Overall Approach

We understood that a big factor of the hackathon is the time constraint. There are endless ways to improve our models by adding engineering creative features, combining ML techniques, etc. We quickly understood that the time constraint would play a large part and agreed that our first priority is achieving a **simple**, yet **rational solution** for each task. Therefore, our guiding principle was to **simplify the data and the models used as much as possible**. When faced with a decision between two options, we preferred the more simple, naïve one. This will become apparent in the rest of the document. If then time allowed for it, we would attempt to add some of the many creative ideas we had initially moved on from and remove simplifying assumptions we had made.

The Dataset

After cleaning garbage samples, we visualized the raw data using explanatory plots (Figures 1-2). This allowed us to get visually acquainted with the data, before delving into our solutions. We were able to draw insightful conclusions regarding correlation between geographical characteristics of events relative to its other features.

Task I: Next Event Prediction

Given an ordered sequence of four complete events which took place in Tel Aviv, predict the 5th event.

We split this task into two parts, depending on the 4 features we are required to predict:

1. The type and subtype of the event – a (multi)classification problem.
2. The x-y coordinates of the event – a regression problem.

We saw these as two independent problems, requiring two different ML solutions. We set out to uniquely a model over the data for each problem and predict for each of the two features in each problem **separately**. Combining the prediction of the two models creates the desired predicted 5th event, which we output.

We cleaned the data by clearing all events with garbage values, and additionally made the decision to drop all events that took place outside of Tel Aviv. We felt this was justified and even necessary given the time constraints. This also allowed for a more representative signal in the training data regarding the events the model would ultimately be tested on. Additionally, we decided to drop the type feature and solely classify the subtype, as we may infer the type the model classifies based on its subtype prediction.

After cleaning we ordered the events in chronological order. We crafted a new DataFrame by grouping events in groups of 5, essentially merging 5 rows into one (we will further call these rows 'sequences'). In each sequence, the 5th event was viewed as a label of the sequence. We removed all redundant features of this 5th event, i.e. all features except the two relevant ones that were used as seen labels (for problem 1 we left type and subtype, for 2 we left the x and y coordinates).

We continued by engineering new, insightful features for our model to learn. Leveraging the magvar feature, we created two binary features indicating the driver's heading. We also created dummy variables for street names. The street of which the event took place is crucial for learning, since predicting the next event in a sequence is highly correlated with the streets of the events of the sequence, and additionally gives more context rather than just the x-y coordinates. For example, if a sequence consists of 4 traffic-related events that are geographically far from one another, we may not assume they are correlated. Only once we learn that all four took place on Ayalon can the model make the connection between them. Therefore, even though these dummies vastly expanded our feature set, we felt this was the correct way to encapsulate the crucial street data into our model.

We then split the DataFrame into a train and test set and trained a KNN with k=6 (see Figure 3) and a two Lasso models (with lambdas of around 3.5, see Figure 4) for the classification and regression problems, respectively. Predictions were then made by processing the new 4 events and formatting them into a sequence, before passing it into the model. Each model outputs its two relevant features, and we combine them to create the final prediction consisting of the 4 relevant values.

We predict our model generalizes well, as we leveraged several powerful ML techniques in the process. We only worry in terms of the size of the training set, which seems to be relatively small for a problem of this complexity.

Task II: Next Event Prediction

Given a date, predict the distribution of events in the defined time slots across the nation.

We took a pure statistical approach to this problem, reminiscent of estimators we explored at the beginning of the course. When trying to identify a suitable ML template, we realized that almost all given features do not affect the output of this problem in any foreseeable way. For example, why should any geographical characteristics of events shed light onto the **quantity** of accidents for a specific time slot?

We found ourselves left with the event's type and its updateDate. We thought that the main driver of an event type's quantity is the day of the week and the time of day, as we know traffic patterns highly vary depending on these two data points.

We created a DataFrame with 21 lines; for each of the 7 days of the week, one line for each time slot. Each line records the totals of each event type seen in the 'training' data. Using statistical bootstrapping, we populated the DataFrame with the means with intent of approaching the limit.

Then, given a date, we calculate its day of the week simply and access the relevant figures in the table.

Here, we feel this model will generalize decently. Our main reason for this is that this model, even more so than others, requires **many** samples for it to be accurate in the general case, as we are relying heavily on the Central Limit Theorem. The more samples to 'train' on, the more we approach the limit – see Figures 5-6

Plots

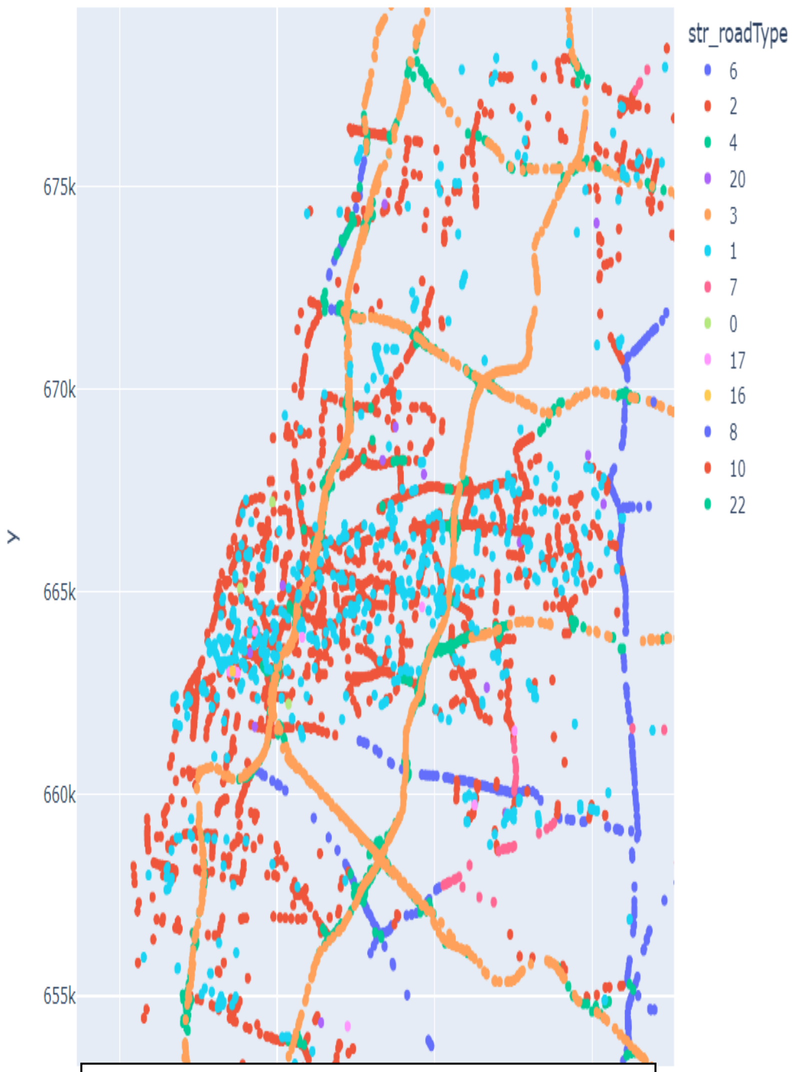

Figure 1: Event coordinates embedded onto the Euclidian plane.

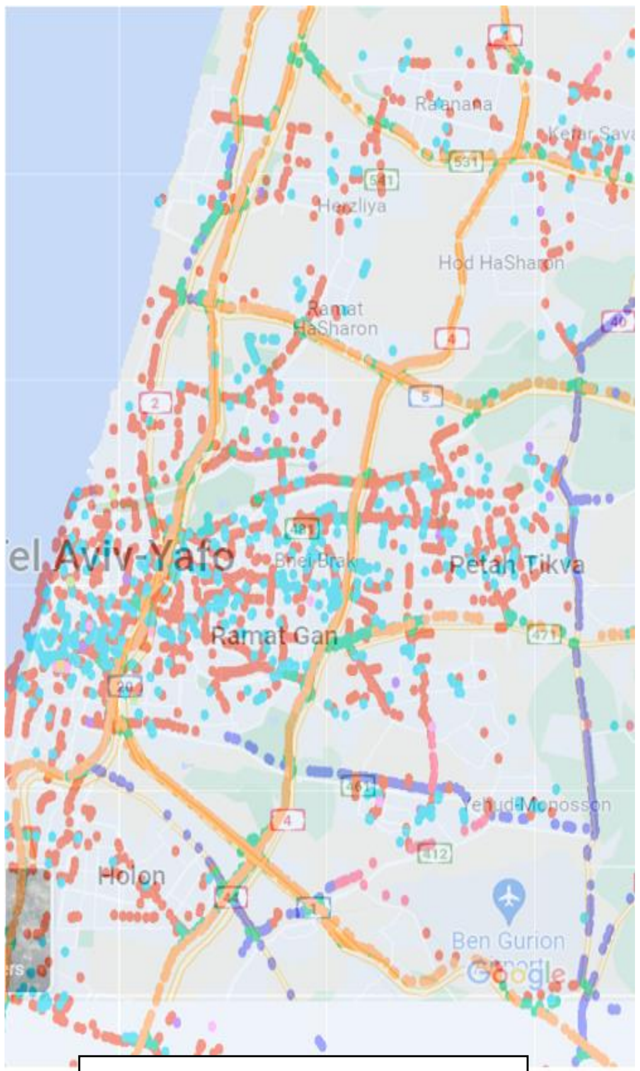

Figure 2: Events overlaying a map of Israel.

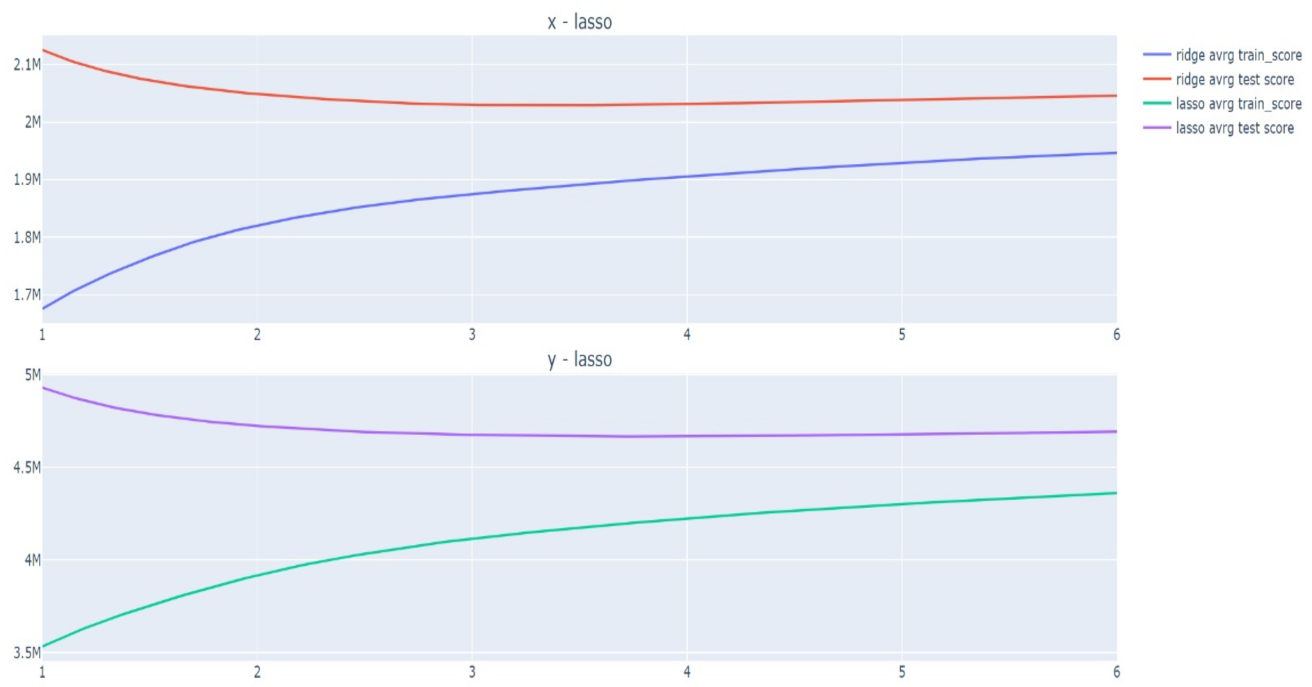

Figure 3: Identifying optimal regularization parameter for Lasso regression on the x-y coordinates.

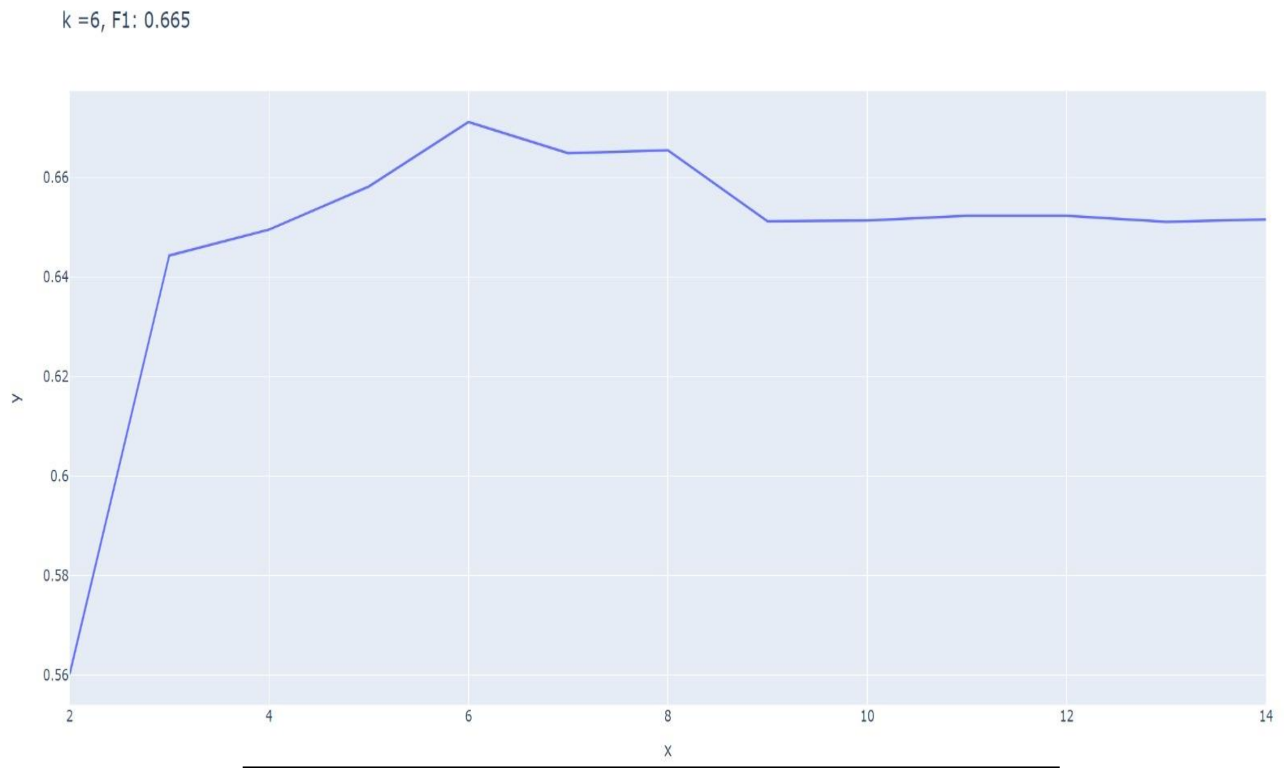

Figure 4: Finding the best k for the regression model.

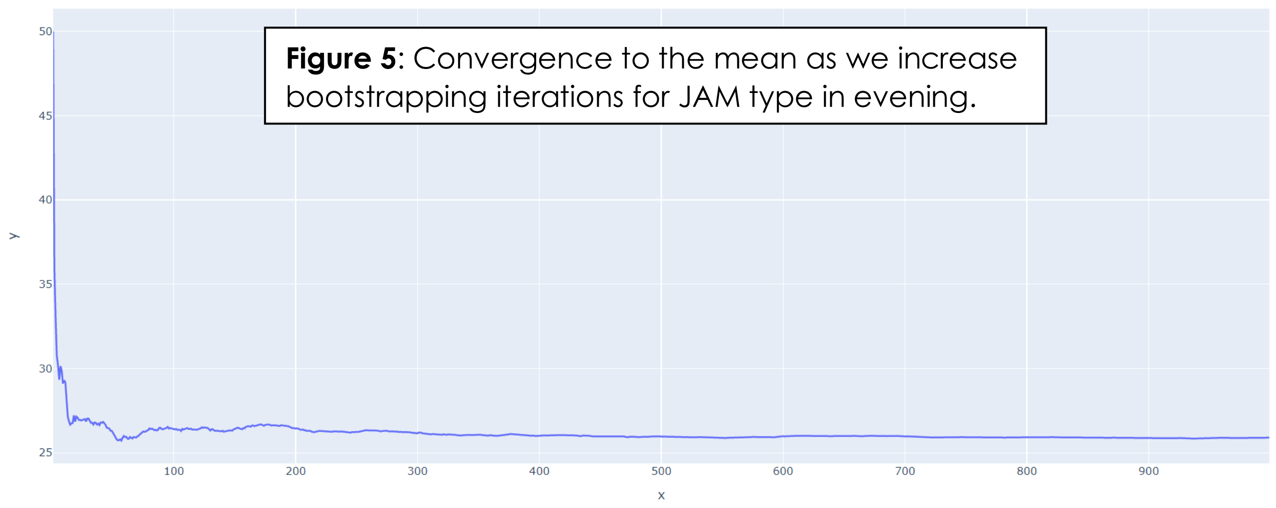

Figure 5: Convergence to the mean as we increase bootstrapping iterations for JAM type in evening.

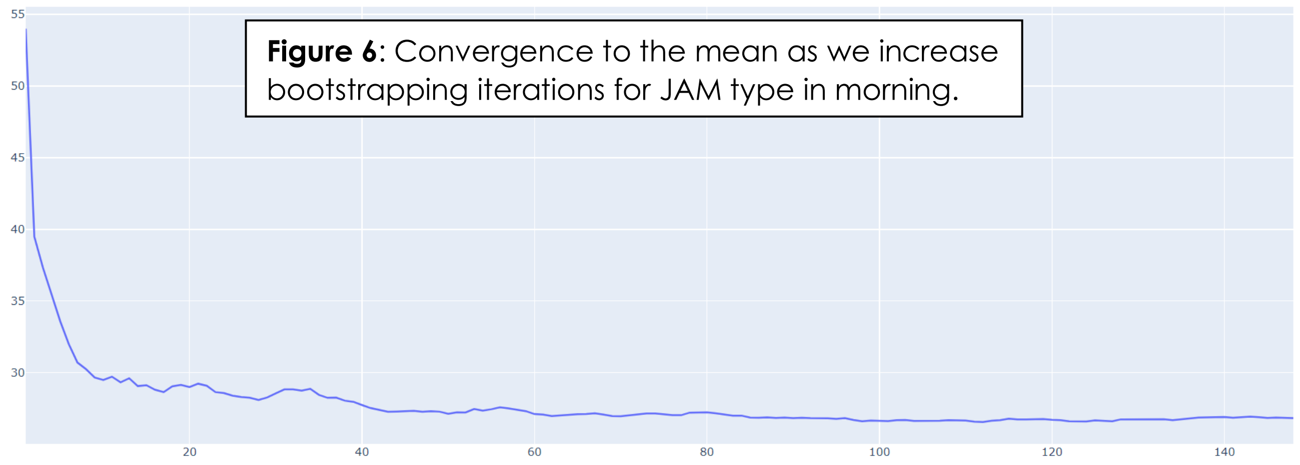

Figure 6: Convergence to the mean as we increase bootstrapping iterations for JAM type in morning.