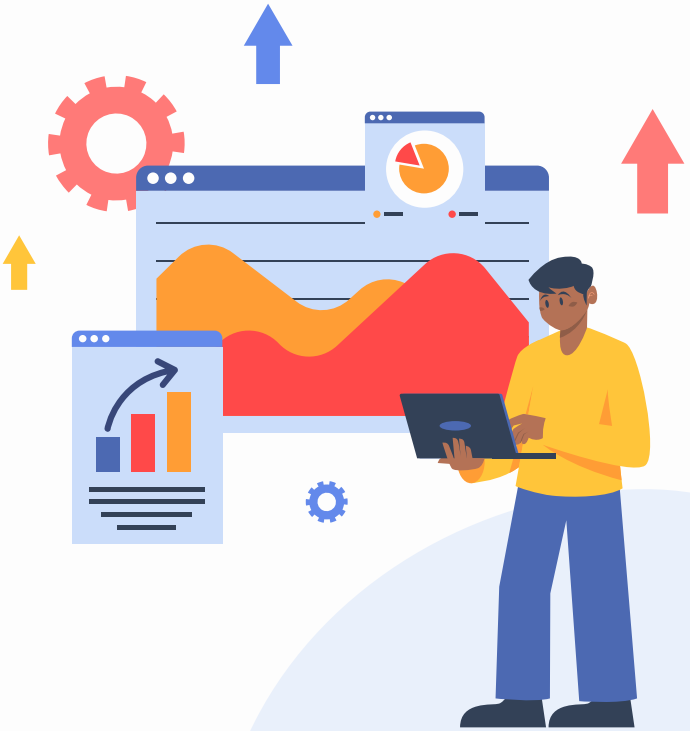
Universidade do Minho
Escola de Engenharia

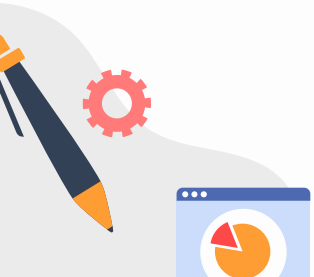
Dados e Aprendizagem Automática

Grupo 37



Beatriz Peixoto (pg59996)
Diogo Miranda (pg60001)
Martim Félix (pg58753)
Sandra Cerqueira (pg60016)

Indice



01 Metodologia

Fluxo de um processo de Aprendizagem Automática

- Definição do problema
- Exploração dos dados
- Tratamento dos dados
- Modelação
- Avaliação.

02 Exploração e Tratamento dos dados

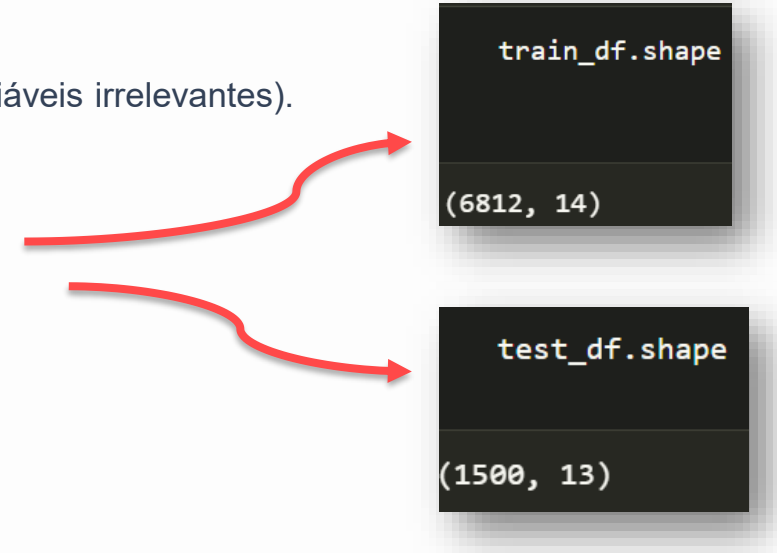
Exploração e Tratamento dos dados

Objetivos da EDA

- Compreender a estrutura dos dados.
- Identificar problemas (missing values, outliers, variáveis irrelevantes).
- Guiar decisões de limpeza e feature engineering.

Datasets

- **dataset de treino**: 14 atributos e 6812 linhas.
- **dataset de teste:** 13 atributos e 1500 linhas.
- Variáveis de tráfego, meteorologia e tempo



Estrutura do dataset

Atributo	Tipo	Descrição
city_name	Categórica/String	Nome da cidade onde os dados foram registados
record_date	Categórica	Data e hora do registo, ou seja, o timestamp do registo
average_speed_diff	Categórica	Diferença entre a velocidade em cenários sem trânsito e a velocidade que realmente se verifica. Valores mais altos indicam trânsito mais lento.
average_free_flow_speed	Numérica	Velocidade média máxima que os carros podem atingir em cenários sem trânsito.
average_time_diff	Numérica	Diferença média do tempo que se demora a percorrer um determinado conjunto de ruas em relação ao tempo de viagem sem trânsito. Valores altos indicam que se demora mais tempo.
average_free_flow_time	Numérica	Tempo médio que se demora a percorrer um determinado conjunto de ruas sem trânsito.
luminosity	Categórica	Nível de luminosidade que se verificava no Porto.
average_temperature	Numérica	Valor médio da temperatura na cidade do Porto para a data e hora registada.
average_atmosp_pressure	Numérica	Valor médio da pressão atmosférica, na cidade do Porto, para a data e hora registada.
average_humidity	Numérica	Valor médio da humidade, na cidade do Porto, para a data e hora registada.
average_wind_speed	Numérica	Valor médio da velocidade do vento, na cidade do Porto, para a data e hora registada.
average_cloudiness	Categórica	Valor médio da percentagem de nuvens, na cidade do Porto, para a data e hora registada.
average_precipitation	Numérica	Valor médio da precipitação, na cidade do Porto, para a data e hora registada.
average_rain	Categórica	Avaliação qualitativa do nível de precipitação para a data e hora registada, na cidade do Porto.

Informação Geral dos Dados

O que analisámos:

- Tipos de dados (train_df.info())
- Estatísticas descritivas (train_df.describe())
- Contagem de valores únicos → deteção de colunas sem variabilidade
- Primeiras observações (head)

Principais conclusões:

- Colunas city_name e AVERAGE_PRECIPITATION têm 1 único valor → removidas

```
unique_counts = train_df.nunique().sort_values()
print(unique_counts)
```

```
city_name                    1
AVERAGE_PRECIPITATION        1
LUMINOSITY                   3
AVERAGE_SPEED_DIFF           5
AVERAGE_CLOUDINESS           9
AVERAGE_RAIN                13
AVERAGE_WIND_SPEED          15
AVERAGE_TEMPERATURE         38
AVERAGE_ATMOSP_PRESSURE     43
AVERAGE_HUMIDITY            77
AVERAGE_FREE_FLOW_SPEED    225
AVERAGE_FREE_FLOW_TIME     442
AVERAGE_TIME_DIFF         1151
record_date               6812
dtype: int64
```

```
train_df.drop(columns=["city_name", "AVERAGE_PRECIPITATION"], inplace= True)
```

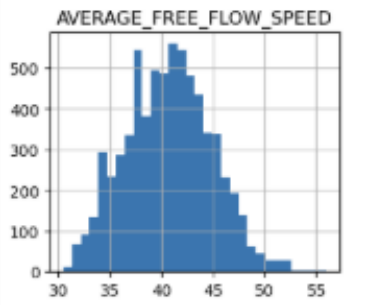
```
test_df.drop(columns=["city_name", "AVERAGE_PRECIPITATION"], inplace= True)
```

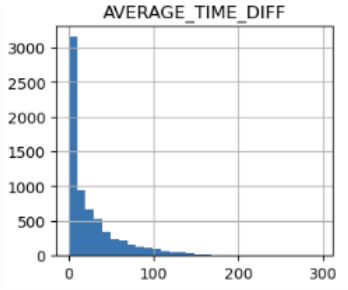
Distribuições

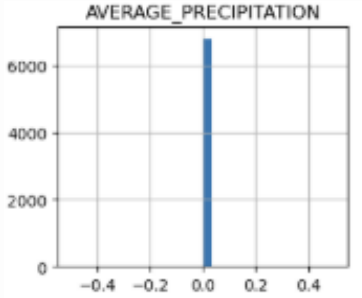
Histogramas de variáveis numéricas
Foram realizados histogramas para visualizar a distribuição de cada variável numérica para identificar padrões, a forma dos dados ou a necessidade de normalização.

Observaram-se:

- Variáveis com distribuições aproximadamente normais como AVERAGE_FREE_FLOW_SPEED, AVERAGE_TEMPERATURE, AVERAGE_ATMOSP_PRESSURE e AVERAGE_FREE_FLOW_TIME.
- Variáveis com distribuições assimétricas como AVERAGE_TIME_DIFF, AVERAGE_WIND_SPEED e AVERAGE_HUMIDITY.
- Variável com uma distribuição quase constante que é o caso da AVERAGE_PRECIPITATION que apresenta um único valor.



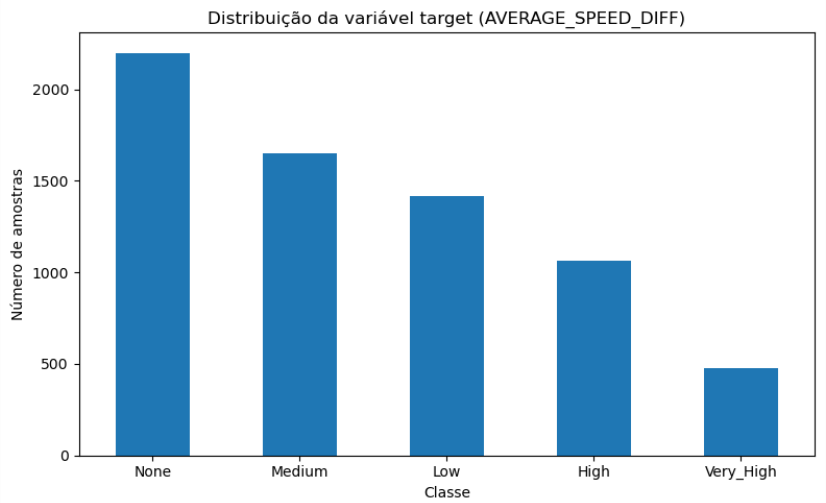




Distribuições

Distribuição da variável target (*AVERAGE_SPEED_DIFF*)

Através da análise do gráfico de barras, é visível que a classe *None* é a mais frequente, enquanto que a classe *Very_High* é claramente minoritária.
Este desbalanceamento poderá levar a alguns ajustes na forma como treinámos o modelo. Por exemplo, no ajuste de pesos.



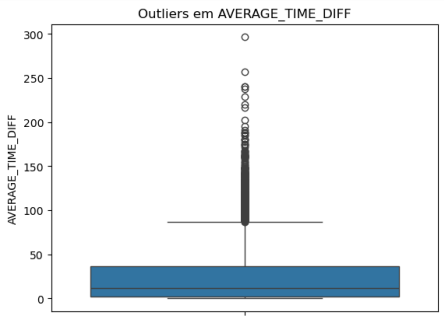
Outliers

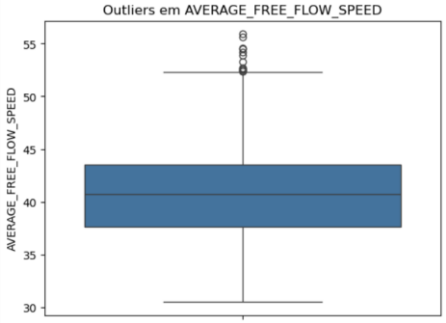
A análise de outliers foi feita através de boxplots para visualizar a existência destes valores.

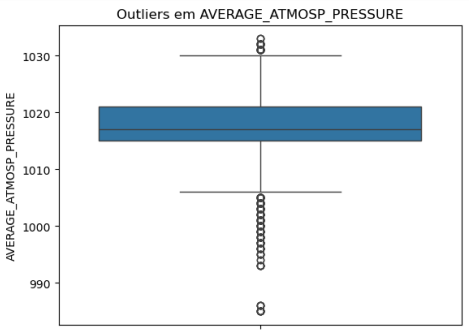
Não foi aplicado tratamento de outliers, porque:

- Os **modelos** utilizados **são robustos a valores atípicos**.
- Alguns destes valores podem representar situações reais de trânsito e a sua remoção ou transformação poderia afetar o desempenho do modelo.

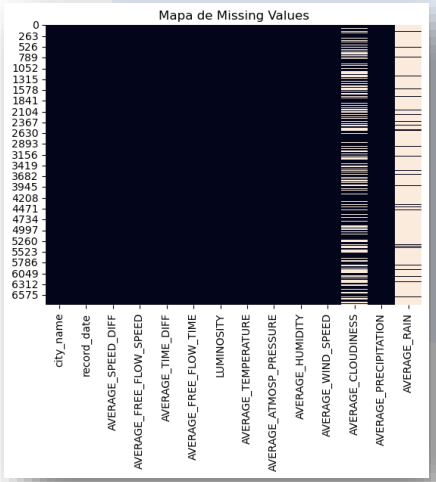
Contudo, futuramente, poderá ser analisado o impacto do tratamento de outliers no desempenho do modelo.







Missing Values



Ações realizadas

- Visualização da distribuição de valores ausentes com *heatmap*
- Cálculo da percentagem de *missing values* para AVERAGE_RAIN

Casos relevantes

- AVERAGE_RAIN: 91.7% de valores ausentes → variável crítica
- AVERAGE_CLOUDINESS: valores ausentes tratados como categoria 0

```
train_df["AVERAGE_CLOUDINESS"] = train_df["AVERAGE_CLOUDINESS"].fillna(0).astype(int)
```

Abordagens testadas para AVERAGE_RAIN

- Remoção da variável → descartada pela perda de informação contextual
- Interpretação dos ausentes como “sem chuva” → adotada

Decisão final

- AVERAGE_RAIN reconstruída como variável ordinal RAIN_INTENSITY (escala 0–4)

```
rain_map = {
    None: 0,
    "": 0,
    "chuvisco fraco": 1,
    "chuva fraca": 1,
    "aguaceiros fracos": 1,
    "chuvisco e chuva fraca": 1,

    "chuva leve": 2,
    "trovoada com chuva leve": 2,

    "chuva moderada": 3,
    "aguaceiros": 3,
    "chuva": 3,
    "trovoada con chuva": 3,
    "trovoada com chuva": 3,

    "chuva forte": 4,
    "chuva de intensidade pesado": 4,
    "chuva de intensidade pesada": 4,
}
def map_rain(x):
    if pd.isna(x):
        return 0
    return rain_map.get(x.strip().lower(), 3)

train_df["RAIN_INTENSITY"] = train_df["AVERAGE_RAIN"].apply(map_rain).astype("int8")
test_df["RAIN_INTENSITY"]  = test_df["AVERAGE_RAIN"].apply(map_rain).astype("int8")
```

Preparação da Variável Temporal

1. **Conversão para datetime:**
Transformação da coluna record_date para o tipo datetime.

2. **Extração de atributos temporais**:
Ano, mês, dia, hora, minuto, segundo e dia da semana.

3. **Análise de variabilidade:**
Verificou-se que minutos e segundos não apresentavam variação relevante.

4. **Remoção de atributos irrelevantes**:
Foram eliminadas as colunas record_date_minute, record_date_seconds e record_date.

```
train_df['record_date'] = pd.to_datetime(train_df['record_date'], format='%Y-%m-%d %H:%M:%S', errors='coerce')
```

```
train_df['record_date_year'] = train_df['record_date'].dt.year
train_df['record_date_month'] = train_df['record_date'].dt.month
train_df['record_date_day'] = train_df['record_date'].dt.day
train_df['record_date_hour'] = train_df['record_date'].dt.hour
train_df['record_date_minute'] = train_df['record_date'].dt.minute
train_df['record_date_seconds'] = train_df['record_date'].dt.second

# Day of week (0 = Monday, 6 = Sunday)
train_df['day_of_week'] = train_df['record_date'].dt.dayofweek
```

```
unique_counts = train_df.nunique().sort_values()
print(unique_counts)
```

```
record_date_minute          1
record_date_seconds         1
record_date_year            2
LUMINOSITY                  3
AVERAGE_SPEED_DIFF          5
RAIN_INTENSITY              5
day_of_week                 7
AVERAGE_CLOUDINESS          9
record_date_month          12
AVERAGE_WIND_SPEED         15
record_date_hour           24
record_date_day            31
AVERAGE_TEMPERATURE        38
AVERAGE_ATMOSP_PRESSURE    43
AVERAGE_HUMIDITY           77
AVERAGE_FREE_FLOW_SPEED   225
AVERAGE_FREE_FLOW_TIME    442
AVERAGE_TIME_DIFF        1151
record_date              6812
```

```
train_df.drop('record_date_minute', axis=True, inplace=True)
train_df.drop('record_date_seconds', axis=True, inplace=True)
train_df.drop('record_date', axis=True, inplace = True)
```

Encoding de Variáveis Categóricas

AVERAGE_CLOUDINESS

- Convertida para escala ordinal 0–4, baseada no nível de nebulosidade.

```
train_df['AVERAGE_CLOUDINESS'].value_counts(dropna=False)
```

```
AVERAGE_CLOUDINESS
NaN                 2682
céu claro           1582
céu pouco nublado    516
nuvens dispersas     459
nuvens quebrados     448
algumas nuvens       422
nuvens quebradas     416
céu limpo            153
tempo nublado         67
nublado               67
```

```
cloud_map = {
    "céu claro": 1,
    "céu limpo": 1,
    "céu pouco nublado": 2,
    "algumas nuvens": 2,
    "nuvens dispersas": 2,
    "nuvens quebrados": 3,
    "nuvens quebradas": 3,
    "tempo nublado": 4,
    "nublado": 4
}
```

LUMINOSITY

- Mapeada para 0 (DARK), 1 (LOW_LIGHT), 2 (LIGHT).

```
train_df['LUMINOSITY'].value_counts(dropna=False)
```

```
LUMINOSITY
LIGHT        3293
DARK         3253
LOW_LIGHT     266
Name: count, dtype: int64
```

```
luminosity_mapping = {
    'DARK': 0,
    'LOW_LIGHT': 1,
    'LIGHT': 2
}
```

AVERAGE_SPEED_DIFF (Target)

- Convertido para classes numéricas 0–4.

```
train_df['AVERAGE_SPEED_DIFF'].value_counts(dropna=False)
```

```
AVERAGE_SPEED_DIFF
None         2200
Medium       1651
Low          1419
High         1063
Very_High     479
```

```
target_mapping = {
    'None': 0,
    'Low': 1,
    'Medium': 2,
    'High': 3,
    'Very_High': 4
}
```

Feature Engineering - DEW POINT (Ponto de Orvalho)

1. **Ajuste dos Dados**

Humidade relativa truncada entre **1% e 100%**
Temperatura limitada ao intervalo **[-20°C, 45°C]**

2. Cálculo do Ponto de Orvalho (biblioteca **MetPy)**
Utilizando a função *dewpoint_from_relative_humidity* com base na temperatura em graus *Celsius* e na humidade relativa.
Nova *feature* criada*:* ***DEW_POINT***

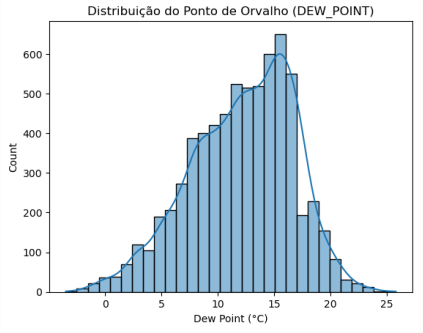
```
# 2) Dew Point com metpy
dp = dewpoint_from_relative_humidity(tC, rh).to("degC").magnitude
# 3) Reduzir ruído e feature mais estável
df["DEW_POINT"] = np.round(dp, 1)
```

3. Criação da versão discreta (*bins*)
Discretização do ponto de orvalho em 5 bins:
[-50, 0], [0, 5], [5, 10], [10, 15], [15+]

```
bins = [-50, 0, 5, 10, 15, 50]
labels = [0, 1, 2, 3, 4]

df["DEW_POINT_BIN"] = pd.cut(df["DEW_POINT"], bins=bins, labels=labels,
                             include_lowest=True).astype("int64")
```

Feature criada: ***DEW_POINT_BIN***



Feature Engineering – Hora de ponta

Passo 1 - Definimos as horas de ponta
[7,8,9,17,18,19,20]
Criação da variável *hora_de_ponta* (é binária e indica se a hora registada é ou não hora de ponta)

Passo 2 - Classificação em 3 níveis
- 0: Fora da hora de ponta
- 1: Hora de ponta menos acentuada
- 2: Hora de ponta mais acentuada

Permite distinguir diferentes níveis de congestionamento.
Criação da variável *tipo_hora_de_ponta* (ordinal e classifica a hora de ponta de acordo com a intensidade)

```python
def is_peak_hour(hora):
    peak_hours = [7,8,9,17,18,19,20]
    return 1 if hora in peak_hours else 0

train_df['hora_de_ponta'] = train_df['record_date_hour'].apply(is_peak_hour)
print(train_df['hora_de_ponta'].value_counts())
```

```python
peak_hours_1 = [7,17,20]

peak_hours_2 = [8,9,18,19]

def categorize_peak_hours(hour):
    if hour in peak_hours_2:
        return 2  # Hora de ponta mais acentuada
    elif hour in peak_hours_1:
        return 1  # Hora de ponta menos acentuada
    else:
        return 0  # Fora da hora de ponta

train_df['tipo_hora_de_ponta'] = train_df['record_date_hour'].apply(categorize_peak_hours)
print(train_df['tipo_hora_de_ponta'].value_counts())
```

Fonte: (https://www.metrodoporto.pt/uploads/document/file/399/3038_-_Estudo_de_Procura_Linha_Amarela_20170911.pdf)

Feature Engineering

Fim de semana

Criação da variável binária ***fim_de_semana*** para identificar os dias que são fim de semana.

```python
def e_fim_de_semana(row):
    weekday = int(row["day_of_week"])

    if weekday in (5,6):
            return 1
    return 0
```

Feriados

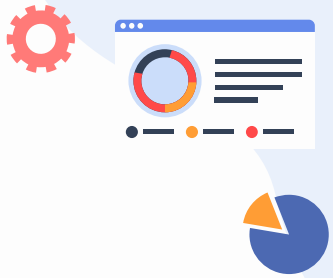
Criação da variável binária ***is_holiday*** para identificar feriados que podem afetar o fluxo de transito.

Lista de **feriados nacionais** e **municipais** (incluindo São João)

Atribuição:
1 → dia de feriado
0 → dia comum

```python
holidays_pt = pd.to_datetime([
    '2018-01-01', # Ano Novo
    '2018-03-30', # Sexta-feira Santa
    '2018-04-01', # Páscoa
    '2018-04-25', # Dia da Liberdade
    '2018-05-01', # Dia do Trabalhador
    '2018-05-31', # Corpo de Deus
    '2018-06-10', # Dia de Portugal
    '2018-06-24', # São João
    '2018-08-15', # Assunção de Nossa Senhora
    '2018-10-05', # Implantação da República
    '2018-11-01', # Dia de Todos os Santos
    '2018-12-01', # Restauração da Independência
    '2018-12-08', # Imaculada Conceição
    '2018-12-25', # Natal
    '2019-01-01', # Ano Novo
    '2019-04-19', # Sexta-feira Santa
    '2019-04-21', # Páscoa
    '2019-04-25', # Dia da Liberdade
    '2019-05-01', # Dia do Trabalhador
    '2019-06-10', # Dia de Portugal
    '2019-06-20', # Corpo de Deus
    '2019-06-24', # São João
    '2019-08-15', # Assunção de Nossa Senhora
    '2019-10-05', # Implantação da República
    '2019-11-01', # Dia de Todos os Santos
    '2019-12-01', # Restauração da Independência
    '2019-12-08', # Imaculada Conceição
    '2019-12-25', # Natal
    ])
```

Feature Engineering



Eventos no Porto

- Criação da variável binária *is_event_day* com base em datas reais de eventos relevantes (concertos, festivais, São João, Queima das Fitas, jogos no Dragão).

Afluência a Supermercados

- Nova variável *supermarket_peak_level* inferida a partir do dia da semana e hora:
- 0 → Baixa
- 1 → Moderada
- 2 → Forte

Período Escolar

- Criação da variável binária periodo_aulas com base no calendário letivo 2018/2019:
- 1 → Dentro do período escolar
- 0 → Fora do período escolar

Feature Engineering – Enconding Cíclico

1. Problema
Variáveis temporais apresentam natureza circular (ex.: hora 23 ≈ hora 0).
Valores numéricos simples (0–23) criam descontinuidades e prejudicam o modelo.

2. Transformação aplicada
Para preservar a ciclicidade, cada variável temporal foi convertida em duas dimensões trigonométricas:
sin(·) → posição angular
cos(·) → componente complementar

3. Variáveis transformadas
Hora → *hour_sin*, *hour_cos*
Dia da semana → *dow_sin*, *dow_cos*
Mês → *month_sin*, *month_cos*

4. Benefícios
Representação contínua e circular do tempo
Facilita a aprendizagem do modelo em padrões horários, semanais e sazonais
Evita saltos artificiais entre valores extremos

```python
def add_cyclical(df: pd.DataFrame) -> pd.DataFrame:
    # hour (0-23)
    df["hour_sin"] = np.sin(2 * np.pi * df["record_date_hour"] / 24)
    df["hour_cos"] = np.cos(2 * np.pi * df["record_date_hour"] / 24)

    # day of week (0-6)
    df["dow_sin"]  = np.sin(2 * np.pi * df["day_of_week"] / 7)
    df["dow_cos"]  = np.cos(2 * np.pi * df["day_of_week"] / 7)

    # month (1-12)
    df["month_sin"] = np.sin(2 * np.pi * df["record_date_month"] / 12)
    df["month_cos"] = np.cos(2 * np.pi * df["record_date_month"] / 12)

    return df
```

```python
train_df = add_cyclical(train_df)
test_df  = add_cyclical(test_df)
```

Feature Engineering

congestion_ratio = AVERAGE_TIME_DIFF / AVERAGE_FREE_FLOW_TIME

- Criada para capturar de forma mais clara o nível real de congestionamento.
  Esta razão permite comparar o tempo atual de deslocação com o tempo esperado em condições sem trânsito.

Interpretação:
- > 1 → aumento significativo do tempo de viagem → tráfego congestionado
- ≈ 1 → tempo normal → fluxo regular
- < 1 → tempo mais rápido que o esperado → leituras ruidosas ou condições atípicas

Justificação da criação:
- É uma métrica normalizada que resume o impacto do trânsito numa única variável, facilitando a aprendizagem do modelo e reduzindo variação causada por escalas diferentes.

03 Modelos e Resultados

Modelação e Avaliação dos Modelos

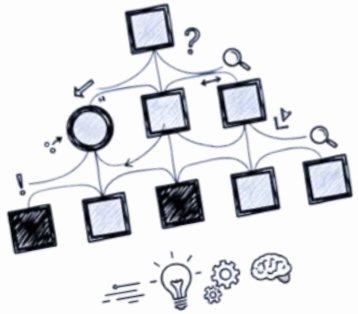
- Comparação de vários modelos para prever *AVERAGE_SPEED_DIFF* (5 classes).
- Métricas principais: *Accuracy* e *F1-Weighted*.

Nesta fase aplicámos diversos modelos e ensembles para comparar performance e escolher o melhor classificador. A métrica central foi o *F1-Weighted* devido ao desbalanceamento.

Modelos Base

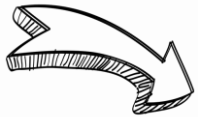
- Serviram para estabelecer referências iniciais.
- Performance baixa comparativamente a outros modelos.
- Úteis apenas para comparação.

Decision Tree



Hiperparâmetros

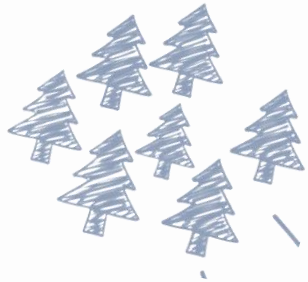
Resultado Grid Search
criterion: **entropy**
max_depth: **5**
min_samples_leaf: **10**
min_samples_split: **2**



Resultados Obtidos

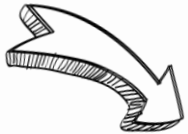
Accuracy Interna	***Accuracy no Kaggle publico***	**Accuracy no Kaggle Privado**
0.7660	0.82222	0.76666

Random Forest



Hiperparâmetros

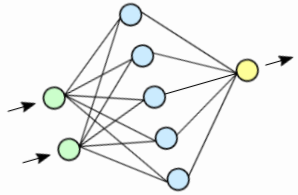
Resultado Grid Search
max_depth= **None**
max_features= **sqrt**
min_samples_leaf: **3**
min_samples_split: **2**
n_estimators: **300**



Resultados Obtidos

Accuracy Interna	***Accuracy no Kaggle publico***	**Accuracy no Kaggle Privado**
0.8020	0.80666	0.80190

Redes Neuronais



Arquitetura da Rede

```python
class MLP(nn.Module):
    def __init__(self, n_inputs):
        super().__init__()

        self.fc1 = nn.Linear(n_inputs, 64)
        self.drop1 = nn.Dropout(0.1)

        self.fc2 = nn.Linear(64, 32)
        self.drop2 = nn.Dropout(0.1)

        self.fc3 = nn.Linear(32, 5)  # 5 CLASSES

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = self.drop1(x)

        x = F.relu(self.fc2(x))
        x = self.drop2(x)

        return self.fc3(x)
```

- 2 camadas ocultas - uma com 64 neurónios e outra com 32
- 9 epochs

Resultados Obtidos

Accuracy Interna	*Accuracy no Kaggle Publico*	Accuracy no Kaggle Privado
0.7772	0.76000	0.76666

Ensemble Models – Comparação Geral

- Testámos ***Bagging***, ***Random Forest***, ***Decision Tree***, ***XGBoost***, ***Voting*** e ***Stacking***.
- Métrica usada na avaliação: F1-Score

Resultados Internos

RESULTADOS FINAIS:

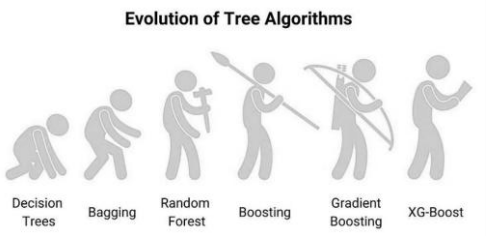
	Model	F1_mean	F1_std
5	Stacking	0.808568	0.011112
2	XGBoost	0.807250	0.009937
1	RandomForest	0.801091	0.008708
3	Bagging	0.794096	0.009363
4	Voting	0.788538	0.013841
0	DecisionTree	0.736431	0.012859

```python
stacking = StackingClassifier(
    estimators=[
        ("dt", dt),
        ("rf", rf),
        ("xgb", xgb)
    ],
    final_estimator=LogisticRegression(class_weight="balanced",  random_state=RSEED),
    cv=5,
    n_jobs=-1
)
```

Resultados Obtidos *Stacking*

F1-score	*Accuracy no Kaggle Publico*	Accuracy no Kaggle Privado
0.8086	0.82222	0.80000

XGBoost com Melhor Perfomance Interna



Hiperparâmetros

Resultado Grid Search
reg_alpha: **0**
n_estimators: **700**
min_child_weight: **3**
max_depth: **n**
learning_rate: **0.03**

Resultados Obtidos

F1-Weighted Interno	***Accuracy no Kaggle Publico***	**Accuracy no Kaggle Privado**
0.8107	0.82444	0.79714

- Métrica de avaliação: *F1 weighted*

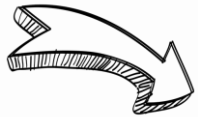
Apesar da melhor *accuracy* interna, este modelo não teve tão bom desempenho no Kaggle como o modelo fazendo tuning com RandomizedSearchCV.

XGBoost – Tuning com RandomizedSearchCV

- Pesquisa aleatória sobre 50 combinações.
- 5-fold CV para avaliação robusta.

Hiperparâmetros

Resultado Grid Search
reg_alpha: **0**
n_estimators: **700**
min_child_weight: **3**
max_depth: **3**
learning_rate: **0.03**



Resultados Obtidos

Accuracy Interna	*Accuracy no Kaggle Publico*	*Accuracy no Kaggle Privado*
0.815847	0.83777	0.81238

- Apesar do tuning, não superou o modelo anterior a nível interno.
- Mas superou muito no Kaggle público.
- Foi selecionado como modelo final

04 Resultados Competição

Resultados Competição

Public score

24	GRUPO_MECD_37		0.83777

Private score

Public Private

The private leaderboard is calculated with approximately 70% of the test data.
This competition has completed. This leaderboard reflects the final standings.

#	△	Team	Members	Score	Entries	Last	Solution
1	▲ 14	GRUPO_MIA_45		0.81809	43	7d	
2	▲ 15	GRUPO_MEI_11		0.81809	31	7d	
3	▲ 1	Grupo_MIA_15		0.81523	26	1mo	
4	▲ 1	GRUPO_MIA_34		0.81333	68	7d	
5	▲ 19	GRUPO_MECD_37		0.81238	39	7d	

Webgrafia

https://scikit-learn.org/stable/supervised_learning.html
https://www.metrodoporto.pt/uploads/document/file/399/3038__Estudo_de_Procura_Linha_Amarela_20170911.pdf
https://fne.pt/pt/noticias/go/acontece-calendario-escolar-2018-2019
https://www.sibsanalytics.com/wp-content/uploads/2024/05/5-anos-SIBS-Analytics-PT.pdf
https://en.wikipedia.org/wiki/NOS_Primavera_Sound_2018
https://www.festivais.pt/Festivais-2019/NOS-Primavera-Sound-2019.html
https://www.coolture.pt/event/queima-das-fitas-do-porto-2019/
https://en.wikipedia.org/wiki/Festa_de_S%C3%A3o_Jo%C3%A3o_do_Porto
https://www.up.pt/portuguesuporto/2019/09/04/feira-do-livro-do-porto/
https://www.transfermarkt.pt/fc-porto/spielplan/verein/720/saison_id/2020
https://www.fpf.pt/pt/selecoes/futebol-masculino/selecao-a/jogos
https://www.introducingporto.com/public-holidays