```
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

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['C:\\Users\\leoni\\OneDrive\\Documents\\Studium\\12. Semster\\Masterarbeit\\mta-model'])

Python 3.10.11 | packaged by Anaconda, Inc. | (main, Apr 20 2023, 18:56:50) [MSC v.1916 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.12.0 -- An enhanced Interactive Python. Type '?' for help.
PyDev console: using IPython 8.12.0

Python 3.10.11 | packaged by Anaconda, Inc. | (main, Apr 20 2023, 18:56:50) [MSC v.1916 64 bit (AMD64)] on win32
In [2]: import pandas as pd
   ...: import plotly.express as px
   ...: import matplotlib.pyplot as plt
   ...: import seaborn as sns
   ...: import numpy as np
   ...: from sklearn.preprocessing import OrdinalEncoder
   ...: import tensorflow as tf
   ...:
   ...: def prep_df(df, max_journ_len):  # function that does the preprocessing
   ...:     # Transform the transaction column s.t. only  last tp before conversion has transaction == 1
   ...:     df['time_diff'] = df['timestamp_conversion'] - df['timestamp']  # create new var for timedifference
   ...:
   ...:     df.drop(df[df.time_diff < 0].index, inplace=True)  # remove these time_diff < 0 i.e. tp
```

```
after transaction
   ...:
   ...:     df = df.sort_values('timestamp')
   ...:     df = df.sort_values('journey_id')
   ...:
   ...:     groups = df.groupby('journey_id').time_diff
   ...:     min_val = groups.transform(min)  # search minimal time_diff in each group <=> closest tp to conversion
   ...:     cond1 = df.time_diff == min_val  # define condition when transaction should be 1
   ...:
   ...:     df['transaction'] = np.select([cond1], [1], default=0)  # transform transaction
   ...:
   ...:     # Long Journeys
   ...:     max_journ_len = 16
   ...:     df = df.groupby('journey_id').filter(lambda x: len(x) <= max_journ_len)
   ...:
   ...:     # Remove Columns
   ...:     df = df.drop(['s', 'timestamp_conversion', 'time_diff'], axis=1)  # cant be used for prediction
   ...:
   ...:     # How to handle object variables
   ...:     # Dummy variables for country, platform and channel, better than Ordinal, but also huge data
   ...:     df = pd.get_dummies(df, columns=['channel_id'], prefix='channel', prefix_sep='_', dtype=float)
   ...:     df = pd.get_dummies(df, columns=['country_name'], prefix='country', prefix_sep='_', dtype=float)
   ...:     df = pd.get_dummies(df, columns=['platform'], prefix='platform', prefix_sep='_', dtype=float)
   ...:
   ...:     return df
   ...:
```

```
   ...: # Next step: transform to tensor:
   ...: def mta2tensor(data):  # function that transforms dataset to tensor
   ...:     df_transaction = data['transaction']
   ...:     data = data.drop('transaction', axis=1)
   ...:     grous = data.groupby('journey_id')
   ...:     x = []
   ...:     y = []
   ...:
   ...:     for i in data['journey_id'].unique():
   ...:         x1 = grous.get_group(i)
   ...:
   ...:         x1 = x1.drop(['journey_id'], axis=1)
   ...:         x1 = x1.values.tolist()
   ...:
   ...:         y1 = df_transaction.loc[groups.get_group(i).index]
   ...:         y1 = y1.values.tolist()
   ...:
   ...:         for j in range(max_journ_len - len(x1)):  # for-loop for data padding (all customer journeys filled with zeros to get same length)
   ...:             x1.append([0] * 52)  # 52 is number of columns without journey_id an transaction
   ...:             y1.append(0)
   ...:         x.append(x1)
   ...:         y.append(y1)
   ...:
   ...:     return tf.convert_to_tensor(x), tf.convert_to_tensor(y)
   ...:
   ...:
   ...: data = pd.read_csv("data_sample1.csv")
   ...: max_journ_len = 16
   ...: data = prep_df(data, max_journ_len)
   ...: x_train, y_train = mta2tensor(data)
   ...: print(x_train)
   ...:
Traceback (most recent call last):
  File "C:\Users\leoni\anaconda3\envs\gru\lib\site-packages\IPython\core\interactiveshell.py", line 3505
```

```
, in run_code
    exec(code_obj, self.user_global_ns, self.user_ns)
  File "<ipython-input-2-cba034653d90>", line 68, in <module>
    x_train, y_train = mta2tensor(data)
  File "<ipython-input-2-cba034653d90>", line 53, in mta2tensor
    y1 = df_transaction.loc[groups.get_group(i).index]
NameError: name 'groups' is not defined
In [3]: import pandas as pd
   ...: import plotly.express as px
   ...: import matplotlib.pyplot as plt
   ...: import seaborn as sns
   ...: import numpy as np
   ...: from sklearn.preprocessing import OrdinalEncoder
   ...: import tensorflow as tf
   ...:
   ...:
   ...: def prep_df(df, max_journ_len):  # function that does the preprocessing
   ...:     # Transform the transaction column s.t. only  last tp before conversion has transaction == 1
   ...:     df['time_diff'] = df['timestamp_conversion'] - df['timestamp']  # create new var for timedifference
   ...:
   ...:     df.drop(df[df.time_diff < 0].index, inplace=True)  # remove these time_diff < 0 i.e. tp after transaction
   ...:
   ...:     df = df.sort_values('timestamp')
   ...:     df = df.sort_values('journey_id')
   ...:
   ...:     groups = df.groupby('journey_id').time_diff
   ...:     min_val = groups.transform(min)  # search minimal time_diff in each group <=> closest tp to conversion
   ...:     cond1 = df.time_diff == min_val  # define condition when transaction should be 1
   ...:
   ...:     df['transaction'] = np.select([cond1], [1], default=0)  # transform transaction
   ...:
   ...:     # Long Journeys
   ...:     max_journ_len = 16
   ...:     df = df.groupby('journey_id').filter(lambda x: len(x) <= max_journ_len)
   ...:
   ...:     # Remove Columns
   ...:     df = df.drop(['s', 'timestamp_conversion', 'time_diff'], axis=1)  # cant be used for prediction
   ...:
   ...:     # How to handle object variables
   ...:     # Dummy variables for country, platform and channel, better than Ordinal, but also huge data
   ...:     df = pd.get_dummies(df, columns=['channel_id'], prefix='channel', prefix_sep='_', dtype=float)
   ...:     df = pd.get_dummies(df, columns=['country_name'], prefix='country', prefix_sep='_', dtype=float)
   ...:     df = pd.get_dummies(df, columns=['platform'], prefix='platform', prefix_sep='_', dtype=float)
   ...:
   ...:     return df
   ...:
   ...: # Next step: transform to tensor:
   ...:
   ...:
   ...: def mta2tensor(df):  # function that transforms dataset to tensor
   ...:     df_transaction = df['transaction']
   ...:     df = df.drop('transaction', axis=1)
   ...:     grous = df.groupby('journey_id')
   ...:     x = []
   ...:     y = []
   ...:
   ...:     for i in data['journey_id'].unique():
   ...:         x1 = grous.get_group(i)
   ...:
   ...:         x1 = x1.drop(['journey_id'], axis=1)
   ...:         x1 = x1.values.tolist()
   ...:
   ...:         y1 = df_transaction.loc[grous.get_group(i).index]
   ...:         y1 = y1.values.tolist()
   ...:
   ...:         for j in range(max_journ_len - len(x1)):
   ...:             # for-loop for data padding (all customer journeys filled with zeros to get same length)
   ...:             x1.append([0] * 52)  # 52 is number of columns without journey_id an transaction
   ...:             y1.append(0)
   ...:         x.append(x1)
   ...:         y.append(y1)
   ...:
   ...:     return tf.convert_to_tensor(x), tf.convert_to_tensor(y)
   ...:
   ...:
   ...: data = pd.read_csv("data_sample1.csv")
   ...: max_journ_len = 16
   ...: data = prep_df(data, max_journ_len)
   ...: x_train, y_train = mta2tensor(data)
   ...: print(x_train)
   ...:

```