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 prévio tratamento do *dataset Traffic_Incidents*. Para isso, quadriplicou-se esse *dataset*, com o intuito de atribuir todas as ruas em estudo a todos os incidentes, para que posteriormente fosse possível avaliar a distância entre os incidentes e as ruas em estudo.

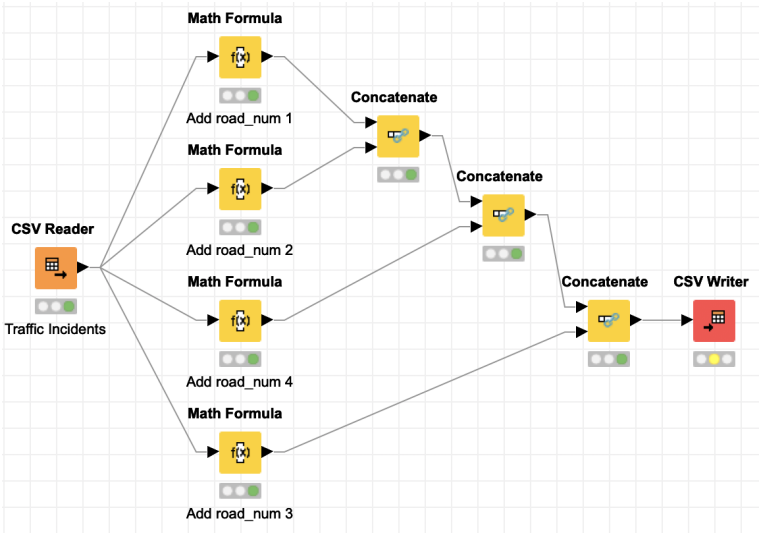

Figura 1: Preparação do *dataset Traffic_Incidentes*.

De seguida, recorrendo à latitude e longitude dos diferentes acontecimentos, calculou-se a distância dos incidentes a cada uma das ruas, para perceber quais os incidentes que podiam influenciar o *speed_diff* de uma determinada rua.

```
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')

def distance(p1, n):
  R = 6371.0
  if n == 1:
  lat2 = radians(41.548331)
  lon2 = radians(-8.421298)
  elif n == 2:
  lat2 = radians(41.551356)
  lon2 = radians(-8.420001)
  elif n == 3:
  lat2 = radians(41.546639)
  lon2 = radians(-8.433517)
  else:
  lat2 = radians(41.508849)
  lon2 = radians(-8.462299)
  lat1, lon1 = radians(p1[0]), radians(p1[1])
  dlon = lon2 - lon1
  dlat = lat2 - lat1
  a = sin(dlat / 2)**2 + cos(lat1) * cos(lat2) * sin(dlon /
```

```
    2)**2
  c = 2 * atan2(sqrt(a), sqrt(1 - a))
  distance = R * c
return distance

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 errados, 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.

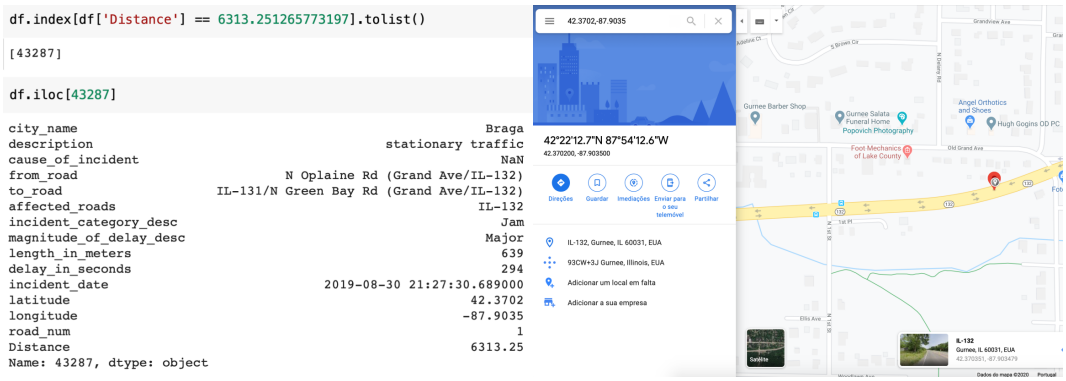

Figura 2: Dado mal classificado.

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*, 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.

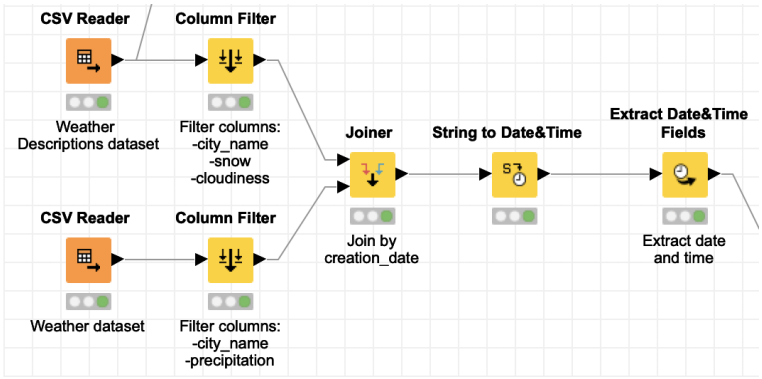

Figura 3: Preparação dos *datasets Weather_Descriptions_Braga* e *Weather_Braga*.

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 e agrupamento dos dados por *road_num*, hora, dia do mês e mês.

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* resultante de unir os *datasets Weather_Descriptions_Braga* e *Weather_Braga*, optou-se por agrupar os registos do *dataset Traffic_Flow_Braga* por hora.

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 incidentes.

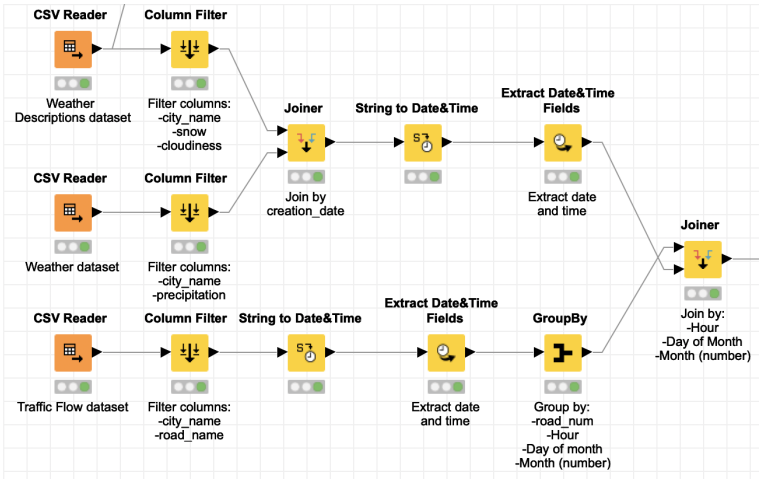

Figura 4: Preparação do *dataset Traffic_Flow_Braga*.

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, fez-se um *merge* das colunas *rain* e *thunderstorm*, tendo-se alterado 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*.

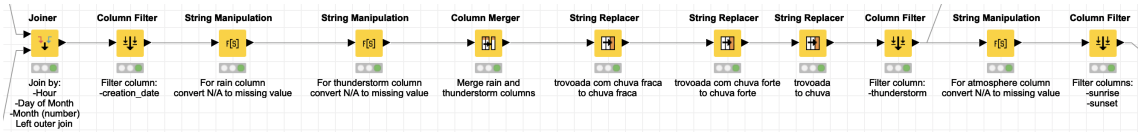

Figura 5: Preparação dos dados.

Por fim, tratou-se o *dataset* cuja *feature Distance* tinha apenas valores inferiores a $0,5$ km. Procedeu-se à extração do dia e da hora e removeram-se as colunas irrelevantes. 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.

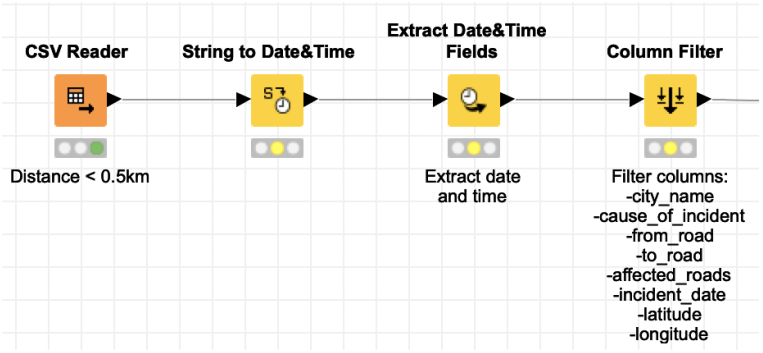

Figura 6: Preparação do *dataset* resultante do tratamento do *dataset Traffic_Incidents_Braga*.

Após se ter apenas um *dataset* eliminaram-se colunas que apresentavam uma correlação muito alta, de modo a tornar o *dataset* mais pequeno. Esta análise foi efetuada recorrendo ao nodo *Rank Correlation*.

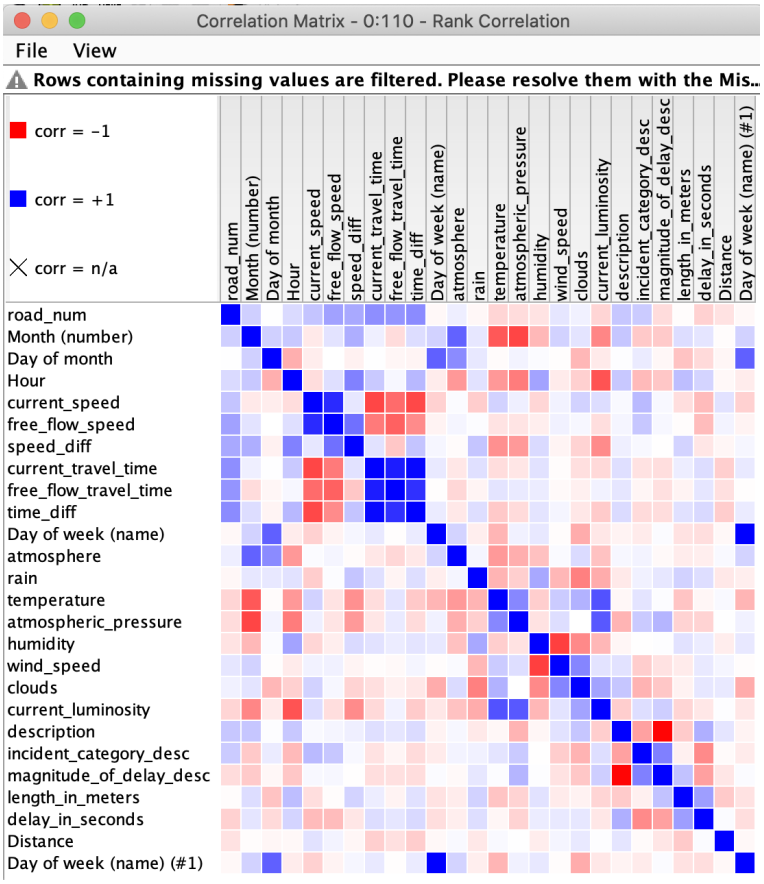

Figura 7: Análise da correlação entre as diferentes *features*.

Aos valores *Undefined* da *feature descriptions* atribui-se o valor *Unknown Delay*, com o objetivo de diminuir a quantidade de atributos desta *feature*.

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 este era o 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.

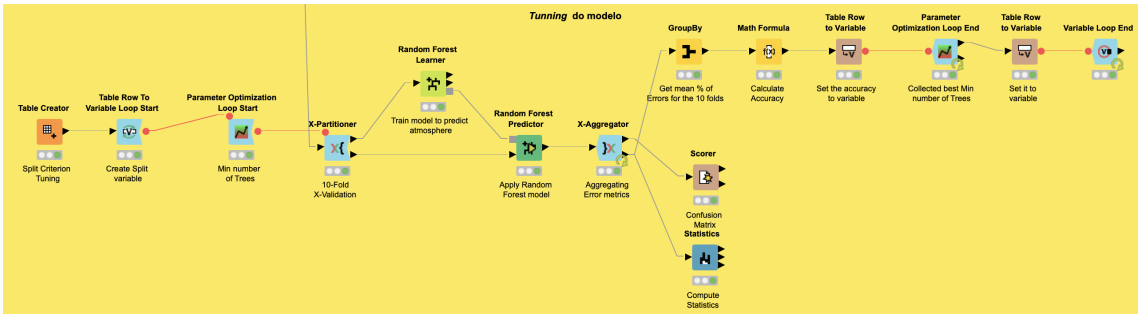

Figura 8: *Tunning* do modelo.

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, fez-se a previsão dos *missing values* da *feature atmosphere*, usando 100% dos dados para treinar o modelo.

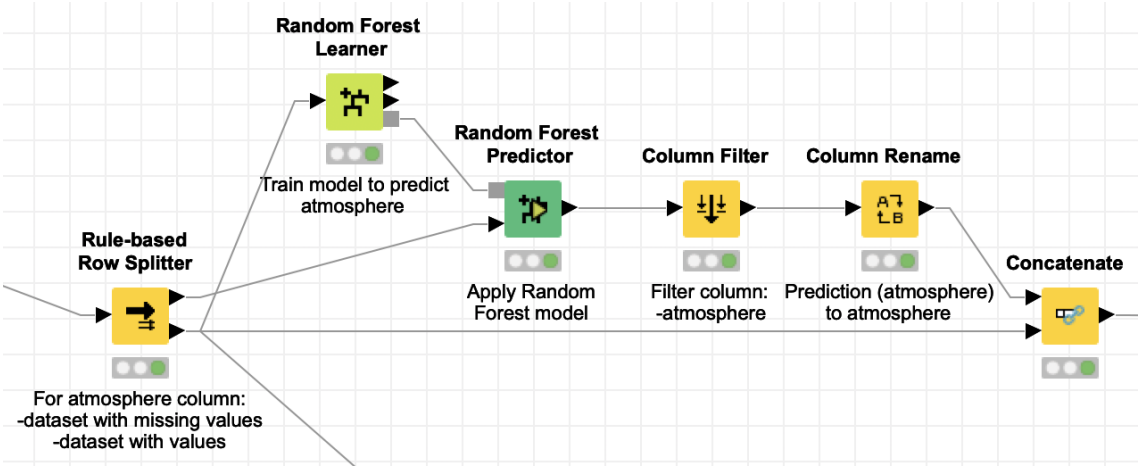

Figura 10: Previsão dos *missing values* da *feature atmosphere*.

Após feita a previsão dos *missing values* para a *feature atmosphere*, procedeu-se à previsão dos *missing values* do atributo *rain*, utilizando os mesmo 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*.

De seguida, trataram-se os *missing values*, substituindo-os por um valor *default*. Os *missing values* existentes correspondiam a dias/horas onde não tinham ocorrido incidentes.

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, de modo a que cada dia tivesse apenas 24 horas, recorreu-se ao nodo *GroupBy* para agrupar os incidentes, optando-se por ficar com o incidente que estava mais próximo da rua em estudo, uma vez que seria esse que mais influenciaria o atributo *speed_diff*.

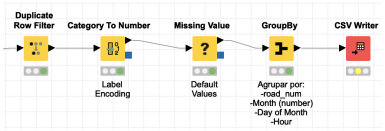

Figura 11: Tratamento final.

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 Maio.

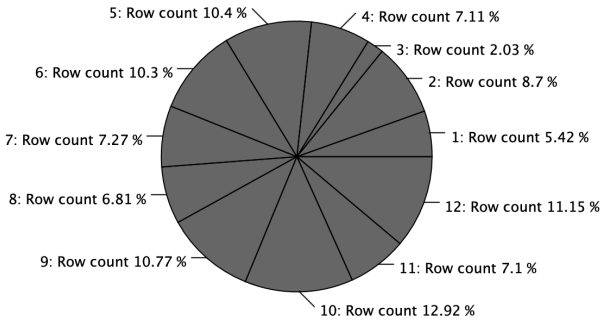

Figura 12: Dias em falta no *dataset*.

Deste modo, foi 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)<96:
  try:
    df.drop(indexNames, inplace=True)
  except:
    pass
```

Com o objetivo de ter a certeza que se treina o modelo com dias seguidos, desenvolveu-se o seguinte código *python*:

```
n_future = 24 # next 4 days temperature forecast
n_past = 24*7 # Past 30 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_168 = dias.iloc[168]['Day of month']
  if (mes == 4 or mes == 6 or mes == 9 or mes == 11) and (
   dia_168 - dia_1 == 7 or dia_168 - dia_1 == -25):
    x_train.append(df_1.iloc[i : i + n_past])
    y_train.append(label.iloc[i + n_past : i + n_past +
   n_future ])
  elif (mes == 1 or mes == 3 or mes == 5 or mes == 7 or mes
   == 8 or mes == 10 or mes == 12) and (dia_168 - dia_1 == 7
   or dia_168 - dia_1 == -24):
    x_train.append(df_1.iloc[i : i + n_past])
    y_train.append(label.iloc[i + n_past : i + n_past +
   n_future ])
  elif mes == 2 and (dia_168 - dia_1 == 7 or dia_168 - dia_1
   == -22):
    x_train.append(df_1.iloc[i : i + n_past])
    y_train.append(label.iloc[i + n_past : i + n_past +
   n_future ])
```

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