

ACIDENTES EM SÃO PAULO: MAS, POR QUÉ?

Fatec-Cotia | CD | 4º semestre, Noturno

Inteligência Computacional | Prof. Meg

Trabalho A2, DEZ DE 2024

#ciencia_de_dados,
#inteligencia_computacional,
#estatistica, #ia, #ml, #k-means,
#cluster, #fatec_cotia,
#acidentes_transito, #transito_sp



GRUPO & ÍNDICE

- OBJETIVO E RELEVÂNCIA
- DATABASE E TOOLS
- ANÁLISE EXPLORATÓRIA
- CLUSTERIZAÇÃO
- RESULTADOS



GLAUCIA
FREITAS



RAFAEL
KOBAYASHI



ISABELA
CACIANO



ANDRÉ
KOGA



MARCOS
SOKABE

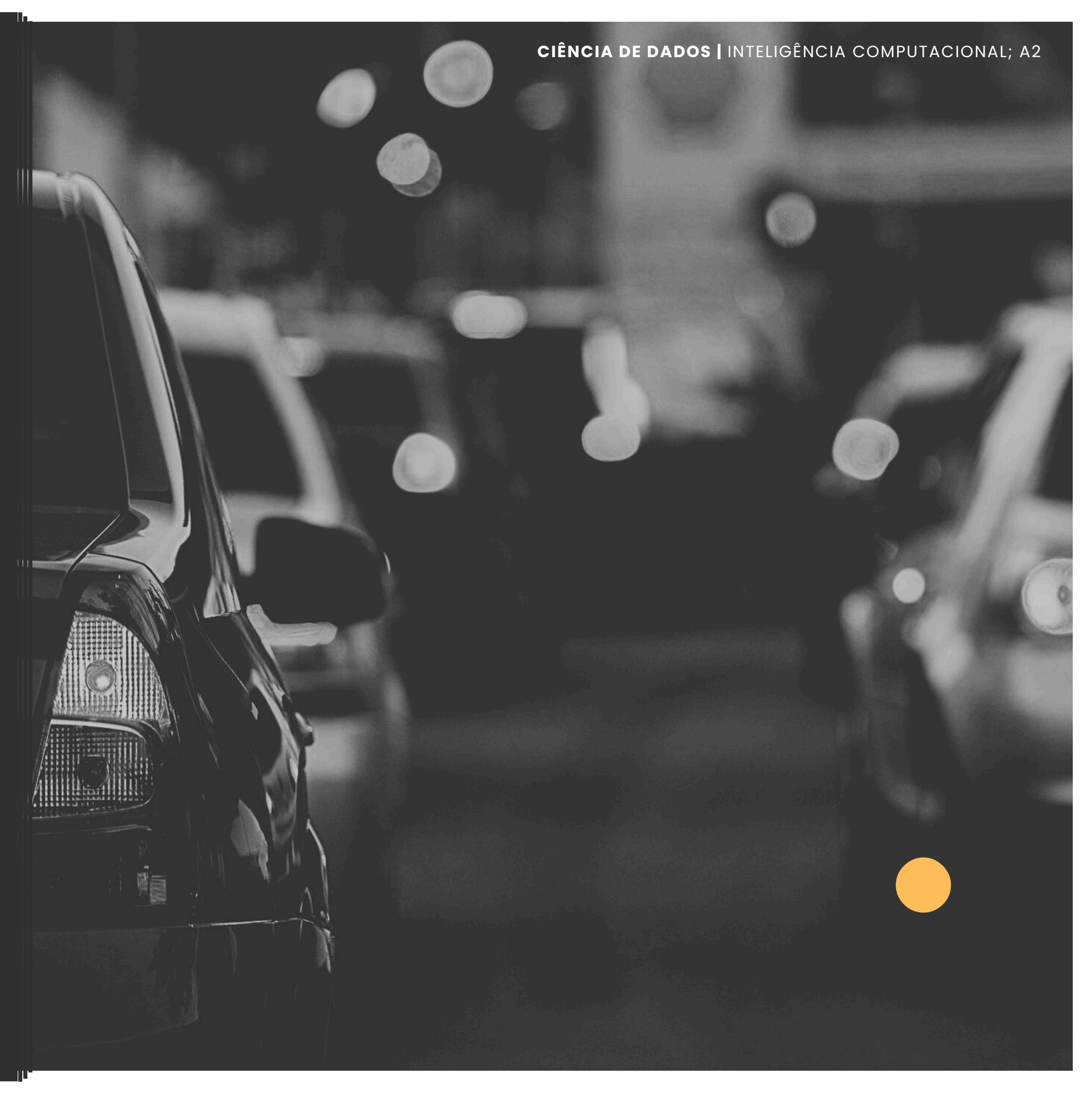


OBJETIVO E RELEVÂNCIA

O Brasil registrou mais de **1 milhão de acidentes de trânsito** em 2022, segundo dados consolidados pela Abramet

- { **34 mil** vítimas fatais
- Média de **92 mortes diárias**
- 1 óbito a cada 15 minutos**
- 1 ferido e sequelado a cada 2 minutos**
- São Paulo** registrou **3 mil mortes**

Este projeto visa entender os motivos de tantos acidentes de trânsito no Estado de São Paulo. Enxergar perfil, tendências e, desse modo, investigar as causas por trás do fenômeno. Consideramos o tema como de máximo interesse, pois toca em saúde e segurança públicas, assim como bem-estar social.



DATABASE: “ACIDENTES.CSV”

O dataset foi retirado do UC Irvine. Ele compila dados relevantes sobre acidentes de trânsito nas principais estradas brasileiras entre 2017 e 2023

{
27 colunas
422mil linhas
136mb
".csv"
UC Irvine

A base de dados foi recortada para contingenciar apenas os acidentes registrados nas principais estradas do **Estado de São Paulo**. Sua estrutura possui 27 variáveis relevantes para análise dos acidentes de trânsito, tais como: **causa**, **período do dia**, **condição climática**, **localização**, **número de feridos** e **fatalidades**, entre outros.



PYTHON LIBRARIES

A vasta oferta de bibliotecas do Python permitiu realizar a manipulação, visualização e análise de dados



pandas e **numpy**: manipulação de dados e arrays

plotly, seaborn, matplotlib: visualização de dados

sklearn: aprendizado de máquina (pré-processamento, seleção de variáveis, classificação e clustering).



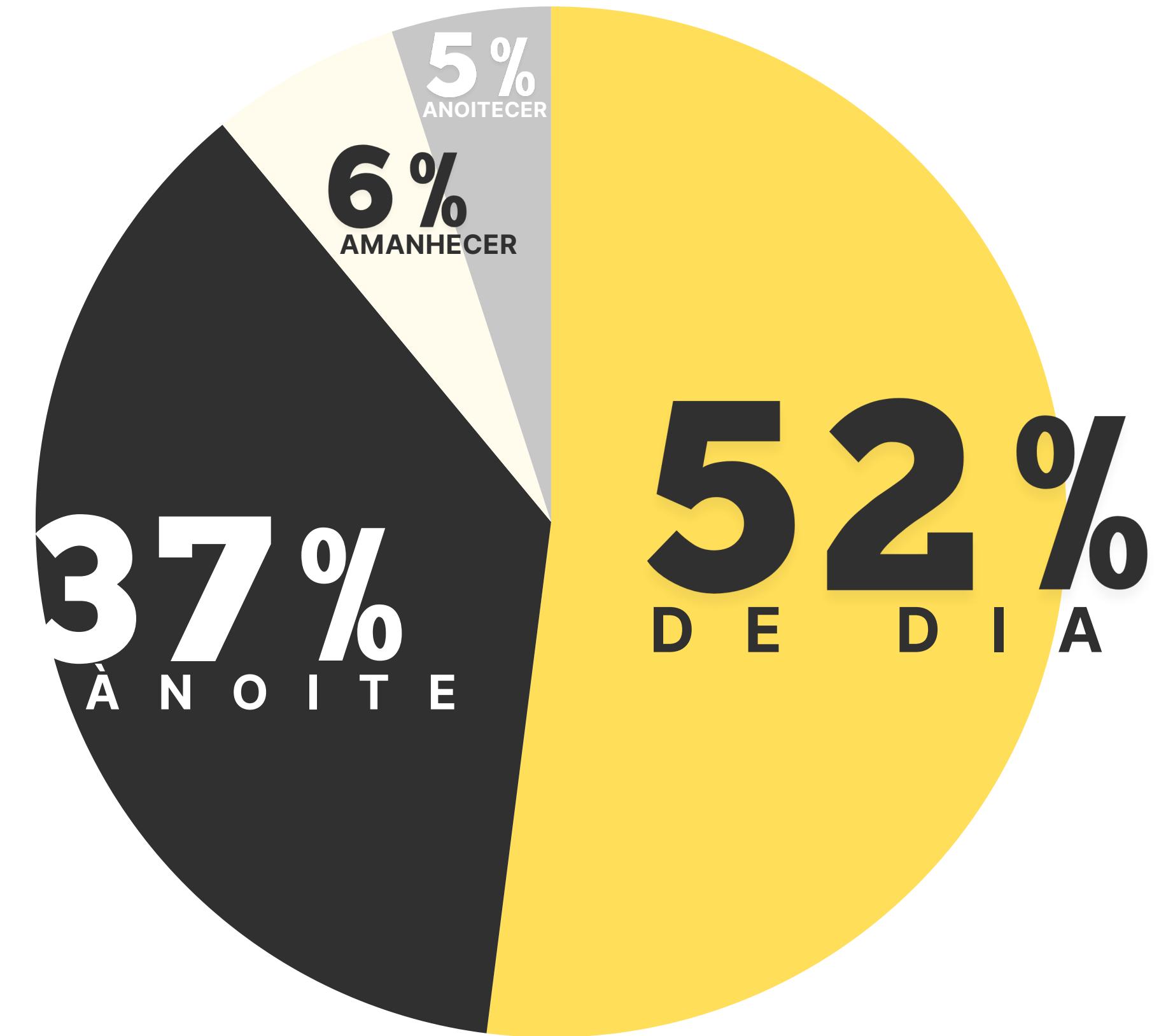
ANÁLISE EXPLORATÓRIA: MUNICÍPIOS X ACIDENTES

Guarulhos foi o município do Estado de São Paulo com maior número de acidentes. Em sequência, a capital paulistana, com 7% do total. Olhando mais profundamente, há um fenômeno interessante quando cruzamos com os dados de população absoluta: **a cidade de São Paulo possui uma população quase dez vezes maior que a população de Guarulhos.**



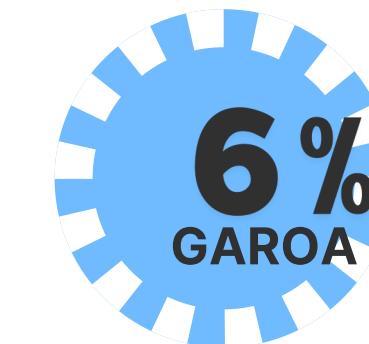
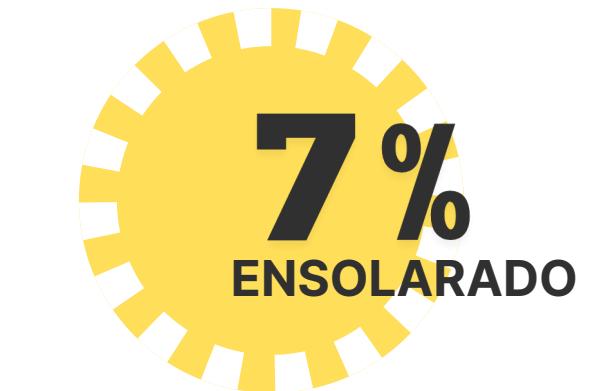
A N Á L I S E EXPLORATÓRIA: PERÍODOS DO DIA X ACIDENTES

Os acidentes no Estado de São Paulo ocorrem mais frequentemente durante o dia. À noite é o segundo maior período em que ocorrem acidentes de trânsito, com quase 40% do total. Amanhecer e anoitecer, entretanto, são os períodos com menor incidência de acidentes de trânsito.



A N Á L I S E EXPLORATÓRIA: CONDICÃO METEOROLÓGICA X ACIDENTES

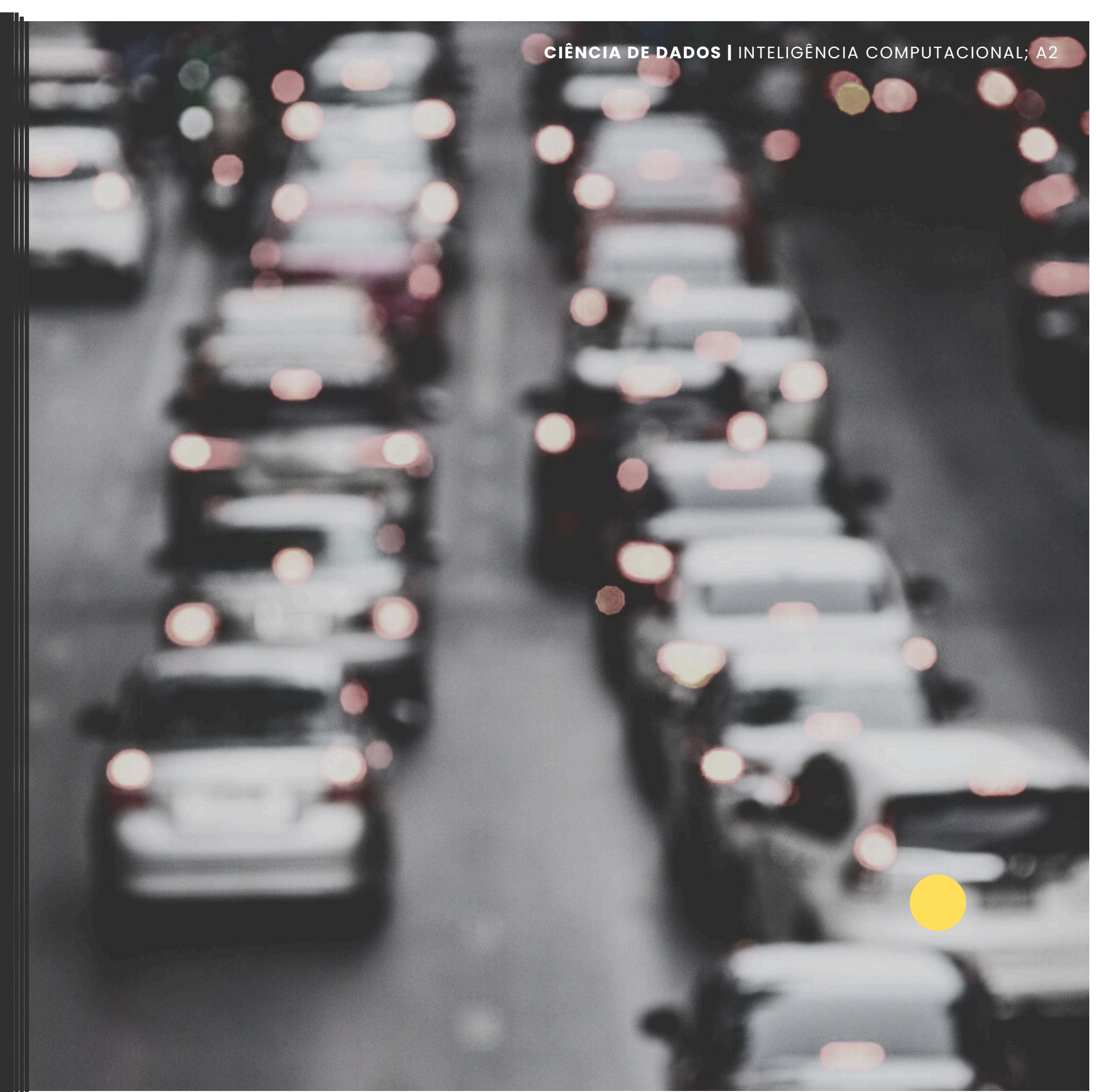
Parece pouco intuitivo, mas “Céu limpo” é a condição meteorológica em que ocorrem mais acidentes de trânsito. **Podemos supor que, ao contrário de dias chuvosos, há mais pessoas em circulação em dias com céu limpo, o que elevaria o número de acidentes.**



CLUSTERIZAÇÃO

A escolha de um algoritmo de clusterização para analisar dados sobre acidentes de trânsito no estado de São Paulo é motivada pela capacidade dessa técnica de identificar padrões e agrupar observações com características similares

- {
 - Identificação de padrões ocultos**
 - Segmentação espacial**
 - Exploração sem supervisão**
 - Medidas preventivas baseadas em evidências**



((((TARGET))))

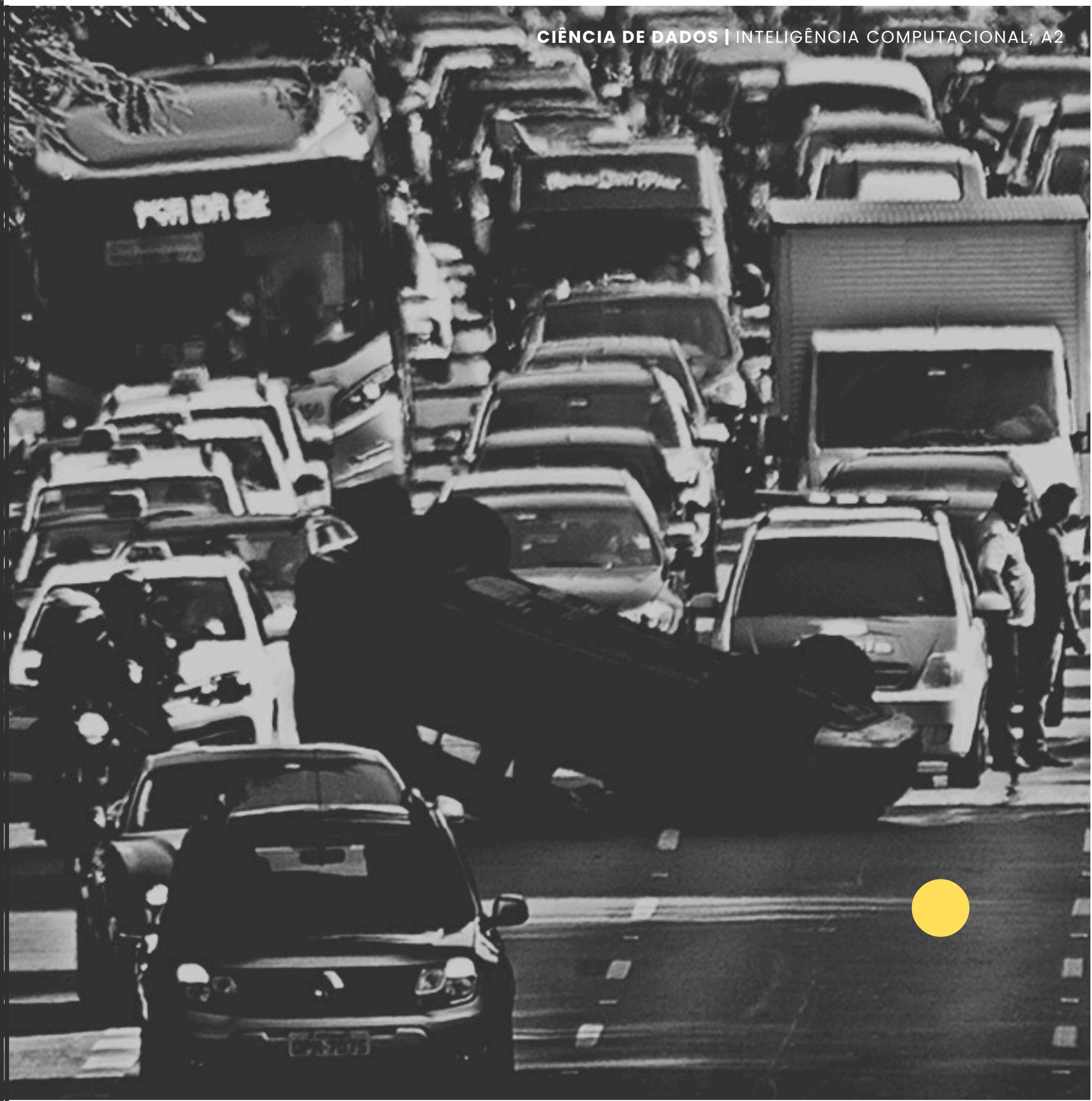
Escolhemos a variável tipo_acidente como target de nosso algoritmo de clusterização. Consideramos como uma variável relevante para definir agrupamentos

Análise do impacto do tipo de acidente: saber os fatores associados a diferentes tipos de acidentes (colisão frontal, atropelamento, tombamento, etc.) pode ajudar na formulação de medidas preventivas

Modelagem para tomada de decisão: Analisar o que influencia o tipo de acidente permite modelar o comportamento de variáveis. Condições climáticas, por exemplo: acidentes em dias de chuva podem diferir daqueles em dias secos.

Aplicação em Machine Learning

Compreensão dos fatores de risco: permite investigar como variáveis independentes (como hora, localização, ou condição da pista) influenciam os diferentes tipos de acidentes.



O QUE ENCONTRAMOS?

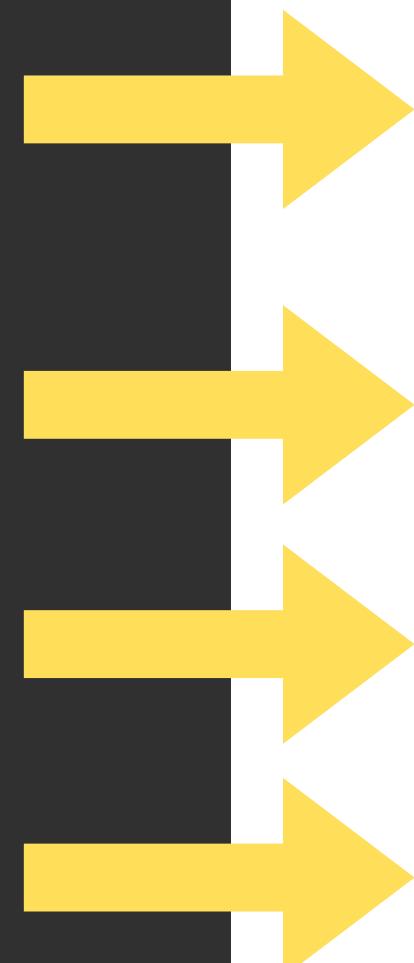
Em análise preliminar, o database continha:

990 valores null identificados nas colunas “br (estradas)” e “km (localização rodoviária)”

Sua alta dimensionalidade (27 variáveis distribuídas em colunas) prejudicava a performance do algoritmo de cluster

Algumas variáveis possuíam valores muito diversos e fragmentados

Dentro das variáveis de interesse, havia frequência desproporcional entre classes



O QUE FIZEMOS?

Então, procedemos com:

Limpeza dos valores nulls, excluindo as linhas transversalmente

Escolha de variáveis de interesse para a clusterização, conforme **teste de k-value**. Elas: **tipo_acidente, causa_acidente, tracado_via, pessoas, ilesos, veiculos, latitude e longitude**

Escolha de 3 classes dentro da variável tipo_acidente (colisao_traseira, tombamento e colisao_lateral)

Uso de **balanceamento** entre classes. Amostragem para diminuir a quantidade de clusters

PRÉ-PROCESSAMENTO I: LIMPEZA E FILTRAGEM

Saneamento de dados

Filtro: estado de São Paulo

Filtro: classes com n ocorrências

Filtro: classes específicas

```
6  df = df[df['tracado_via'] != 'Não Informado']
7  df = df[df['tipo_acidente'].notnull()]
8  df['br'].replace('NA', None)
9  df['km'].replace('NA', None)
10 df.dropna(inplace=True)
11 df = df.loc[df['uf'] == 'SP']
```

Trecho do código que **remove** linhas com **informações irrelevantes ou incompletas**, como: vias não informadas, tipos de acidentes nulos ou valores inválidos ('NA').

Filtrar os dados para manter apenas os acidentes ocorridos no **estado de São Paulo**.

```
14 df_filtered = df.groupby('tipo_acidente').filter(lambda x: len(x) > n)
15 df_filtered = df_filtered.loc[
16     (df_filtered['tipo_acidente'] == 'Colisão traseira') |
17     (df_filtered['tipo_acidente'] == 'Tombamento') |
18     (df_filtered['tipo_acidente'] == 'Colisão lateral')
19 ]
```

Mantém apenas as classes de acidentes com mais de **n ocorrências**.

Filtrar para focar em **três tipos** de acidentes específicos.

PRÉ-PROCESSAMENTO II: NORMALIZAÇÃO E BALANCEAMENTO

{

Identificação de colunas

Transformação de variáveis

Balanceamento de classes

```
22     colunas_categoricas = X.select_dtypes(include=['object', 'category']).columns  
23     colunas_numericas = X.select_dtypes(include=['int64', 'float64']).columns
```

Identificação de colunas **numéricas e categóricas**

Diferencia colunas categóricas (texto) e numéricas (valores).

```
28     preprocessor = ColumnTransformer(  
29         transformers=[  
30             ('cat', OneHotEncoder(sparse_output=False, handle_unknown='ignore'), colunas_categoricas),  
31             ('num', StandardScaler(), colunas_numericas)  
32         ])
```

Transforma variáveis categóricas em numéricas (**OneHotEncoder**) e padroniza variáveis numéricas (**StandardScaler**).

```
53     min_size = df_filtered['tipo_acidente'].value_counts().min()  
54     df_balanced = (  
55         df_filtered.groupby('tipo_acidente')  
56         .sample(n=min_size, random_state=42)  
57         .reset_index(drop=True)  
58     )
```

Balanceia as classes para que tenham o mesmo número de ocorrências, reduzindo viés nos modelos.

SELECT K-BEST, MRL, VALIDAÇÃO CRUZADA E RANDOM FOREST



Modelo de Regressão Logística

Random Forest

Select k-best

Matriz de confusão

```
60     pipeline = Pipeline(steps=[  
61         ('preprocessor', preprocessor),  
62         ('selector', SelectKBest(score_func=f_classif)),  
63         ('classifier', LogisticRegression())  
64     ])
```

MLS: Combina pré-processamento, seleção das melhores variáveis (SelectKBest) e o modelo de classificação (LogisticRegression).

```
66     scores = cross_val_score(pipeline, X_train, y_train, cv=5, scoring='accuracy')  
67  
68     pipeline = Pipeline(steps=[  
69         ('preprocessor', preprocessor),  
70         ('classifier', RandomForestClassifier(random_state=42, n_estimators=100))  
71     ])
```

Avalia o desempenho do modelo em diferentes conjuntos de dados para garantir generalização.
Utiliza um modelo de **Random Forest** para melhorar a robustez e explorar relações não lineares entre as variáveis.

```
73     conf_matrix = confusion_matrix(y_test, y_pred)  
74     sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
```

Avalia o desempenho do modelo, visualizando acertos e erros em uma matriz de confusão.

RESULTADOS & RESSALVAS I

Colisão Lateral: o modelo apresentou um desempenho moderado para colisões laterais, que são frequentes em regiões urbanas e interseções.

Colisão Traseira: essa classe teve um desempenho inferior, especialmente em recall. Isso pode indicar que colisões traseiras estão mais distribuídas geograficamente ou são confundidas com colisões laterais em pontos críticos.

Tombamento: o modelo detectou bem essa classe, possivelmente porque tombamentos estão mais associados a locais específicos (curvas acentuadas ou rodovias), onde as características dos dados são mais consistentes.

Relatório de Classificação:

	precision	recall	f1-score	support
Colisão lateral	0.66	0.77	0.71	519
Colisão traseira	0.71	0.56	0.63	485
Tombamento	0.90	0.92	0.91	479
accuracy			0.75	1483
macro avg	0.76	0.75	0.75	1483
weighted avg	0.75	0.75	0.75	1483

RESULTADOS & RESSALVAS II

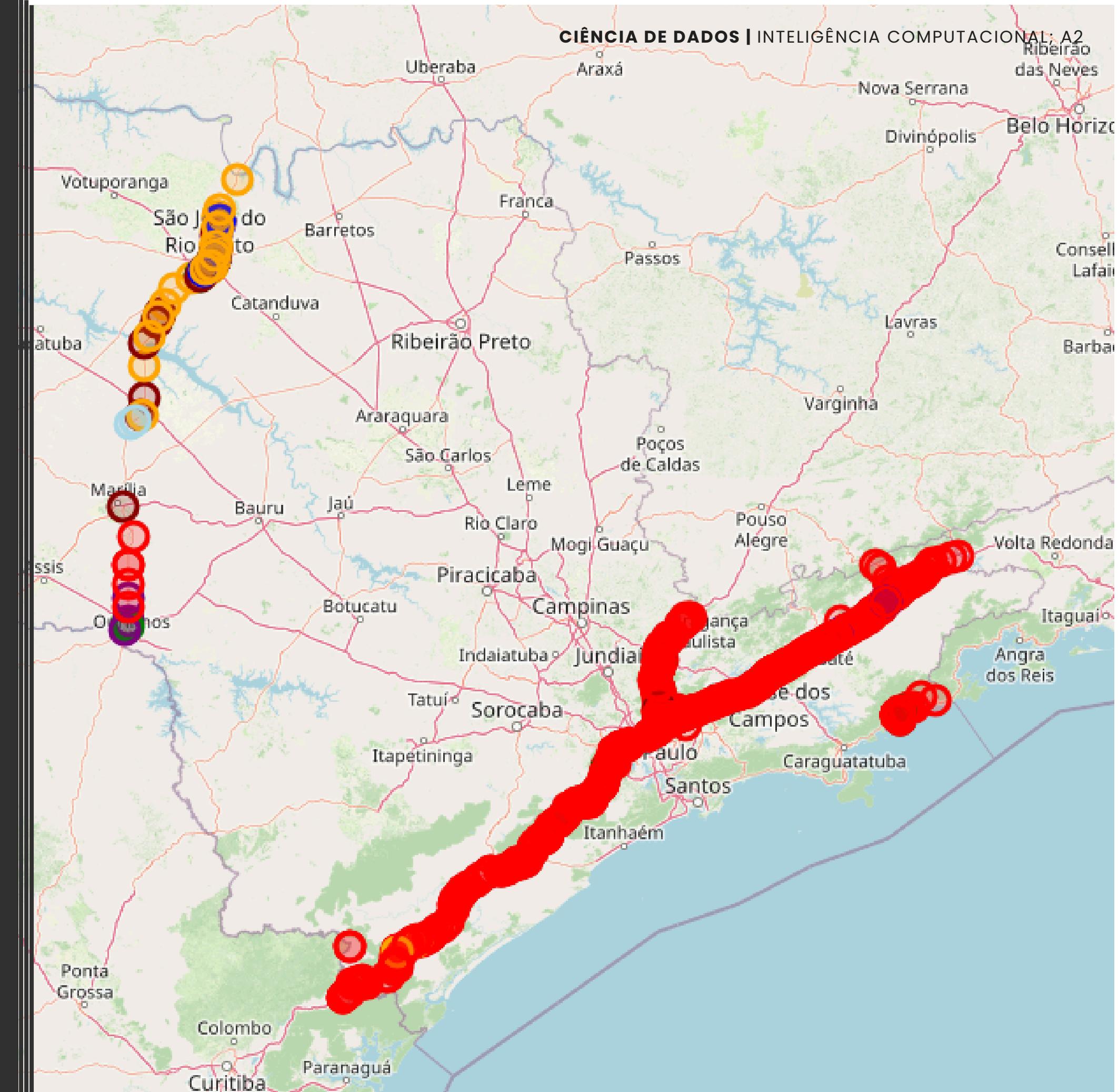
Há uma concentração de clusters ao longo de rodovias principais, como as que conectam o interior do estado à capital e ao litoral.

Grandes aglomerados aparecem em áreas urbanas e rodovias movimentadas, indicando possíveis zonas críticas para acidentes.

O litoral e as proximidades da capital destacam-se como zonas de alta densidade de acidentes, **possivelmente** devido ao tráfego intenso, fins de semana mais frequentes para viagens e condições climáticas convidativas para viagens longas (embora sejam apenas suposições).

O algoritmo gerou muitos clusters, o que indica grande variabilidade nos padrões de densidade dos dados.

Algumas áreas apresentam grupos mais dispersos, o que sugere que o DBSCAN interpretou ruídos ou pequenos agrupamentos como clusters independentes.



RESULTADOS & RESSALVAS III

Entendemos que o algoritmo de clusterização apresentou overfitting e confusão entre a classe de colisão lateral e colisão traseira.

Assumindo as ressalvas deste contexto, o modelo apresentou, por exemplo, o Cluster 0:

Causas associadas: majoritariamente relacionadas à má direção ou condução de, ao menos, uma parte

Envolvidos: majoritariamente 2 pessoas ou mais

Traçado da via: majoritariamente em retas



REFERÊNCIAS

{

<https://archive.ics.uci.edu/>

<https://teams.microsoft.com/v2/ Material de aula.>

