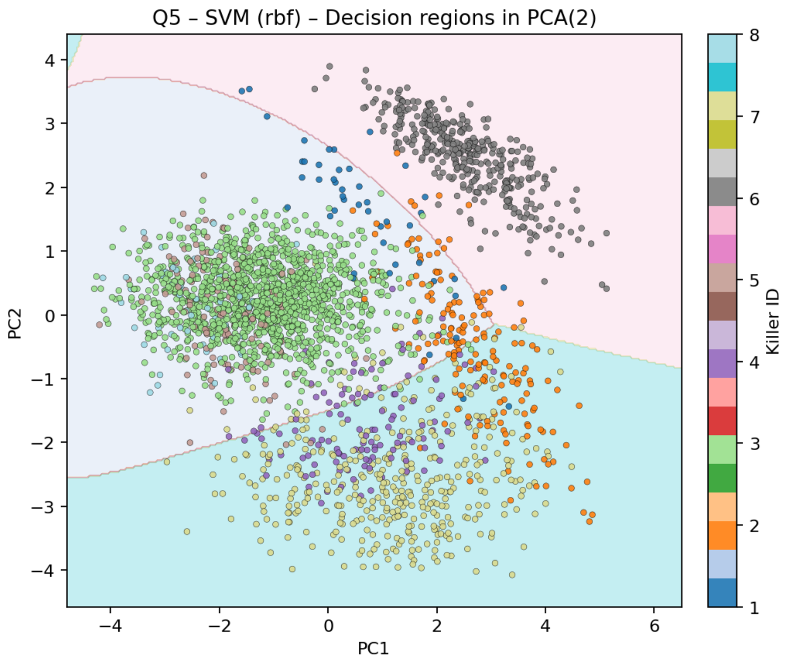
Q5: Support Vector Machine

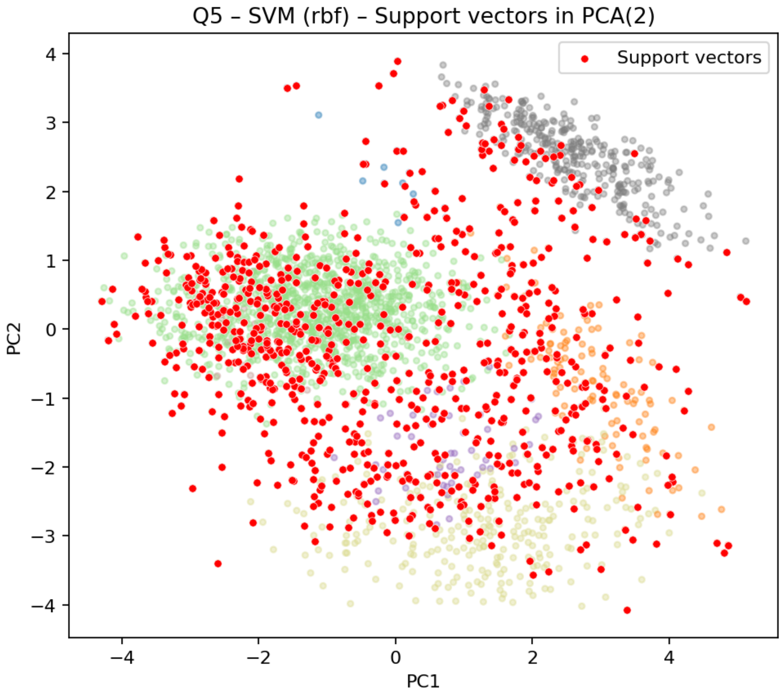
The VAL accuracy for the RBF method is 94%. The confusion matrix is the following. Once again it slightly resembles a diagonal matrix which means we are getting closer.

Table 4: Confusion matrix for the Gaussian Bayes classifier on VAL.

True / Pred	1	2	3	4	5	6	7	8
1	14	0	3	0	0	0	0	0
2	0	62	0	0	0	0	0	0
3	0	0	484	0	1	0	2	4
4	0	0	1	35	1	0	8	0
5	0	0	14	0	30	0	1	3
6	0	0	0	0	0	133	0	0
7	0	0	0	1	1	0	141	0
8	0	0	6	0	3	0	0	10

In the images that follow we can see the decision regions as well as the support vectors.





Explanation

Our SVM setup is formed by choosing either RBF or Polynomial kernel. We prefered RBF as it allows non-linear decision boundaries. The code is modified for either one.

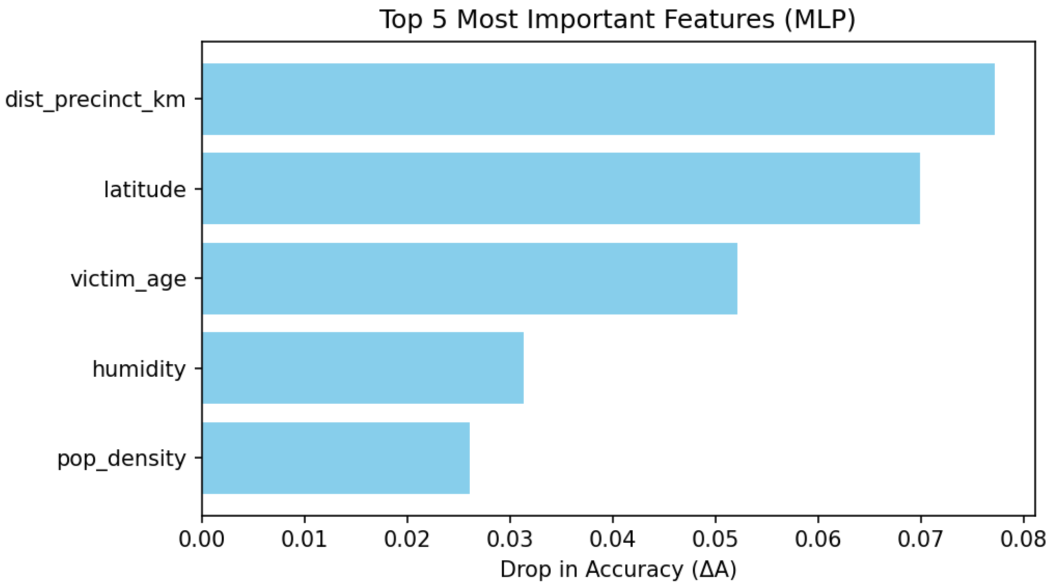
```
def small_search_space(kernel: str):
    if kernel == "rbf":
        Cs = [0.3, 1, 3]
        gammas = ["scale", 0.1, 0.03]
        return [(c, g, None, None) for c in Cs for g in gammas]
    else:
        Cs = [0.3, 1, 3]
        gammas = ["scale", 0.1]
        degrees = [2]
        coef0s = [0, 1]
        return [(c, g, d, c0) for c in Cs for g in gammas for d in degrees for c0 in
        ↪  coef0s]
```

We continued with tuning the hyperparameters using a small manual grin search on VAL. Each candidate model was trained on TRAIN and evaluated on VAL. We chose the one that seemed more suitable to us based on the highest validation accuracy. Latsly, since the SVM is a binary classifier due to inheritance, we used One vs Rest strategy. This way, there is one binary SVN trained per killer, which means we have 8 classifiers, each one able to seperate class k from the rest and as far as prediction is concerned, it is the class with the highest score that is selected. All preprocessing was applied inside a Pipeline thus ensuring there will be no data leakage.

Q6: Multi-Layer Perceptron

The VAL accuracy is equal to 94% (same as the SVM's).
The features of most importance seem to be(in ascending order):

1. dist_previnct_km $\Delta A = 0.07$
2. latitude $\Delta A = 0.06$
3. victim_age $\Delta A = 0.05$
4. humidity $\Delta A = 0.03$
5. pop_density $\Delta A = 0.02$



Explanation

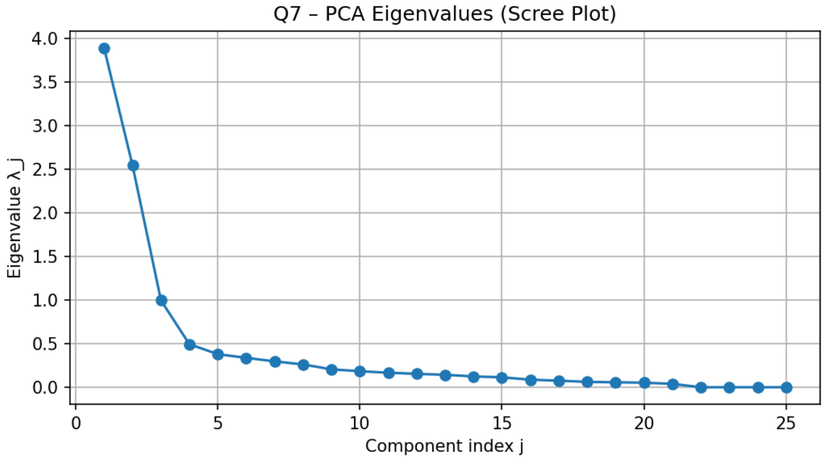
We implemented a Multilayer Perceptron (MLP) as classifier using the following code:

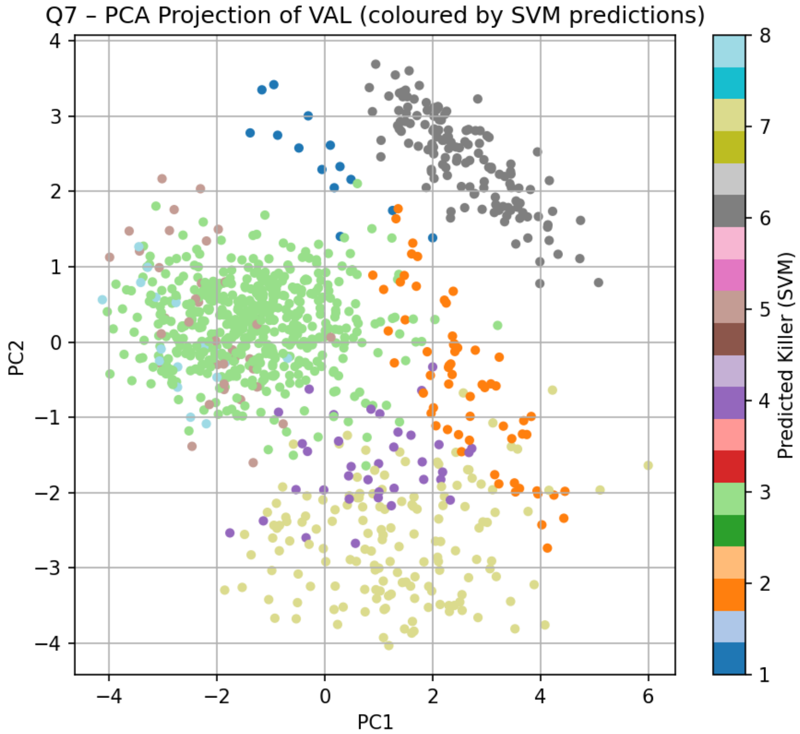
```
mlp = MLPClassifier(
    hidden_layer_sizes=(64, 32),    # 2 hidden layers
    activation="relu",
    solver="adam",
    max_iter=200,
    random_state=0,
    early_stopping=True
)
```

We continued by once again training the model on TRAIN, calculated VAL accuraciy etc. Lastly, feature importance was estimated using permutation importance.

Q7: Principal Component Analysis

Bellow we provide you with a plot of eigenvalues versus component index and a PC1–PC2 scatter plot coloured by predicted killer labels, as requested.





Explanation

Before applying PCA, we started off by preprocessing the features this way once again ensuring no data leakage.

```
pre = ColumnTransformer([
    ("scaler", StandardScaler(), CONT),
    ("onehot", OneHotEncoder(handle_unknown="ignore", sparse_output=False), CAT)
])
```

Subsequently, PCA was fitted on TRAIN only and then applied to both TRAIN and VAL. Firstly, we let PCA run without any restrictions on the number of components. However, for visualization purposes only we chose to set the parameter n_components to 2 to achieve projection in R^2.

Q8: k-means in PCA space

Explanation

To start with, the data was once again preprocessed, fitted on TRAIN and applied on VAL and TEST. The PCA was applied on TRAIN only. The number of components was set to 5 (based on Q7's results). So, we ran k-means with k equal to S, where equals to the nnumber of killers(in our case 8). K-means is unsupervised, so the clusters do not correspond directly to the killer IDs. In order to convert clusters into class labels, we used majority voting on the TRAIN set. For each cluster, we identified all TRAIN samples assigned to that cluster, examined their true killer labels, and assigned each cluster the label of the most frequent killer in that cluster. For evaluation on VAL, samples were assigned to clusters using the fitted k-means model, mapped to killer IDs using the majority-vote mapping, and accuracy was of course calculated. Finally, for the TEST set, we produced a probability distribution over the clusters by calculating the Euclidean distance from each sample to all centroids. Distances were converted to scores using score = −distance, and a softmax function was applied to obtain normalized probabilities.

4 Discussion and conclusions

Let's summarise the VAL accuracies of these methods up until now:

Table 5: Validation accuracy for different models.

Model	VAL accuracy
Gaussian Bayes	0.905
Linear classifier	0.782
SVM (RBF kernel)	0.948
MLP	0.944
PCA (+ k-means)	xxxxx

So, in ascending order of performance we have:

1. SVM (Q5)
2. MLP (Q6)
3. Gaussian Bayes (Q3)
4. Linear classifier (Q4)
5. PCA + k-means (Q8)

We have concluded that the more helpful features in order to model behavior where geographic features (latitude and longtitude), victim-related information (their age and gender), information about the place where the crime was commited and the prevailing weather conditions (weather, temperature and scene type) and of course the weapon type (not necessarily in that order). Potentially, more complex generative models could be included and explored. Additionally, there are other factors to could be analyzed and taken into consideration, such as crime frequency patterns, killer's motive or mental health disorders. Overall, the current study highlights the importance of combining proper preprocessing, nonlinear modeling and feature analysis when trying to solve real-world pattern recognition problems.

Code organisation

Briefly describe your Python code structure (scripts, notebooks, helper modules) and how to reproduce:

- the main results and figures in this report;
- the prediction file `submission.csv`.

5 LLM prompts and responses

If you used ChatGPT or any other LLM, list all prompts and the corresponding responses you used during the preparation of this project.

Example format

Prompt:

```
Explain how to compute a covariance matrix from data.
```

Response:

```
[Paste the model's response here...]
```

Comment: We checked the formulas and then implemented them independently.