

```

1 C:\Users\leoni\anaconda3\envs\gru\python.exe "C:/
  Program Files/JetBrains/PyCharm 2022.3.1/plugins/
  python/helpers/pydev/pydevconsole.py" --mode=client
  --host=127.0.0.1 --port=51057
2
3 import sys; print('Python %s on %s' % (sys.version,
  sys.platform))
4 sys.path.extend(['C:\\Users\\leoni\\OneDrive\\
  Documents\\Studium\\12. Semester\\Masterarbeit\\mta-
  model'])
5
6 Python 3.10.11 | packaged by Anaconda, Inc. | (main,
  Apr 20 2023, 18:56:50) [MSC v.1916 64 bit (AMD64)]
7 Type 'copyright', 'credits' or 'license' for more
  information
8 IPython 8.12.0 -- An enhanced Interactive Python.
  Type '?' for help.
9 PyDev console: using IPython 8.12.0
10
11 Python 3.10.11 | packaged by Anaconda, Inc. | (main,
  Apr 20 2023, 18:56:50) [MSC v.1916 64 bit (AMD64)] on
  win32
12 In [2]: import pandas as pd
13     ...: import plotly.express as px
14     ...: import matplotlib.pyplot as plt
15     ...: import seaborn as sns
16     ...: import numpy as np
17     ...: from sklearn.preprocessing import
  OrdinalEncoder
18     ...: import tensorflow as tf
19     ...:
20     ...: def prep_df(df, max_journ_len): # function
  that does the preprocessing
21     ...:     # Transform the transaction column s.t.
  only last tp before conversion has transaction == 1
22     ...:     df['time_diff'] = df['
  timestamp_conversion'] - df['timestamp'] # create
  new var for timedifference
23     ...:
24     ...:     df.drop(df[df.time_diff < 0].index,
  inplace=True) # remove these time_diff < 0 i.e. tp

```

```

24 after transaction
25     ...:
26     ...:     df = df.sort_values('timestamp')
27     ...:     df = df.sort_values('journey_id')
28     ...:
29     ...:     groups = df.groupby('journey_id').
time_diff
30     ...:     min_val = groups.transform(min) # search
minimal time_diff in each group <=> closest tp to
conversion
31     ...:     cond1 = df.time_diff == min_val # define
condition when transaction should be 1
32     ...:
33     ...:     df['transaction'] = np.select([cond1], [1
], default=0) # transform transaction
34     ...:
35     ...:     # Long Journeys
36     ...:     max_journ_len = 16
37     ...:     df = df.groupby('journey_id').filter(
lambda x: len(x) <= max_journ_len)
38     ...:
39     ...:     # Remove Columns
40     ...:     df = df.drop(['s', 'timestamp_conversion
', 'time_diff'], axis=1) # cant be used for
prediction
41     ...:
42     ...:     # How to handle object variables
43     ...:     # Dummy variables for country, platform
and channel, better than Ordinal, but also huge data
44     ...:     df = pd.get_dummies(df, columns=['
channel_id'], prefix='channel', prefix_sep='_', dtype=
float)
45     ...:     df = pd.get_dummies(df, columns=['
country_name'], prefix='country', prefix_sep='_',
dtype=float)
46     ...:     df = pd.get_dummies(df, columns=['
platform'], prefix='platform', prefix_sep='_', dtype=
float)
47     ...:
48     ...:     return df
49     ...:

```

```

50     ...: # Next step: transform to tensor:
51     ...: def mta2tensor(data): # function that
    transforms dataset to tensor
52     ...:     df_transaction = data['transaction']
53     ...:     data = data.drop('transaction', axis=1)
54     ...:     grouss = data.groupby('journey_id')
55     ...:     x = []
56     ...:     y = []
57     ...:
58     ...:     for i in data['journey_id'].unique():
59     ...:         x1 = grouss.get_group(i)
60     ...:
61     ...:         x1 = x1.drop(['journey_id'], axis=1)
62     ...:         x1 = x1.values.tolist()
63     ...:
64     ...:         y1 = df_transaction.loc[grouss.
    get_group(i).index]
65     ...:         y1 = y1.values.tolist()
66     ...:
67     ...:         for j in range(max_journ_len - len(x1
    )): # for-loop for data padding (all customer
    journeys filled with zeros to get same length)
68     ...:             x1.append([0] * 52) # 52 is
    number of columns without journey_id and transaction
69     ...:             y1.append(0)
70     ...:             x.append(x1)
71     ...:             y.append(y1)
72     ...:
73     ...:     return tf.convert_to_tensor(x), tf.
    convert_to_tensor(y)
74     ...:
75     ...:
76     ...: data = pd.read_csv("data_sample1.csv")
77     ...: max_journ_len = 16
78     ...: data = prep_df(data, max_journ_len)
79     ...: x_train, y_train = mta2tensor(data)
80     ...: print(x_train)
81     ...:
82 Traceback (most recent call last):
83   File "C:\Users\leoni\anaconda3\envs\gru\lib\site-
    packages\IPython\core\interactiveshell.py", line 3505

```

```

83 , in run_code
84     exec(code_obj, self.user_global_ns, self.user_ns
    )
85     File "<ipython-input-2-cba034653d90>", line 68, in
        <module>
86         x_train, y_train = mta2tensor(data)
87     File "<ipython-input-2-cba034653d90>", line 53, in
        mta2tensor
88         y1 = df_transaction.loc[groups.get_group(i).
            index]
89 NameError: name 'groups' is not defined
90 In [3]: import pandas as pd
91         ...: import plotly.express as px
92         ...: import matplotlib.pyplot as plt
93         ...: import seaborn as sns
94         ...: import numpy as np
95         ...: from sklearn.preprocessing import
            OrdinalEncoder
96         ...: import tensorflow as tf
97         ...:
98         ...:
99         ...: def prep_df(df, max_journ_len): # function
            that does the preprocessing
100         ...:     # Transform the transaction column s.t.
            only last tp before conversion has transaction == 1
101         ...:     df['time_diff'] = df['
            timestamp_conversion'] - df['timestamp'] # create
            new var for timedifference
102         ...:
103         ...:     df.drop(df[df.time_diff < 0].index,
            inplace=True) # remove these time_diff < 0 i.e. tp
            after transaction
104         ...:
105         ...:     df = df.sort_values('timestamp')
106         ...:     df = df.sort_values('journey_id')
107         ...:
108         ...:     groups = df.groupby('journey_id').
            time_diff
109         ...:     min_val = groups.transform(min) #
            search minimal time_diff in each group <=> closest
            tp to conversion

```

```

110     ...:     cond1 = df.time_diff == min_val #
        define condition when transaction should be 1
111     ...:
112     ...:     df['transaction'] = np.select([cond1], [
        1], default=0) # transform transaction
113     ...:
114     ...:     # Long Journeys
115     ...:     max_journ_len = 16
116     ...:     df = df.groupby('journey_id').filter(
        lambda x: len(x) <= max_journ_len)
117     ...:
118     ...:     # Remove Columns
119     ...:     df = df.drop(['s', 'timestamp_conversion
        ', 'time_diff'], axis=1) # cant be used for
        prediction
120     ...:
121     ...:     # How to handle object variables
122     ...:     # Dummy variables for country, platform
        and channel, better than Ordinal, but also huge data
123     ...:     df = pd.get_dummies(df, columns=['
        channel_id'], prefix='channel', prefix_sep='_',
        dtype=float)
124     ...:     df = pd.get_dummies(df, columns=['
        country_name'], prefix='country', prefix_sep='_',
        dtype=float)
125     ...:     df = pd.get_dummies(df, columns=['
        platform'], prefix='platform', prefix_sep='_', dtype
        =float)
126     ...:
127     ...:     return df
128     ...:
129     ...: # Next step: transform to tensor:
130     ...:
131     ...:
132     ...: def mta2tensor(df): # function that
        transforms dataset to tensor
133     ...:     df_transaction = df['transaction']
134     ...:     df = df.drop('transaction', axis=1)
135     ...:     grouos = df.groupby('journey_id')
136     ...:     x = []
137     ...:     y = []

```

```
138     ...:
139     ...:         for i in data['journey_id'].unique():
140     ...:             x1 = grouper.get_group(i)
141     ...:
142     ...:             x1 = x1.drop(['journey_id'], axis=1)
143     ...:             x1 = x1.values.tolist()
144     ...:
145     ...:             y1 = df_transaction.loc[grouper.
get_group(i).index]
146     ...:             y1 = y1.values.tolist()
147     ...:
148     ...:             for j in range(max_journ_len - len(
x1)):
149     ...:                 # for-loop for data padding (all
customer journeys filled with zeros to get same
length)
150     ...:                 x1.append([0] * 52) # 52 is
number of columns without journey_id and transaction
151     ...:                 y1.append(0)
152     ...:                 x.append(x1)
153     ...:                 y.append(y1)
154     ...:
155     ...:         return tf.convert_to_tensor(x), tf.
convert_to_tensor(y)
156     ...:
157     ...:
158     ...: data = pd.read_csv("data_sample1.csv")
159     ...: max_journ_len = 16
160     ...: data = prep_df(data, max_journ_len)
161     ...: x_train, y_train = mta2tensor(data)
162     ...: print(x_train)
163     ...:
164
```