Universidade do Minho

Escola de Ciências da Universidade do Minho
Departamento de Informática

Mestrado em Matemática e Computação
Mestrado Integrado em Engenharia Informática

Redes Neuronais Recorrentes para previsão do fluxo de tráfego rodoviário

Alunos:
Andreia Costa (PG37013)
Henrique Faria (A82200)
Paulo Barbosa (PG40160)
Rui Teixeira (PG37021)

Docentes:
Bruno Fernandes
Victor Alves

Unidade Curricular: Classificadores e Sistemas Conexionistas

Maio
2020

Conteúdo

1 Introdução

2 *Dataset*

Aquando da apresentação do presente trabalho foram disponibilizados dados referentes a duas cidades: Braga e Porto, sendo que o grupo escolheu os dados relativos à cidade de Braga para trabalhar.

Os dados encontram-se distribuídos em 4 *datasets*:

- *Traffic Flow Braga Until 20191231*;
- *Traffic Incidents Braga Until 20191231*;
- *Weather Braga Descriptions Until 20191231*;
- *Weather Braga Until 20191231.*

Todos os *datasets* contêm dados relativos ao período entre 15 Janeiro 2019 e 31 Dezembro 2019.

2.1 *Traffic Flow Braga*

O *dataset "Traffic Flow Braga"* é constituído pelos seguintes atributos:

- $city\_name$;
- $road\_num$;
- $road\_name$;
- $functional\_road\_class\_desc$;
- $current\_speed$;
- $free\_flow\_speed$;
- $speed\_diff$;
- $current\_travel\_time$;
- $free_flow\_travel\_time$;
- $time\_diff$;
- $creation\_date$.

2.2 *Traffic Incidents Braga*

- $city\_name$;
- $description$;
- $cause\_of\_incident$;
- $from\_road$;
- to_road;
- $affected\_roads$;
- $incident\_category\_desc$;
- $magnitude\_of\_delay\_desc$;
- $length\_in\_meters$;
- $delay\_in\_seconds$;
- $incident\_date$;
- $latitude$;
- $longitude$.

2.3 *Weather Braga Descriptions*

- $city\_name$;
- $cloudiness$;
- $atmosphere$;
- $snow$;
- $thunderstorm$;
- $rain$;
- $sunrise$;
- $sunset$;
- $creation\_date$.

2.4 *Weather Braga*

- *city_name*;
- *temperature*;
- *atmospheric_pressure*;
- *humidity*;
- *wind_speed*;
- *clouds*;
- *precipitation*;
- *current_luminosity*;
- *sunrise*;
- *sunset*;
- *creation_date*.

2.5 Preparação dos dados

Após análise dos quatro *datasets* concluiu-se que, antes de se desenvolver o modelo para a previsão da *feature speed_diff*, era necessário fazer uma prévia preparação dos dados.

Começou-se por fazer um tratamento inicial do *dataset Traffic_Incidents*. Para isso, a cada incidente atribuiu-se os vários valores da coluna *road_num*, para que posteriormente fosse possível avaliar a distância entre os incidentes e as ruas em estudo e verificar de que forma estes incidentes afetam uma determinada rua.

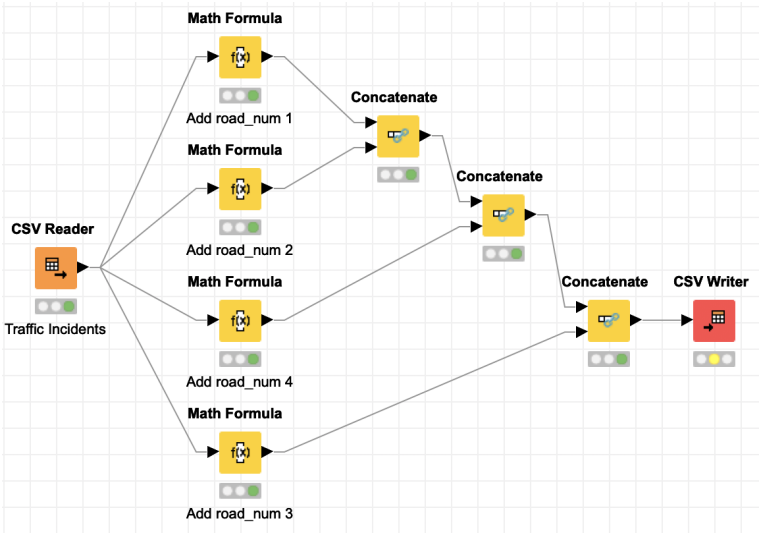

Figura 1: Preparação do *dataset Traffic_Incidentes*.

De seguida, recorrendo à latitude e longitude dos diferentes acontecimentos, calculou-se a distância dos incidentes a cada uma das ruas, para perceber o raio de influência dos incidentes para as ruas em estudo, de modo, a ser possível, posteriormente, remover incidentes que se encontrem muito afastados das ruas em estudo. Para isso, recorreu-se à fórmula:

$$dist(A,B) = R * \arccos(sin(lat_A) * \sin(lat_B) + \cos(lat_A) * \cos(lat_B) * \cos(lon_A - lon_B)).$$

onde,

- lat_A: latitude do ponto A;
- lat_B: latitude do ponto B;
- lon_A: longitude do ponto A;
- lon_B: longitude do ponto B;
- R: raio da Terra.

tendo-se implementado o seguinte código.

```
import pandas as pd
from math import radians, sin, cos, atan2, sqrt

df = pd.read_csv('Traffic_Incidents.csv', delimiter = ',',
    error_bad_lines = False, encoding = 'ISO-8859-1')
```

```
5
6  def distance(p1, n):
7    R = 6371.0
8    if n == 1:
9    lat2 = radians(41.548331)
10   lon2 = radians(-8.421298)
11   elif n == 2:
12   lat2 = radians(41.551356)
13   lon2 = radians(-8.420001)
14   elif n == 3:
15   lat2 = radians(41.546639)
16   lon2 = radians(-8.433517)
17   else:
18   lat2 = radians(41.508849)
19   lon2 = radians(-8.462299)
20   lat1, lon1 = radians(p1[0]), radians(p1[1])
21   dlon = lon2 - lon1
22   dlat = lat2 - lat1
23   a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon /
      2)**2
24   c = 2 * atan2(sqrt(a), sqrt(1 - a))
25   distance = R * c
26 return distance
27
28 df['Distance'] = df.apply(lambda row: distance((row['latitude
     '],row['longitude']), row['road_num']), axis=1)
```

Após calculadas todas as distâncias fez-se um tratamento estatístico, tendo-se obtido os seguintes resultados:

- $max = 6313,251$;
- $min = 0,0228$;
- $mean = 4,507$;
- $standard\ deviation = 81,789$.

Através dos resultados obtidos é possível verificar que existem dados mal classificados, uma vez que, sendo os dados recolhidos referentes apenas à cidade de Braga era impossível que a distância máxima dos incidentes às ruas fosse de cerca de 6313 km. Fez-se um estudo desta informação e verificou-se que estes dados dizem respeito a uma cidade que não pertence a Braga.

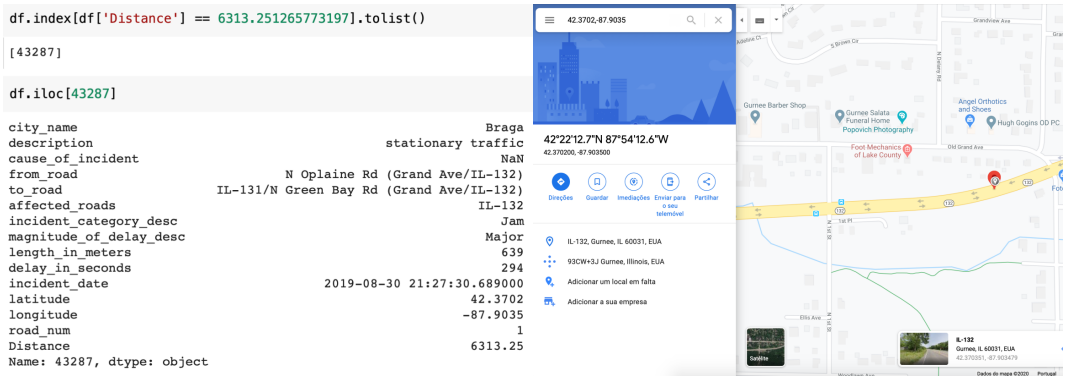

Figura 2: Dado mal classificado.

Devido a este facto, optou-se por remover alguns dados do *dataset*. Uma vez que a distância é medida em linha reta utilizou-se como *threshold*, para remover dados, vários valores, nomeadamente, 0.5, 1 e 1.5.

Após feito este tratamento procedeu-se à preparação dos dados referentes aos restantes *datasets*, com o intuito de se obter, no final, um único *dataset*.

Começou-se por fazer o tratamento do *dataset Weather_Descriptions_Braga*, tendo-se removido as colunas: *city_name*, *snow* e *cloudiness*. A coluna *snow* apresentava apenas *missing values*, daí se ter optado pela sua remoção. Relativamente à coluna *cloudiness*, optou-se por fazer a remoção da mesma, uma vez que existe uma coluna que está diretamente relacionada com esta, a coluna *cloud*, pertencente ao *dataset Weather_Braga*, e que não apresenta *missing values*.

De seguida, procedeu-se à remoção das colunas *city_name* e *precipitation* do *dataset Weather_Braga*. A remoção da coluna *precipitation* deveu-se ao facto desta apenas apresentar um único valor, o 0.

De modo a unir o resultado da preparação dos dados feita para os *datasets* anteriores, recorreu-se ao nodo *Joiner*, e uniram-se os *datasets* por *creation_date*, tendo-se efetuado, de seguida, a extração da data e do tempo, tendo-se extraído: o mês como uma variável numérica, a hora, o dia do mês e o dia da semana.

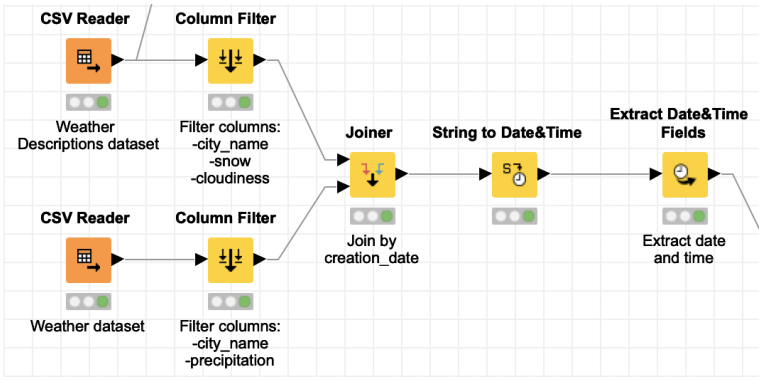

Figura 3: Preparação dos *datasets* *Weather_Descriptions_Braga* e *Weather_Braga*.

De seguida, procedeu-se à preparação do *dataset* *Traffic_Flow_Braga*, procedendo-se à remoção das colunas *city_name* e *road_name*, seguida da extração da data e hora, à semelhança do que foi feito para o *dataset* anterior.

O *dataset* *Traffic_Flow_Braga* tinha registos de 20 em 20 minutos e o *dataset* obtido anteriormente tinha registos de hora em hora, assim, de modo a unir o *dataset* com os dados relativos ao *Weather*, optou-se por agrupar os registos do *dataset* *Traffic_Flow_Braga* por hora, mês, dia e rua, recorrendo-se ao nodo *GroupBy*, tendo-se feito a média de todos os valores numéricos para as restantes colunas.

Assim, de modo a juntar este *dataset* ao obtido anteriormente, recorreu-se ao nodo *Joiner*, unindo-se os *datasets* por hora, dia do mês e mês, fazendo-se um *Left Outer Join*. Optou-se por fazer um *Left Outer Join*, uma vez que não se queriam as condições atmosféricas de registos em que não havia dados de tráfego.

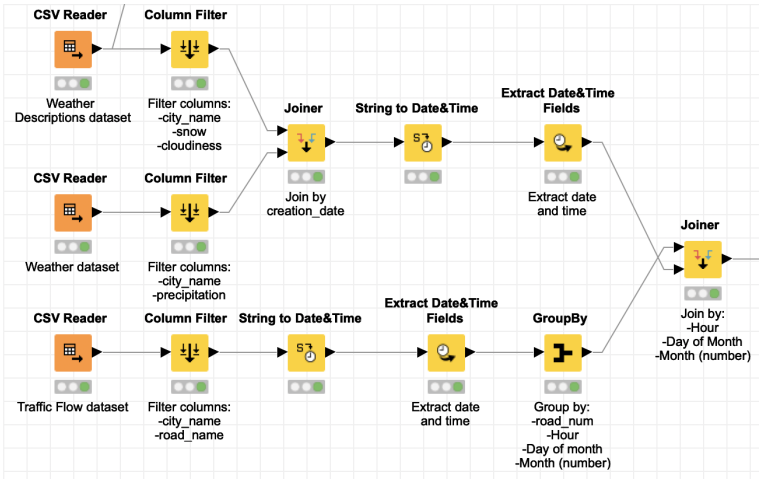

Figura 4: Preparação do *dataset Traffic_Flow_Braga*.

Após a junção dos *datasets*, eliminou-se a coluna *creation_date* e transformaram-se os valores "N/A", das colunas *rain*, *thunderstorm* e *atmosphere*, em *missing values*, recorrendo ao nodo *String Manipulation*. De seguida, quando não existiam valores na coluna *rain* atribuia-se o valor da coluna *thunderstorm*, uma vez que as *labels* da coluna *thunderstorm* faziam referência ao estado da chuva. De seguida, alteraram-se alguns dos valores ("$trovoada\ com\ chuva\ fraca$" $\rightarrow$ "$chuva\ fraca$", "$trovoada\ com\ chuva\ forte$" $\rightarrow$ "$chuva\ forte$" e "$trovoada$" $\rightarrow$ "$chuva$"), tendo-se removido, no final, a coluna *thunderstorm*. Por fim, eliminaram-se as colunas *sunrise* e *sunset*, uma vez que não se achou que estas colunas eram relevantes para prever o *speed_diff*.

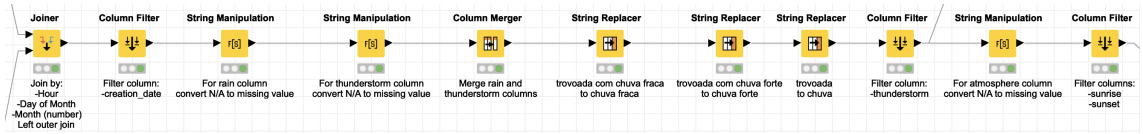

Figura 5: Preparação dos dados.

Por fim, tratou-se o *dataset Traffic_Incidents_Braga* cuja *feature Distance* tinha apenas valores inferiores a 0.5 km. Recorrendo à coluna *incident_date*, procedeu-se à extração do dia, da hora e do mês e removeram-se colunas irrelevantes, nomeadamente, as colunas *city_name*, *incident_date* e *cause_of_incident*, uma vez que esta última apresentava maioritariamente *missing values*.

Uma vez que a coluna *Distance* já inclui informação que permite relacionar a influência de um dado inicidente com as ruas em estudo, optou-se por remover as colunas: *from_road*, *to_road*, *affected_roads*, *latitute* e *longitude*.

Após tratado este *dataset*, e recorrendo ao nodo *Joiner*, uniu-se este *dataset* com o obtido anteriormente por hora, dia do mês, mês e *road_num*. Deste modo, uniram-se os 4 *datasets* iniciais num único.

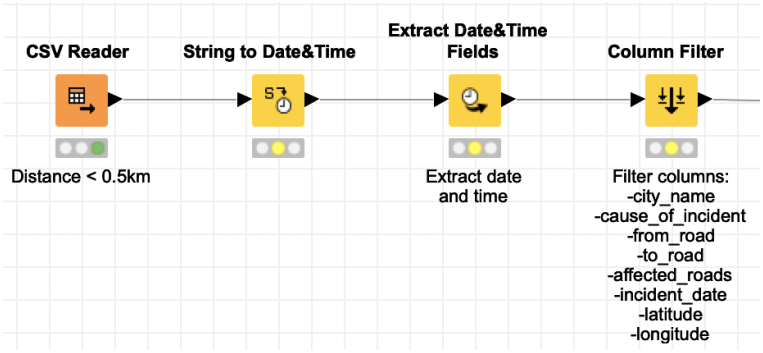

Figura 6: Preparação do *dataset* resultante do tratamento do *dataset Traffic_Incidents_Braga*.

Após se ter apenas um *dataset* verificou-se que este continha 26 colunas, o que se achou serem demasiadas. Assim, de modo a tornar o *dataset* mais pequeno, recorreu-se ao nodo *Rank Correlation* e avaliou-se a correlação que existia entre as diferentes colunas, tendo-se removido as seguintes: *free_flow_speed*, *current_travel_time*, *free_flow_travel_time* , *atmospheric_pressure*, *humidity*, *current_luminosity* e *magnitude_of_delay_desc*. Deste modo, o *dataset* ficou apenas com 18 colunas.

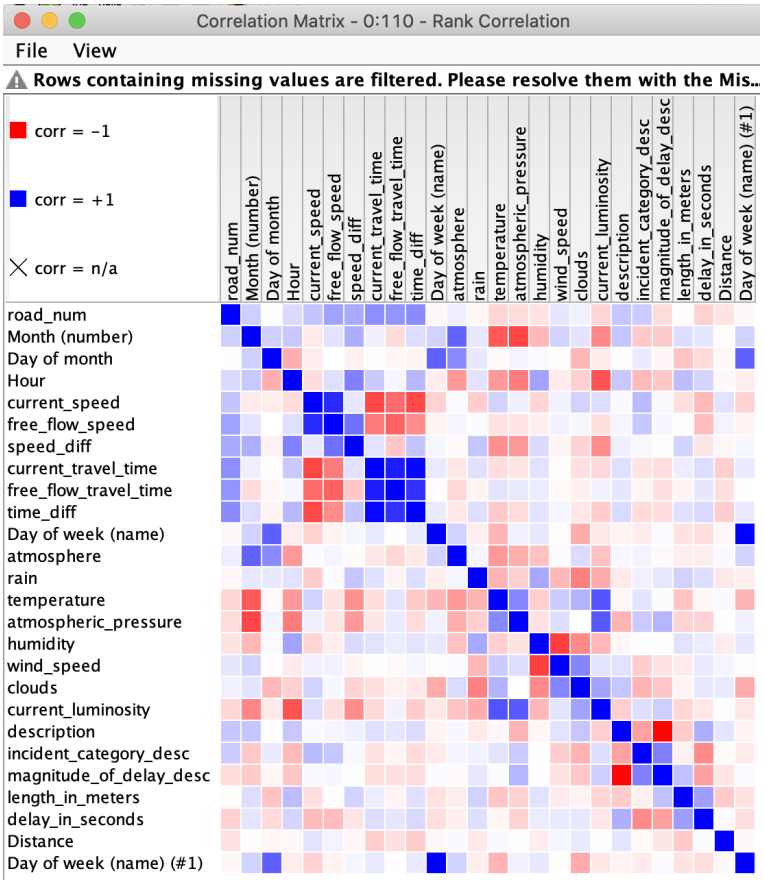

Figura 7: Análise da correlação entre as diferentes *features*.

Aos valores *Undefined* da *feature descriptions* atribui-se o valor *Unknown Delay*, uma vez que estes têm significado semelhante.

De seguida, e tendo em conta que as colunas *atmosphere* e *rain* apresentam muitos *missing values*, procedeu-se ao tratamento dos mesmos.

Começou-se, então, por tratar os *missing values* da coluna *atmosphere*, uma vez que esta era a que apresentava menos *missing values*, tendo-se separado o *dataset* em dois, recorrendo ao nodo *Rule-based Row Splitter*. Um *dataset* apresenta a coluna *atmosphere* apenas com *missing values* e o outro apresenta a coluna *atmosphere* com os vários valores. De seguida, utilizaram-se *Random Forest* para fazer a previsão dos *missing values*.

Com o intuito de perceber quais os melhores parâmetros a utilizar efetuou-se o *tunning* do modelo.

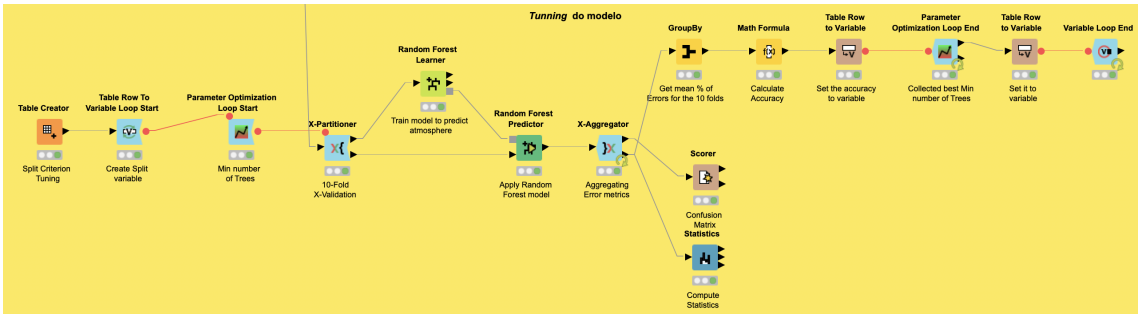

Figura 8: *Tunning* do modelo.

Após efetuado o *tunning* do modelo, concluiu-se que este apresentava melhores valores se fosse treinado com 60 árvores e usando como critério de *split* o *Information Gain*, tendo-se uma *accuracy* de cerca $99,5\%$

Data table of flow variables - 0:129 - Va...

File Hilite Navigation View

Table "default" – Rows: 3

Row ID	Trees	Split
Row0	60	InformationGain
Row1	160	InformationGainRatio
Row2	160	Gini

Output data - 0:133 - Math Form...

File Hilite Navigation View

Table "default" – Rows: 7

Row ID	Error in %	Accuracy
Row0	0	99.446
Row1	0.298	99.446
Row2	0.299	99.446
Row3	0.595	99.446
Row4	0.597	99.446
Row5	0.896	99.446
Row6	1.194	99.446

Figura 9: Melhores parâmetros para construir o modelo.

Por fim, sabendo quais os melhores parâmetros, contruiu-se um novo modelo usando 100% dos dados para o treinar.

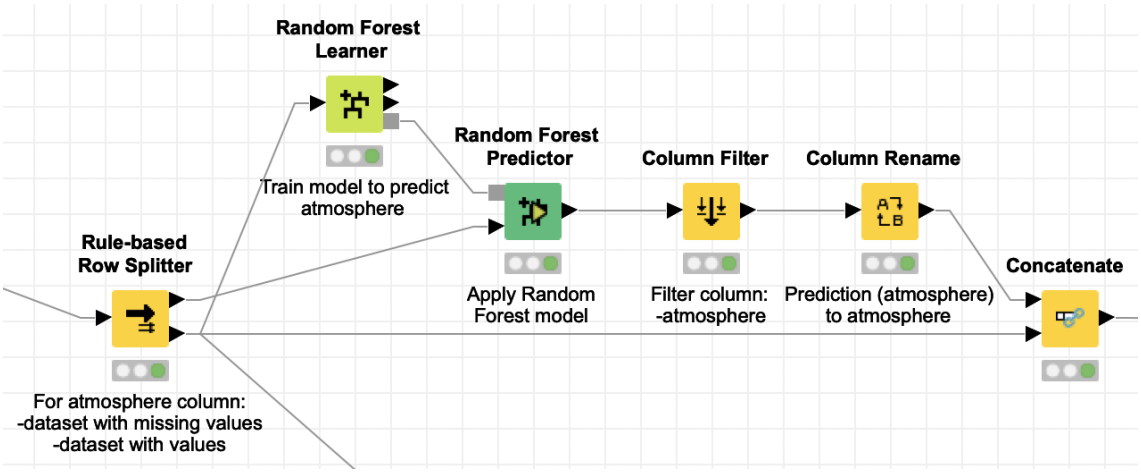

Figura 10: Previsão dos *missing values* da *feature atmosphere*.

Após feita a previsão dos *missing values* para a *feature atmosphere*, procedeu-se à previsão dos *missing values* do atributo *rain*, tendo-se utilizado o mesmo esquema para obter os melhores parâmetros, tendo-se obtido uma *accuraccy* de cerca de $97,8\%$.

Para finalizar o tratamento de dados, no *Knime*, recorrendo ao nodo *Duplicate Row Filter*, eliminaram-se linhas repetidas e efetuou-se o *Label Encoding* dos valores correspondentes às *features*: *Day of week (name)*, *description*, *incident_category_desc*, *atmosphere* e *rain*, uma vez que o objetivo é utilizar redes neuronais para prever o *speed_diff*, e estas apenas aceitam valores numéricos.

É de notar que, após feito todo este tratamento, existem colunas que apresentam *missing values*. No entanto, estes *missing values* ocorrem nas colunas correspondentes aos incidentes, porque não houve incidentes naquela hora, numa distância inferior a 0.5 km. Estes *missing values* foram substituídos por um valor *default*, -1.

Por fim, observou-se que existiam dias com horas repetidas, devido ao facto de para uma mesma hora existir mais do que um incidente. Assim, para que isto não acontecesse, recorreu-se ao nodo *GroupBy* para agrupar os incidentes, optando-se por ficar com o incidente que estava mais próximo da rua em estudo, ou seja, o incidente que apresentava na coluna *Distance* o valor mais baixo.

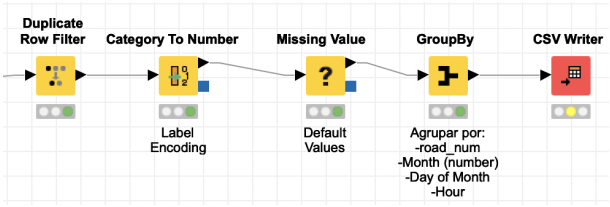

Figura 11: Tratamento final.

Após feito este tratamento, recorrendo ao nodo *Pie chart (local)* percebeu-se que o *dataset* apresentava dias e horas em falta como, por exemplo, o mês de Março.

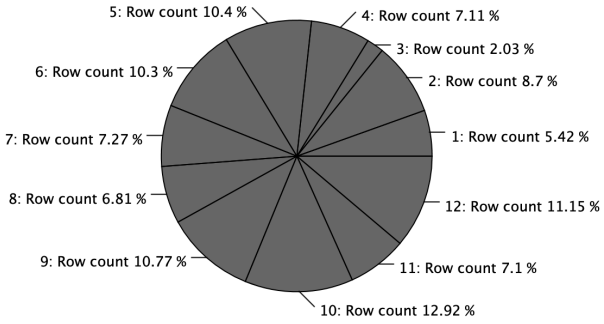

Figura 12: Dias em falta no *dataset*.

3 Problema

O objetivo do trabalho consiste em utilizar Redes Neuronais para fazer previsão do fluxo de tráfego rodoviário. Tendo em conta os dados disponibilizados, o grupo optou por fazer a previsão da *feature speed_diff* de uma dada rua, baseando-se em 3 dias para prever o dia seguinte. Optou-se por fazer previsões para as ruas em separado, uma vez que se acredita que o fluxo de tráfego das ruas é distinto entre elas. Para isso, dividiu-se o *dataset* original em 4, sendo que cada *dataset* correspondia a uma determinada rua.

Uma vez que se trata de um problema de séries temporais, optou-se por implementar um modelo *multistep* e *multivariate* que utiliza *LSTM*'s.

3.1 Resolução do Problema

Para resolver este problema era então necessário perceber quais os dias que estavam incompletos, ou seja, quais os dias que não tinham as 24 horas

preenchidas, procedendo-se à eliminação destes, com o intuito de se ter um *dataset* sem "buracos". Para isso, implementou-se o seguinte algoritmo:

```
i=0
for i in range(1,13):
  for j in range(1,32):
  L=df[(df['Month (number)']==i)&(df['Day of month']==j)].
   dropna()
  L1=L[['Month (number)','Day of month','Hour','road_num']]
  L1 = L1.drop_duplicates()
  indexNames = df[(df['Month (number)']==i)&(df['Day of month
   ']==j)].index
  if len(L1)<24:
  try:
    df.drop(indexNames, inplace=True)
  except:
    pass
```

Visto que no final do capítulo **??** verificámos que existiam dias em falta e como se pretende que sejam dados ao modelo, como *input*, 3 dias para prever o próximo, quer garantir-se que, de facto, esses 4 dias são seguidos, ou seja, que os 3 dias dados como *input* mais o dia a prever sejam seguidos, evitando que ocorram situações em que o primeiro dia seja, por exemplo, 16 de Janeiro e o dia a prever seja 30 de Janeiro.

Assim, para ter a certeza que se treina o modelo com dias consecutivos, percorreu-se o *dataset* construindo blocos de 4 dias, para verificar se estes 4 dias são seguidos, calcula-se a diferença entre o último dia do bloco e o primeiro e, dependendo do mês, verifica-se se é 4. Note-se, no entanto, que se para um dado bloco tivermos dias de meses distintos, esta diferença é negativa, sendo este problema corrigido dependendo do mês em causa.

Antes de se aplicar o seguinte algoritmo, ordenou-se o *dataset* por mês, dia e hora.

```
n_future = 24 # next 24 hours speed diff forecast
n_past = 24*3 # Past 3 days

x_train = []
y_train = []
label = df_1['speed_diff']

for i in range(0,len(df_1)-n_past-n_future+1):
  dias = df_1.iloc[i : i + n_past+24]
  mes = dias.iloc[0]['Month (number)']
  dia_1 = dias.iloc[0]['Day of month']
  dia_4 = dias.iloc[24*3+1]['Day of month']
  if (mes == 4 or mes == 6 or mes == 9 or mes == 11) and (
   dia_4 - dia_1 == 3 or dia_4 - dia_1 == -29):
    x_train.append(df_1.iloc[i : i + n_past])
```

```
15      y_train.append(label.iloc[i + n_past : i + n_past +
        n_future ])
16    elif (mes == 1 or mes == 3 or mes == 5 or mes == 7 or mes
        == 8 or mes == 10 or mes == 12) and (dia_4 - dia_1 == 3 or
        dia_4 - dia_1 == -28):
17      x_train.append(df_1.iloc[i : i + n_past])
18      y_train.append(label.iloc[i + n_past : i + n_past +
        n_future ])
19    elif mes == 2 and (dia_4 - dia_1 == 3 or dia_4 - dia_1 ==
        -26):
20      x_train.append(df_1.iloc[i : i + n_past])
21      y_train.append(label.iloc[i + n_past : i + n_past +
        n_future ])
```

Sabendo-se que cada *input* é constituído por 3 dias seguidos, com as 24 horas completas e, portante, as colunas *Month (number)*, *Day of month* e *Hour* já não são relevantes, tendo-se feito a remoção das mesmas. Além disso, removeram-se as colunas *Day of week (name)*, *incident_category_desc* e *Distance*. Esta última foi removida, uma vez que apenas serviu para saber quais os incidentes que deviam permanecer no *dataset*.

Após feito todo o tratamento acima mencionado, o *dataset* está pronto para ser aplicado a uma rede que permita prever a *feature speed_diff*.

4 Modelo

O problema que se pretende resolver é um problema de séries temporais, como já foi anteriormente referido. Deste modo, para prever a *feature speed_diff* optou-se por construir um modelo *multistep* e *multivariate* que utiliza *LSTM*'s.

Tendo em conta que o objetivo é utilizar 3 dias para prever as 24 horas seguintes, utilizando 11 *features*, definiu-se como $input_shape = (24 * 3, 11)$.

Antes de se começar a treinar o modelo procedeu-se à normalização dos dados:

```
1  # Features normalization
2  scalers=[]
3  for i in range(11):
4  sc = MinMaxScaler(feature_range=(0,1))
5  x_train[:,i] = sc.fit_transform(x_train[:,i])
6  x_test[:,i] = sc.fit_transform(x_test[:,i])
7  scalers.append(sc)
8
9  # Labels normalization
10 sc1 = MinMaxScaler(feature_range=(0,1))
11 y_train = sc1.fit_transform(y_train)
12 y_test_n = sc1.fit_transform(y_test)
```

Uma vez que os dados estavam normalizados no intervalo $[0, 1]$ recorreu-se à função de ativação *sigmoid*.

Relativamente às métricas utilizadas, utilizou-se como *loss* o *mean square error* e como métrica o *root mean square error*. Estas foram as métricas escolhidas, uma vez que o objetivo era penalizar erros grandes, e estas são as melhores métricas para o fazer.

Assim, recorrendo a técnicas de intuição e experimentação construiu-se o seguinte modelo:

```
model = Sequential()
model.add(CuDNNLSTM(units=24*3, return_sequences=True,
    input_shape = (24*3,11) ) )
model.add(Dropout(0.2))
model.add(CuDNNLSTM(24*3 , return_sequences=True))
model.add(Dropout(0.2))
model.add(CuDNNLSTM(24*3, return_sequences=True))
model.add(Dropout(0.2))
model.add(CuDNNLSTM(24*2))
model.add(Dropout(0.2))
model.add(Dense(24,activation='sigmoid'))
model.compile(optimizer='adam', loss='mean_squared_error',
    metrics=rmse)
callback = tf.keras.callbacks.EarlyStopping(monitor='loss',
    patience=20)
history=model.fit(x_train, y_train, validation_data=(x_test,
    y_test_n), epochs=1000, callbacks=[callback])
```

4.1 Avaliação do comportamento do modelo

Após construído o modelo, procedeu-se a um conjunto de testes, com o intuito de ser possível avaliar o comportamento do mesmo.

Para se perceber se o comportamento do modelo era o esperado procedeu-se à utilização de duas métricas de erro.

Uma primeira abordagem passa por fazer a média dos erros para os diferentes dias da semana.

Outra consiste em considerar o valor absoluto da máxima diferença entre o valor real e o valor previsto

4.1.1 Rua 1

Começou-se por treinar o modelos com os dados da rua 1, e fazer a previsão para essa mesma rua.

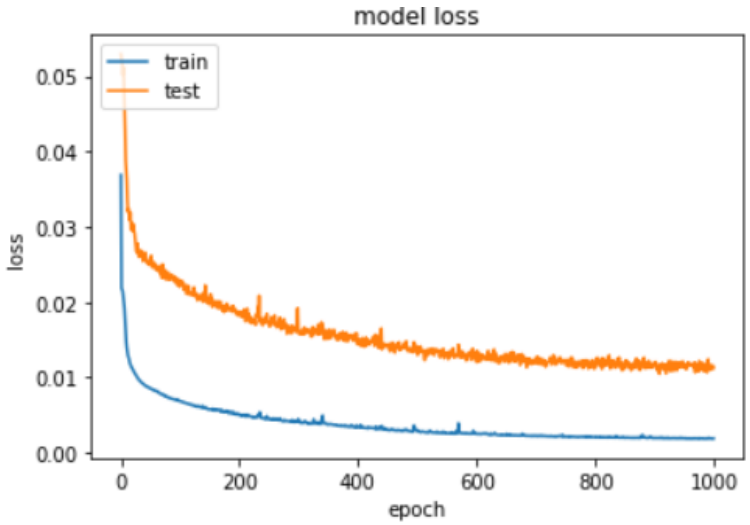

Figura 13: Curvas de aprendizagem.

Observando a Figura 13, conclui-se que o modelo criado apresenta *underfitting*, uma vez que as curvas de aprendizagem são distintas uma da outra. Pode ainda referir-se que, apesar de existir *underfitting*, é provável que com o aumento do número de épocas não se observe uma convergência das curvas, uma vez que se observa que ambas as curvas parecem ter estagnado, o que evidencia que o modelo não está a aprender.

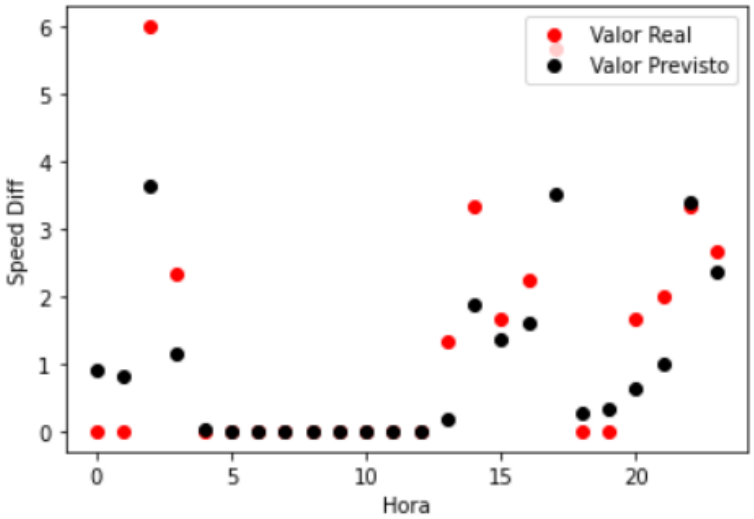

Figura 14: Valores reais vs previstos, em 24 horas.

A Figura 14 permite comparar, graficamente o valor teórico com o valor real, durante um período de 24 horas. Ora, através da análise do gráfico

observa-se que não existe uma grande discrepância entre o valor real e o valor previsto.

4.1.2 Rua 2

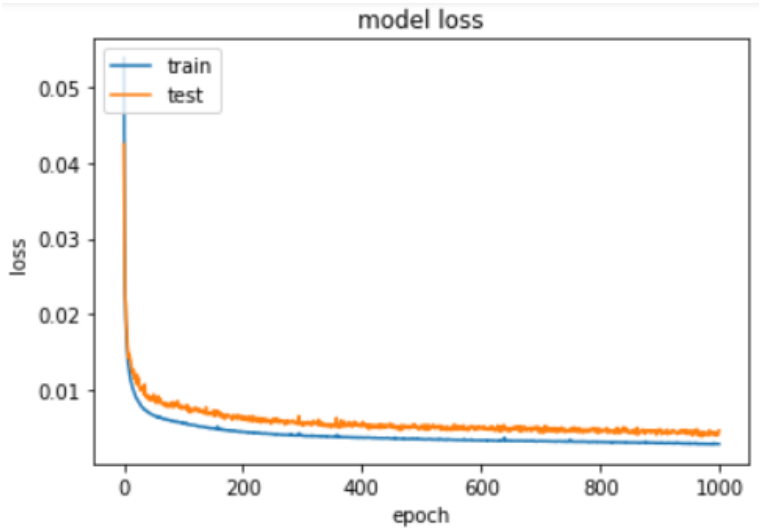

Figura 15: Curvas de aprendizagem.

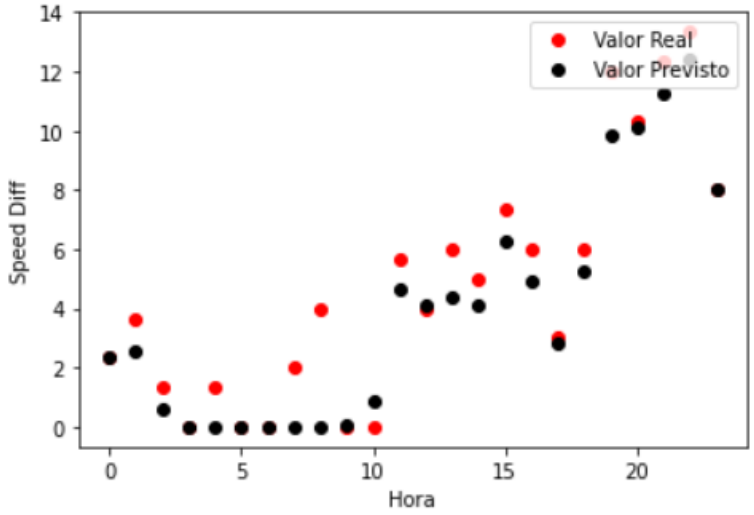

Figura 16: Valores reais vs previstos, em 24 horas.

4.1.3 Rua 3

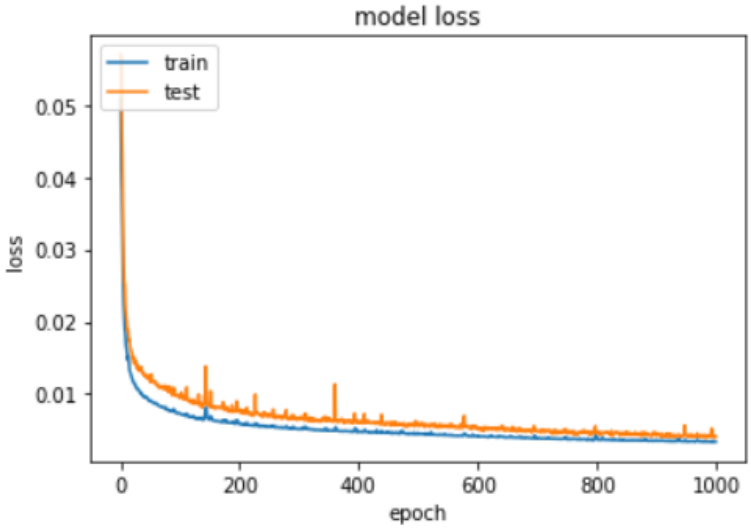

Figura 17: Curvas de aprendizagem.

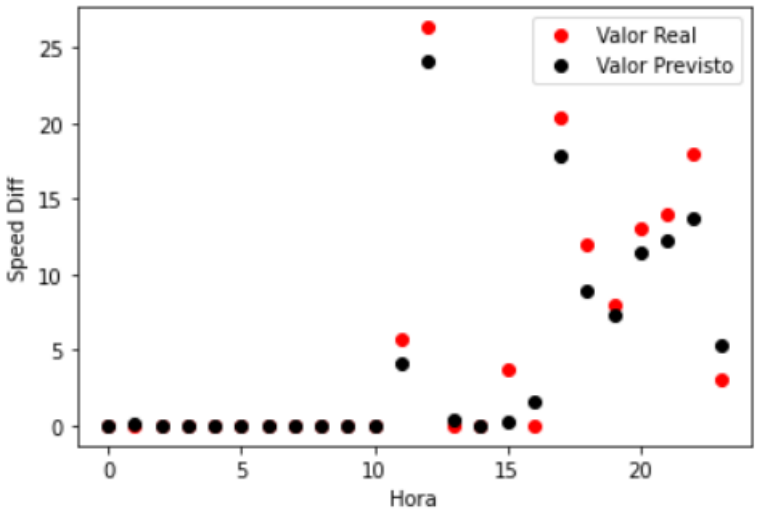

Figura 18: Valores reais vs previstos, em 24 horas.

4.1.4 Rua 4

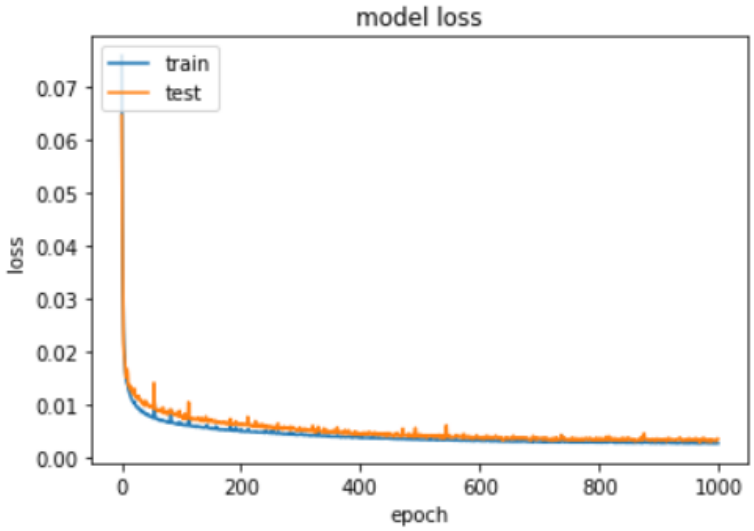

Figura 19: Curvas de aprendizagem.

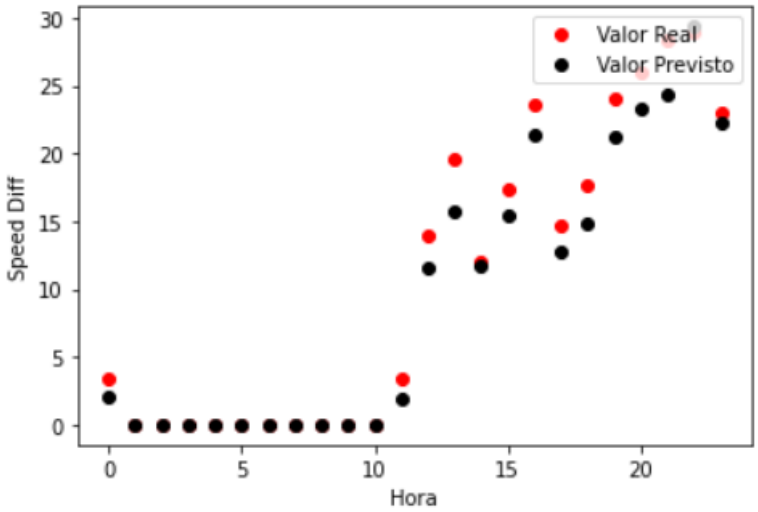

Figura 20: Valores reais vs previstos, em 24 horas.