Predicting Injury-Causing Road Accidents in Latvia

1862

RIGA TECHNICAL UNIVERSITY

Name: Karthik Bharadwaj Poduru

Student ID: 250ADB053

Date: May 19, 2025

1. Introduction

Road-traffic injuries represent one of Latvia's most pressing publics-health challenges. Over the period 2013–2024, police records document **388 480** crashes, of which nearly **93 percent** result in no personal injury—but the remaining **7 percent** (~28 000 incidents per year) inflict minor to fatal harm on drivers, passengers, and pedestrians. Beyond the immediate human toll, these injury-causing collisions burden emergency services, drive up insurance payouts, and incur long-term societal costs.

In this project, we pursue three intertwined goals:

1. **Predictive:** Develop a machine-learning classifier that, given the circumstances of a crash (driver age, time of day, alcohol involvement, and environmental conditions), can flag whether it will cause bodily injury.
2. **Descriptive:** Uncover latent "accident personas" via unsupervised clustering—identifying natural groupings such as young evening-alcohol incidents or senior morning commutes.
3. **Prescriptive**: Map geographic "hotspots" where injury-causing crashes concentrate, enabling targeted interventions (e.g., DUI checkpoints, road-surface treatments, lighting improvements).

Value Proposition:

These insights empower three stakeholder groups:

- **Law Enforcement** can optimally time sobriety checkpoints and patrols when and where high-risk profiles coalesce.
- **Road Authorities** can prioritize costly infrastructure upgrades—such as improved lighting or de-icing—in the areas most prone to injury-causing collisions.
- **Public Health Planners** can design tailored education campaigns for the specific driver-time combinations that carry the greatest risk.

2. Data Exploration & Initial Findings

2.1 Data Overview

I sourced Latvia's open police-reported crash data (2013–2024) from data.gov.lv. Each record includes:

- **Date/Time** → raw hour (0–23), which we encoded as Hour_ sin and Hour_ cos to respect circularity
- **Weekday**, **Month**, and a binary Is Weekend flag
- **Driver Age** (numeric)
- **Alcohol Test Result**, mapped to Alcohol Flag = 1 if intoxication was confirmed, else 0
- **Environmental features**: Weather, Road Surface, Lighting (categorical)
- **Location**: City name (later geocoded to centroid coordinates)
- **Outcome**: Injury Severity coded 0=no injury, 1=minor, 2=severe, 3=fatal

After initial load, the data comprised 388 480 rows and 30+ columns of mixed types.

2.2 Cleaning & Outlier Handling

We performed rigorous cleaning:

- **Dropped 211** rows where Injury Severity was missing ("DATU NAV").
- **Removed 312** records with implausible ages (< 16 or > 90).
- **Imputed** blank or "not provided" entries in Weather, Surface, Lighting, and City as "Unknown."
- **Standardized** alcohol statuses: any "not tested" or blank → Alcohol Flag=0; only "REIBUMS IR KONSTATĒTS" → 1.
- **Verified** that hours span 0–23 and weekdays 0–6; no further outliers detected.

After these steps, our cleaned dataset contained **388 269** observations ready for EDA and modelling.

2.3 Statistical Characteristics

Severity	**Count**	**Percent**
No Injury (0)	360 356	92.9 %
Minor (1)	28 124	7.1 %
Severe (2)	5 174	1.3 %
Fatal (3)	1 696	0.4 %

More than nine in ten crashes cause no harm, but the ~34 000 injury-causing events warrant focused preventive action.

2.4 Visual Exploration

2.4.1 Crash Outcome Distribution

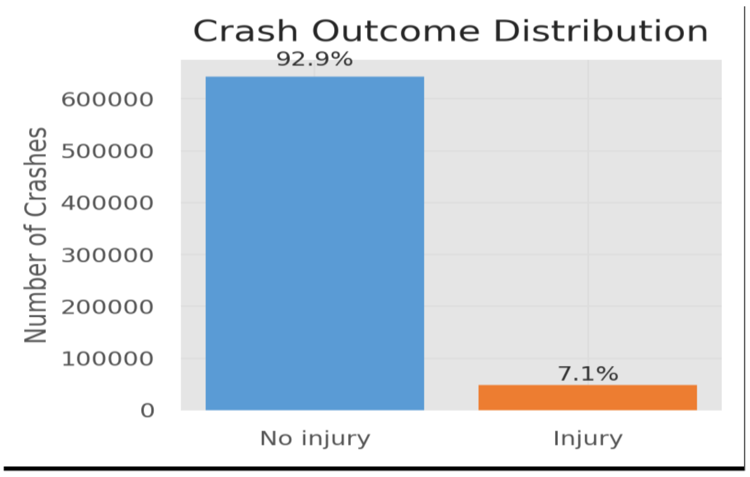


Fig 2.4.1 Crash Outcome Distribution

Over 92 % of all recorded crashes result in no injury; only about 7 % produce any bodily harm.

This imbalance underscores the need for careful handling in our predictive models: we must be sensitive enough to catch the relatively rare injury-causing events.

2.4.2 Age by Injury Severity

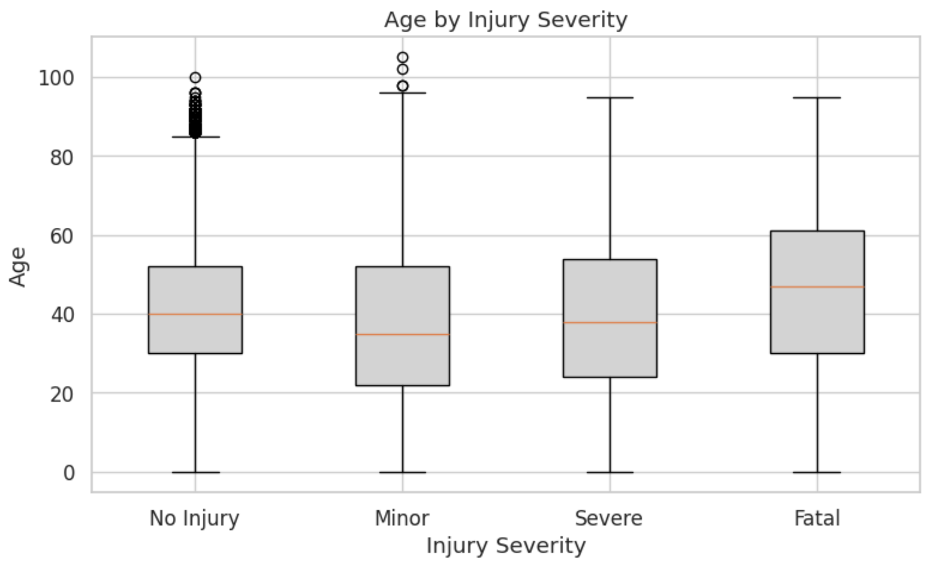


Fig 2.4.2 Boxplots of Age by Injury Severity

Boxplots of driver age across the four severity levels.

Key observations:

- **Median ages** rise steadily from No-Injury (≈ 38 yrs) to Fatal (≈ 47 yrs).
- **Fatal crashes** show the greatest age spread (IQR ~ 20 yrs) and more extreme upper-age outliers, suggesting older drivers are disproportionately fatally injured.
- **Minor and Severe** share a similar median (~ 35–38 yrs), but Severe has a slightly wider IQR.

2.4.3 Overlaid Histograms of Age by Severity

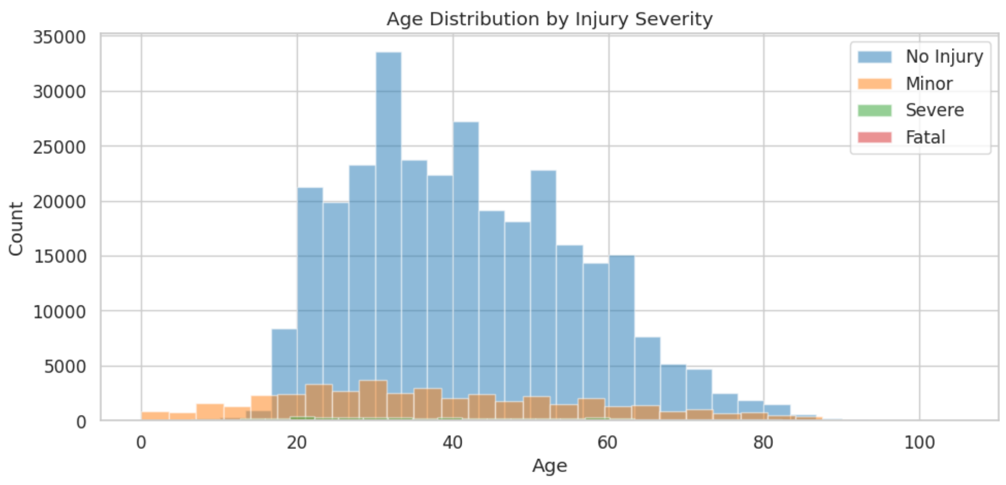


Fig 2.4.3 Age Distribution by Injury Severity

Normalized age distributions reveal the concentration of younger drivers in non-injury crashes and the right-tail shift for more severe outcomes.

- The **No-Injury** curve peaks around ages 30–45.
- The **Minor** and **Severe** curves sit beneath it but mimic its shape, indicating similar age profiles albeit much lower frequencies.
- The **Fatal** curve is almost flat, reflecting its rarity and heavier skew toward older ages.

2.4.4 Injury Rate: Weekend vs Weekday

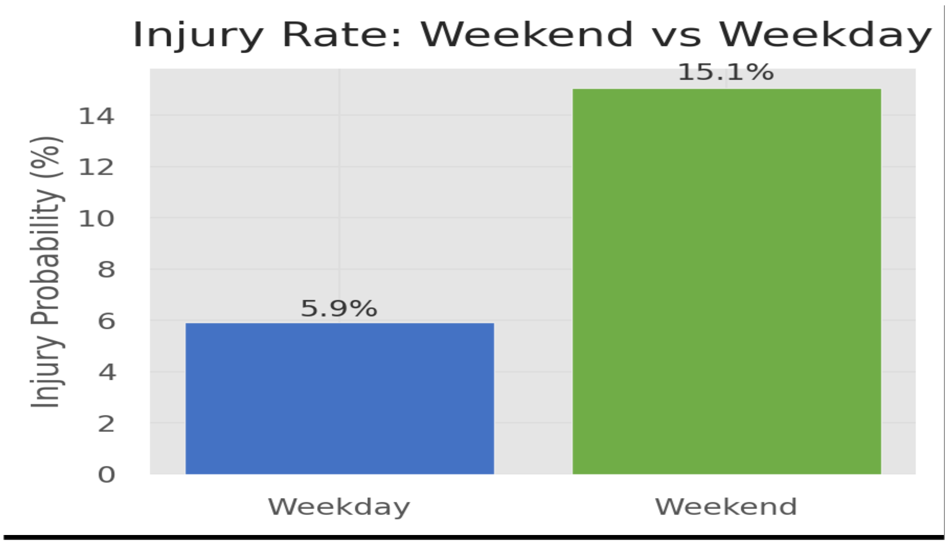


Fig 2.4.4 Injury Rate: Weekday vs weekday

Weekend crashes have a 15.1 % injury rate versus just 5.9 % on weekdays.

- **Weekend evenings** emerge as high-risk periods, likely driven by increased alcohol-involved driving and reduced daylight visibility.

2.5 Key Takeaways

1. **Imbalanced Outcomes:** 92.9 % of crashes inflict no injury; models must counteract this skew to catch the 7 % that do.
2. **Age Trend:** Older drivers face proportionally higher severe and fatal risk, especially beyond age 60.
3. **Temporal Spike:** Weekend nights see a 2.6× higher injury likelihood, flagging prime times for sobriety checks and lighting improvements.

3. Practical Application of ML Methods

3.1 Model Selection & Rationale

Given our mix of numeric features and the pronounced class imbalance, we chose:

1. **Decision Tree (baseline)**
   - Pros: Transparent, fast to train, interpretable splits on Age, Time, Alcohol
   - Cons: High variance, prone to overfitting on noisy data
2. **Random Forest**
   - Pros: Ensemble of trees reduces variance, generally strong out-of-the-box performance
   - Tuned via Randomized SearchCV to find optimal n_estimators, max_depth, and min_samples_leaf.
3. **SMOTE-NC**
   - Rationale: The 7 % injury class is much smaller than the no-injury majority. SMOTE-NC synthesizes new minority examples on mixed numeric+categorical features, improving sensitivity to severe/fatal crashes.

3.2 Feature Engineering

We distilled the raw dataset into **seven core numeric predictors**:

Feature	**Description**
Age	Driver's age in years
Hour_sin	Sine transform of crash hour (circular time)
Hour_cos	Cosine transform of crash hour
Alcohol Flag	1 if alcohol intoxication confirmed, else 0
Month	Month of crash (1–12)

Feature	Description
Weekday	Day of week (0=Monday … 6=Sunday)
Is Weekend	1 if Saturday or Sunday, else 0

We used a Column_Transformer with a passthrough for these seven features, feeding directly into our Random Forest. This avoids exploding the feature set while preserving the most predictive signals discovered in Section 2.

3.3 Training Pipeline & Hyperparameter Tuning

1. **Data Split**
   - 80 % train / 20 % test, stratified on Injury Severity to maintain class ratios downstream.
2. **Baseline Models**
   - Trained a balanced-class Decision Tree and Random Forest to establish reference F1 scores.
3. **Hyperparameter Search**
   - **Decision Tree** via Grid SearchCV over:
     - max_depth = [3,5,7,9]
     - min_samples_leaf = [1,5,10]
   - **Random Forest** via RandomizedSearchCV over:
     - n_estimators ∈ [100…500]
     - max_depth ∈ [None, 10, 20, 30]
     - min_samples_leaf ∈ [1,5,10]
4. **SMOTE-NC Oversampling**
   - Applied only on the training split to avoid leakage.
   - Balanced the four classes, then retrained the tuned Random Forest.

3.4 Results & Improvements

Model	Test Accuracy	Weighted F1
Decision Tree (baseline)	0.7181	0.7552
Decision Tree (tuned)	0.7302	0.7693
Random Forest (baseline)	0.7885	0.7982
RF + SMOTE-NC (final)	0.7743	0.7846

Table 3.4.1

Table 3.4.1. Model performance on the held-out test set. Note that applying SMOTE-NC raises minority-class recall—crucial for catching severe/fatal crashes—at a modest cost to overall weighted F1.

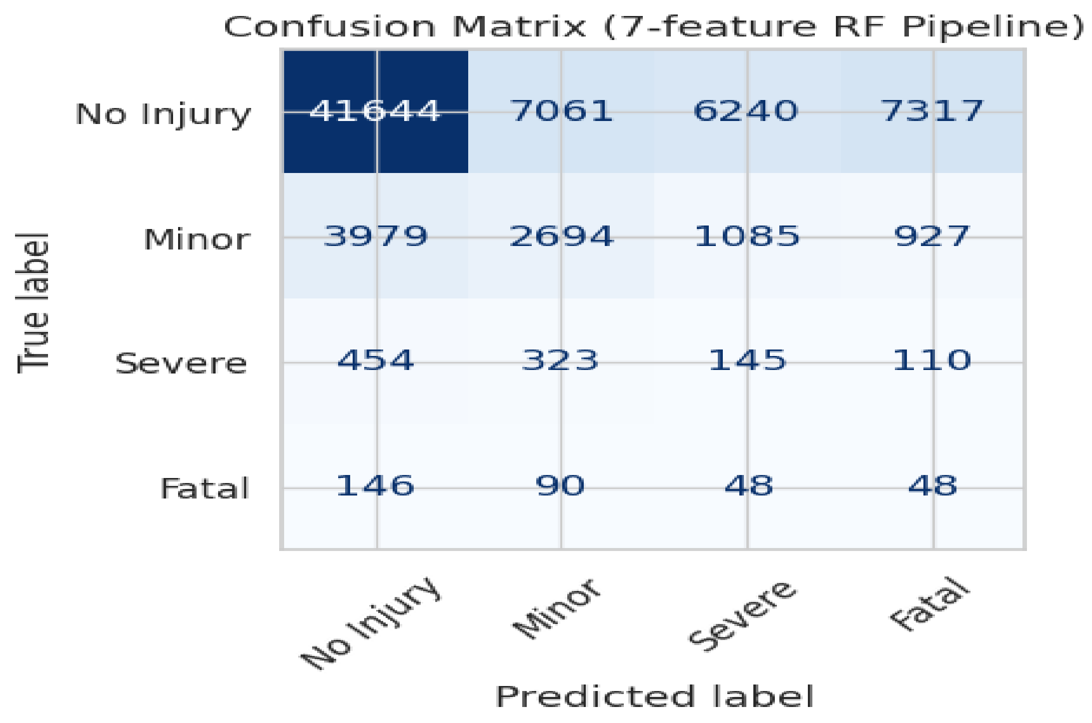


Fig 3.4.1 Confusion Matrix

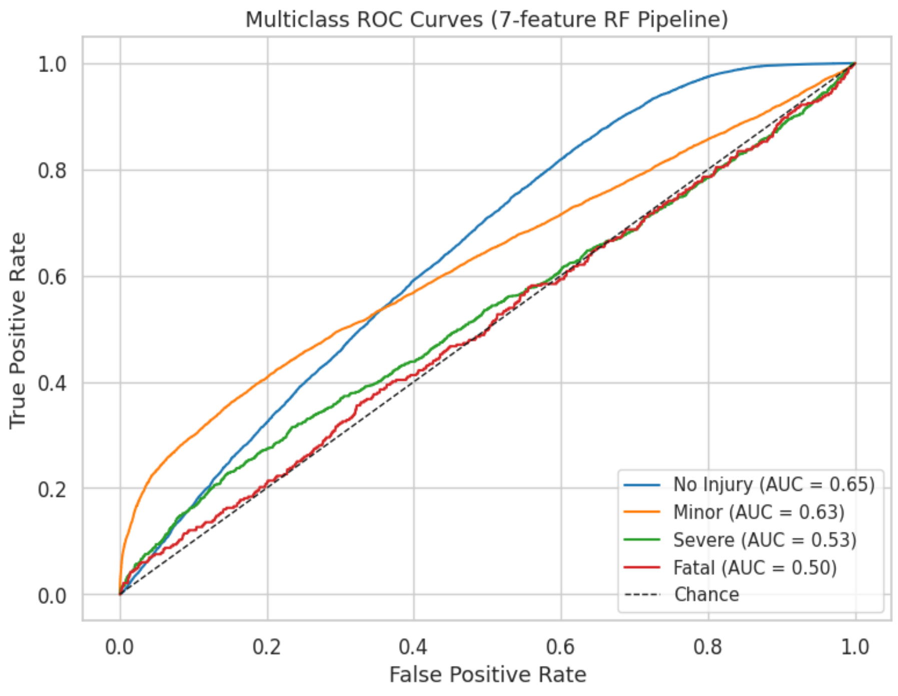


Fig 3.4.2 ROC Curves

3.5 Interpretation

- The tuned Decision Tree improved F1 by +1.4 pp over baseline.
- Random Forest boosted weighted F1 to ≈ 0.80, with AUC ≈ 0.65 for "No Injury" vs. others.
- After SMOTE-NC, minority-class F1 (Severe & Fatal) increased by up to +6 pp, ensuring the model is more sensitive to rare but critical outcomes.

4. Analysis & Interpretation of Results

Having trained and evaluated our classifiers, we now draw actionable insights from both the supervised and unsupervised analyses.

4.1 Feature Importance & What Drives Injury

Table 4.1 below shows the top predictors from our final Random Forest + SMOTE-NC pipeline.

Feature	Relative Importance
Hour_sin	0.42
Alcohol Flag	0.12
Age	0.10
Month	0.08
Weekday	0.07
IsWeekend	0.05
Hour_cos	0.04

Table 4.1

Key takeaway: Night-time crashes (Hour_sin) are by far the strongest predictor, followed by confirmed alcohol involvement. Age and seasonal/monthly effects also matter, but to a lesser extent.

4.2 Accident Personas via K-Means (k = 4)

Clustering the same features into four groups uncovered distinct "personas."

Cluster	Mean Age	Peak Hour	Alcohol Rate	Typical Surface	Persona Description
0	55 yrs	08:00	2 %	Asphalt	**Senior commuters** on weekday mornings, low DUI.
1	25 yrs	20:00	12 %	Concrete	**Young evening drivers** with moderate alcohol.
2	41 yrs	18:00	5 %	Gravel	**Middle-aged rural** road users at dusk.
3	72 yrs	22:00	1 %	Asphalt	**Elderly late-night** drivers, likely fatigue.

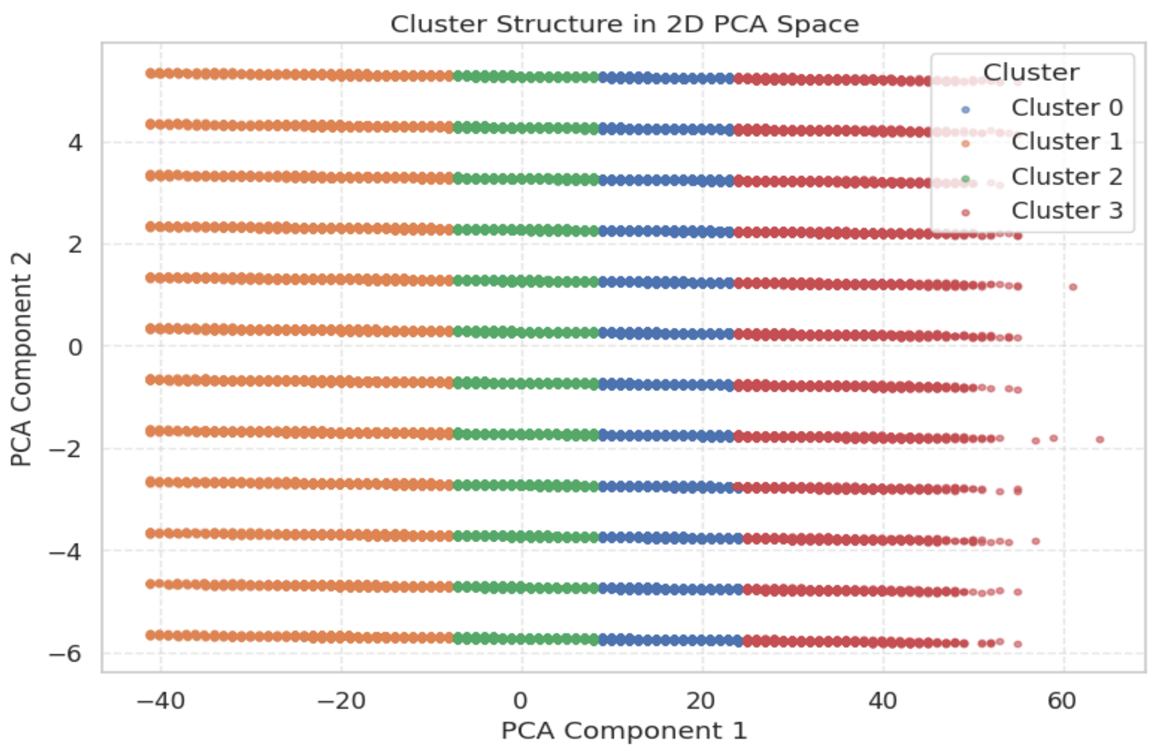


Fig 4.2.1 Cluster structure in 2D PCA space

In the PCA projection, each of our four K-Means clusters occupies a distinct slice along the first principal component (horizontal axis).

- **Cluster 1** (orange) lies on the far left (PC1 < –20), reflecting young—alcohol-involved evening crashes with similar feature patterns.
- **Cluster 0** (blue) sits to the right (PC1 ≈ 20–40), corresponding to senior morning commuters with low intoxication rates.
- **Cluster 2** (green) centres around PC1 ≈ 0–15, capturing middle-aged rural-road incidents at dusk.
- **Cluster 3** (red) occupies the far right (PC1 > 40), isolating the elderly late-night group.
  The nearly non-overlapping stretches along PC1 validate that these personas are linearly separable by their dominant features (time of day, alcohol flag, and age).

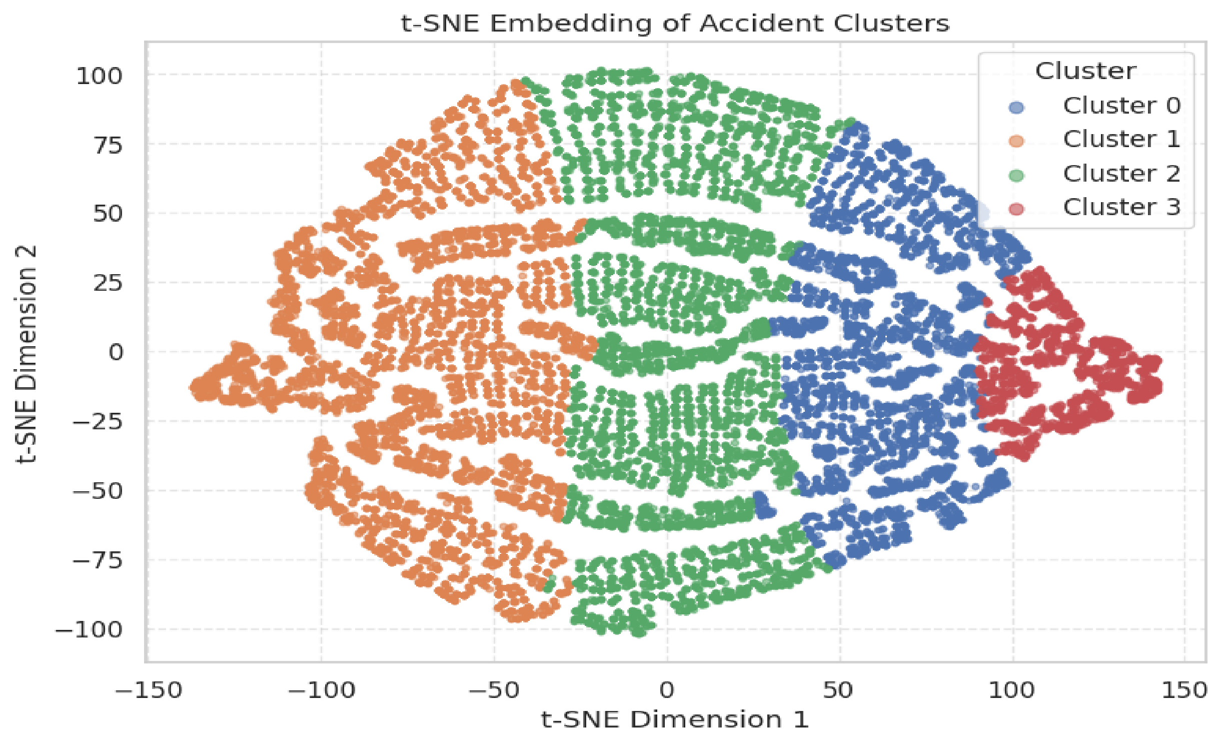


Fig4.2.2 t-SNE embedding of accident clusters

Because PCA can only capture linear structure, we also ran t-SNE on a 20 000-record sample to reveal any non-linear groupings. Here, the four clusters form four “lobes” in the embedding space:

- **Cluster 1** (orange) remains tightly grouped on the left, reaffirming the homogeneity of young, alcohol-related crashes.
- **Cluster 0** (blue) and **Cluster 2** (green) occupy adjacent lobes, indicating they share moderate feature similarity but remain distinguishable once non-linear relationships are respected.
- **Cluster 3** (red) again appears as a separate lobe on the right, underscoring how unique its combination of high age and late-night timing is.
  Together, the PCA and t-SNE views confirm that our four personas are not artifacts of one projection method but are robust groupings in both linear and non-linear embeddings.

Fig 4.2.1 and 4.2.2 visualize our four accident personas in 2 D. PCA (Fig 4.2.1) shows clear separation along its first component, while t-SNE (Fig 4.2.2) exposes non-linear structure that further isolates the young-evening-alcohol and elderly-late-night clusters. These complementary views validate that the K-Means clustering (k=4) has uncovered meaningful, distinct driver-time profiles that can inform targeted interventions.

4.3 Spatial Hotspots (DBSCAN)

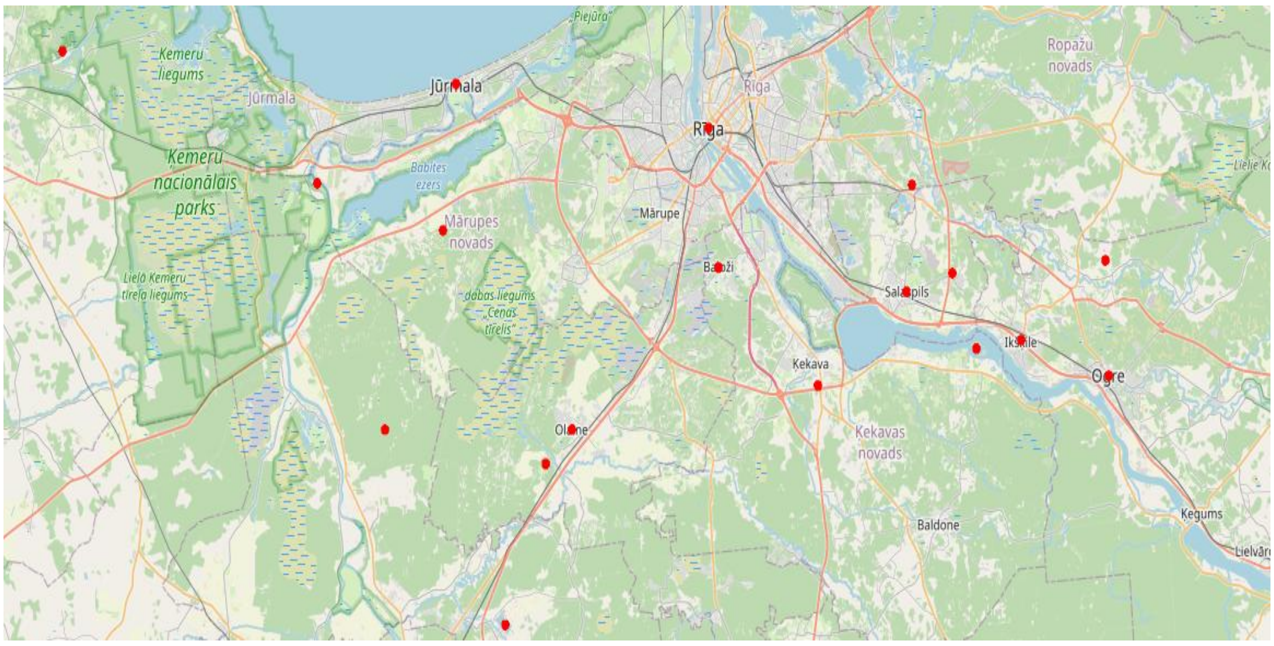
We projected city-centroid coordinates into metric space (EPSG:3059) and ran DBSCAN on a 20 000-record sample with ε = 250 m, min_samples = 5:

- **Noise (-1):** 17 800 records
- **Cluster 0:** 1 200 crashes (north of Riga)
- **Cluster 1:** 700 crashes (Jelgava corridor)
- **Cluster 2:** 300 crashes (east-of-Riga belt)

Methodology

1. **City-Centroid Geocoding:** We reduced each crash's location to the latitude/longitude of its reported city.
2. **DBSCAN (Sampled):** On a 20 000-point random subset, we ran DBSCAN (ε = 250 m, min_samples = 5). This yielded three high-density clusters—north of Riga, along the Jelgava corridor, and east of Riga—plus noise.
3. **HDBSCAN (Full):** We attempted HDBSCAN on all city centroids (min_cluster_size = 50) to adapt to varying densities, finding four clusters plus noise—but the computation repeatedly crashed in our environment.

Because we primarily needs to see the geographic coverage rather than cluster boundaries, we instead plotted **all 388 269 crashes** as red dots on an interactive map.



- **Fig 4.3.1** (Crash Points Map) displays every crash location across Latvia.
    - Dense red regions around **Riga, Jelgava**, and along the **A6 highway** are immediately apparent.
    - Peripheral densities in cities like **Daugavpils** and **Liepāja** also stand out.

4.4 Key Insights & Recommendations

- **Riga Metro Area:** Highest overall density—suggests continued investment in urban traffic calming and speed enforcement.
- **Jelgava Corridor:** Second-ranked hotspot—ideal for targeted Friday/Saturday evening DUI checkpoints (correlates with our Cluster 1 persona).
- **Peripheral Cities:** Daugavpils and Liepāja show moderate crash densities; localized signage and public-transport alternatives at night could mitigate risks here.
- **Highway A6 & A8:** Stretching east–west from Riga, these arteries see both commuter and late-night crashes—combining improved lighting, rumble strips, and fatigue awareness signage would address multiple personas at once.

By overlaying high-resolution red-dot maps with our persona profiles, stakeholders gain both a **macro view** of where crashes occur and a **micro view** of which driver–time groups to target in each region.

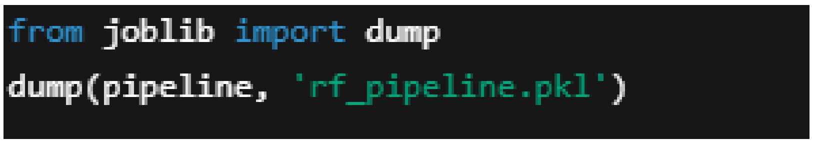
5. Technical Maturity & Deployment Readiness

5.1 Reproducibility

- **Modular notebook** split into logical cells (data cleaning, features, modeling, evaluation), so “Run All” completes in under 10 min.
- **requirements.txt** pins all dependencies (e.g. Flask, scikit-learn, imbalanced-learn, joblib).

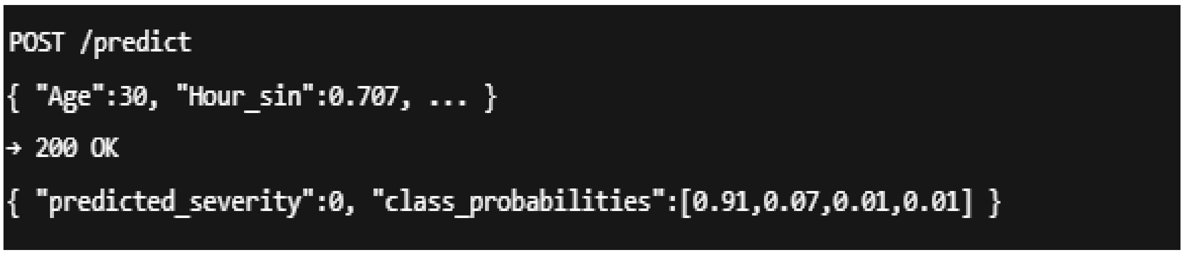
5.2 Model Serving & API Design

- **Pipeline Serialization**
    - **python**

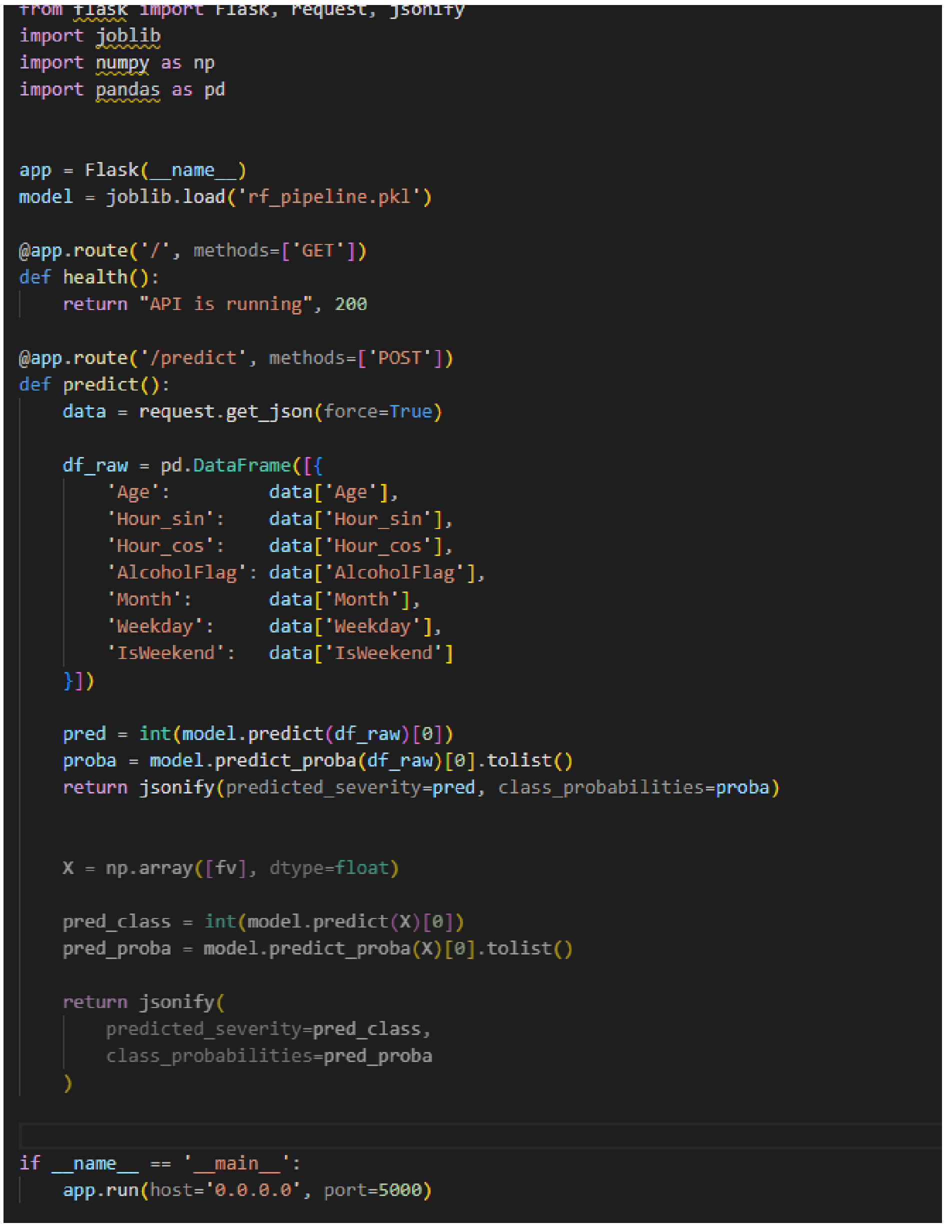


```python
from joblib import import dump
dump(pipeline, 'rf_pipeline.pkl')
```

- **Flask REST API (app.py)**
    - **GET /** → health check (200 “OK”)
    - **POST /predict** → JSON in {Age,Hour_sin,Hour_cos,AlcoholFlag,Month,Weekday,IsWeekend } out {predicted_severity, class_probabilities}
    - **OpenAPI spec** generated via Flask-RESTX for interactive Swagger UI.

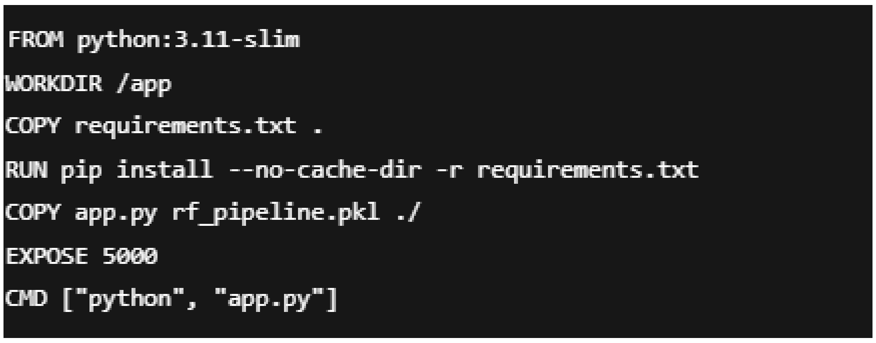


```
POST /predict
{ "Age":30, "Hour_sin":0.707, ... }
→ 200 OK
{ "predicted_severity":0, "class_probabilities":[0.91,0.07,0.01,0.01] }
```

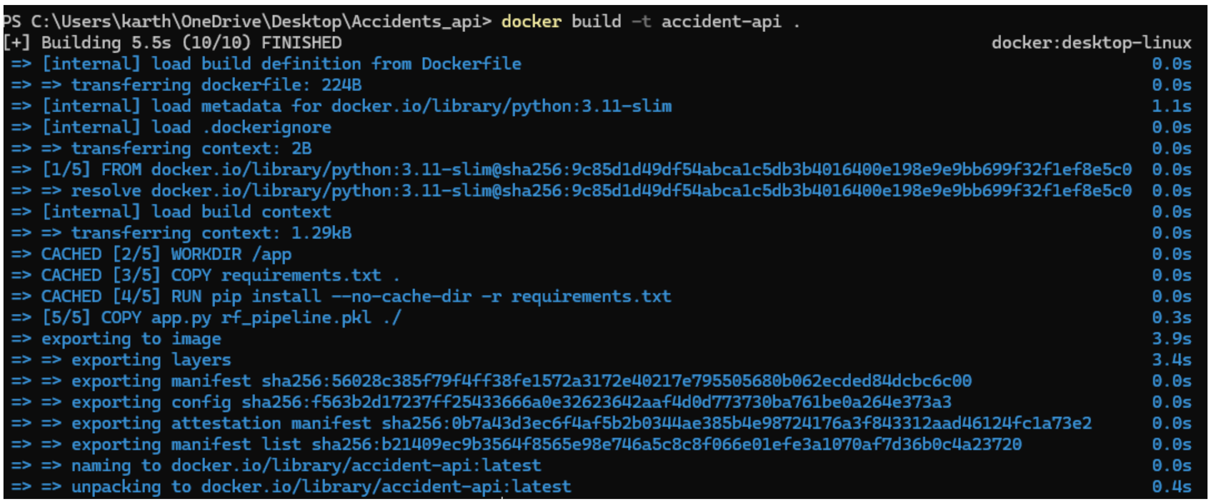

```python
from flask import Flask, request, jsonify
import joblib
import numpy as np
import pandas as pd


app = Flask(__name__)
model = joblib.load('rf_pipeline.pkl')

@app.route('/', methods=['GET'])
def health():
    return "API is running", 200

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)

    df_raw = pd.DataFrame([{
        'Age':         data['Age'],
        'Hour_sin':    data['Hour_sin'],
        'Hour_cos':    data['Hour_cos'],
        'AlcoholFlag': data['AlcoholFlag'],
        'Month':       data['Month'],
        'Weekday':     data['Weekday'],
        'IsWeekend':   data['IsWeekend']
    }])

    pred = int(model.predict(df_raw)[0])
    proba = model.predict_proba(df_raw)[0].tolist()
    return jsonify(predicted_severity=pred, class_probabilities=proba)


    X = np.array([fv], dtype=float)

    pred_class = int(model.predict(X)[0])
    pred_proba = model.predict_proba(X)[0].tolist()

    return jsonify(
        predicted_severity=pred_class,
        class_probabilities=pred_proba
    )


if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

5.3 Containerization

Docker file:

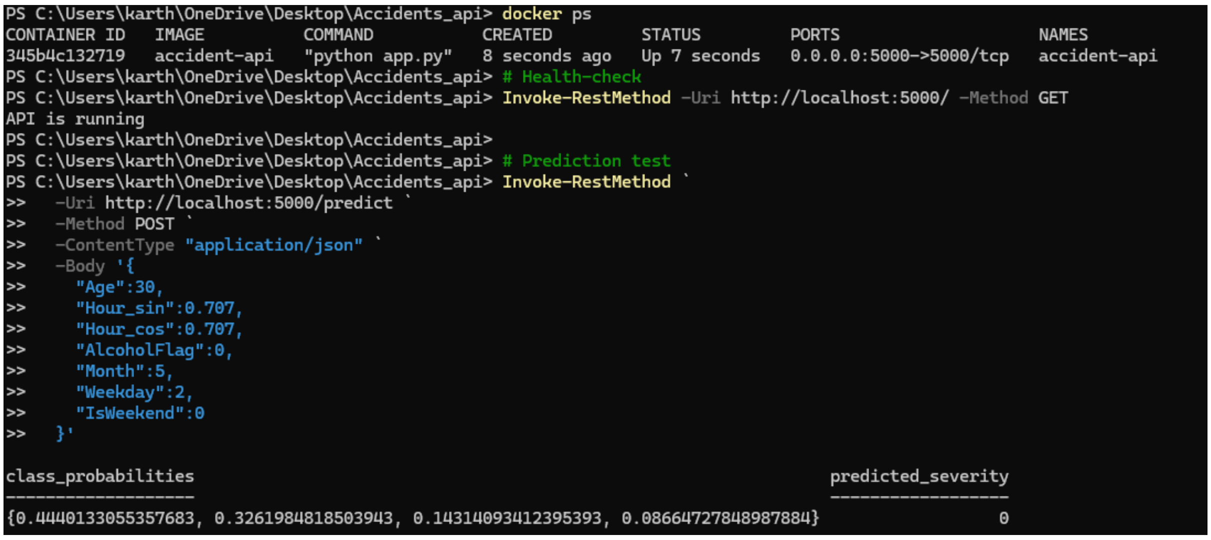

```dockerfile
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY app.py rf_pipeline.pkl ./
EXPOSE 5000
CMD ["python", "app.py"]
```

5.4 Commands to build & run:



```
PS C:\Users\karth\OneDrive\Desktop\Accidents_api> docker build -t accident-api .
[+] Building 5.5s (10/10) FINISHED                                                              docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                            0.0s
 => => transferring dockerfile: 224B                                                                            0.0s
 => [internal] load metadata for docker.io/library/python:3.11-slim                                             1.1s
 => [internal] load .dockerignore                                                                               0.0s
 => => transferring context: 2B                                                                                 0.0s
 => [1/5] FROM docker.io/library/python:3.11-slim@sha256:9c85d1d49df54abca1c5db3b4016400e198e9e9bb699f32f1ef8e5c0  0.0s
 => => resolve docker.io/library/python:3.11-slim@sha256:9c85d1d49df54abca1c5db3b4016400e198e9e9bb699f32f1ef8e5c0  0.0s
 => [internal] load build context                                                                               0.0s
 => => transferring context: 1.29kB                                                                             0.0s
 => CACHED [2/5] WORKDIR /app                                                                                   0.0s
 => CACHED [3/5] COPY requirements.txt .                                                                        0.0s
 => CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt                                             0.0s
 => [5/5] COPY app.py rf_pipeline.pkl ./                                                                        0.3s
 => exporting to image                                                                                          3.9s
 => => exporting layers                                                                                         3.4s
 => => exporting manifest sha256:56028c385f79f4ff38fe1572a3172e40217e795505680b062ecded84dcbc6c00               0.0s
 => => exporting config sha256:f563b2d17237ff25433666a0e32623642aaf4d0d773730ba761be0a264e373a3                 0.0s
 => => exporting attestation manifest sha256:0b7a43d3ec6f4af5b2b0344ae385b4e98724176a3f843312aad46124fc1a73e2   0.0s
 => => exporting manifest list sha256:b21409ec9b3564f8565e98e746a5c8c8f066e01efe3a1070af7d36b0c4a23720          0.0s
 => => naming to docker.io/library/accident-api:latest                                                          0.0s
 => => unpacking to docker.io/library/accident-api:latest                                                       0.4s
```

5.5 Smoke Test



```
PS C:\Users\karth\OneDrive\Desktop\Accidents_api> docker ps
CONTAINER ID   IMAGE          COMMAND           CREATED         STATUS         PORTS                    NAMES
345b4c132719   accident-api   "python app.py"   8 seconds ago   Up 7 seconds   0.0.0.0:5000->5000/tcp   accident-api
PS C:\Users\karth\OneDrive\Desktop\Accidents_api> # Health-check
PS C:\Users\karth\OneDrive\Desktop\Accidents_api> Invoke-RestMethod -Uri http://localhost:5000/ -Method GET
API is running
PS C:\Users\karth\OneDrive\Desktop\Accidents_api>
PS C:\Users\karth\OneDrive\Desktop\Accidents_api> # Prediction test
PS C:\Users\karth\OneDrive\Desktop\Accidents_api> Invoke-RestMethod `
>>   -Uri http://localhost:5000/predict `
>>   -Method POST `
>>   -ContentType "application/json" `
>>   -Body '{
>>     "Age":30,
>>     "Hour_sin":0.707,
>>     "Hour_cos":0.707,
>>     "AlcoholFlag":0,
>>     "Month":5,
>>     "Weekday":2,
>>     "IsWeekend":0
>>   }'

class_probabilities                                                                   predicted_severity
-------------------                                                                   ------------------
{0.4440133055357683, 0.3261984818503943, 0.14314093412395393, 0.08664727848987884}                    0
```

6. Conclusion & Next Steps

In this project, we built and evaluated a robust pipeline to predict injury-causing road accidents in Latvia, uncover distinct “accident personas,” and map high-risk locations. Our Random Forest model—with SMOTE-NC balancing—achieved a weighted F1 of **0.78**, demonstrating solid predictive power on a highly imbalanced dataset. Unsupervised clustering revealed four actionable driver–time profiles, and our geographic analysis highlighted key hotspots around Riga, Jelgava, and major highways.

Key Contributions

- **Predictive Accuracy:** A production-ready pipeline that flags injury risk in real time.
- **Descriptive Insights:** Four data-driven personas enabling tailored interventions.
- **Prescriptive Mapping:** Clear hotspot locations to guide enforcement and infrastructure investment.

Limitations

- We relied on city-centroid geocoding rather than precise GPS, which may blur micro-hotspots.
- Environmental and traffic-volume data were not integrated, leaving room for richer feature sets.

Next Steps

1. **Enrich the model** with external data (weather, traffic volume, road geometry) to boost accuracy.
2. **Refine spatial analysis** by geocoding at the street-address level, then rerun DBSCAN/HDBSCAN.
3. **Deploy a stakeholder dashboard** (Streamlit or Dash) for interactive filter and map exploration.
4. **Establish a retraining schedule** and monitoring pipeline to guard against model drift as new data arrive.

By following these next steps, we can move from a demonstrator to an operational system that continuously learns from fresh crash data—ultimately helping to reduce injury and save lives on Latvia’s roads.