eda

July 14, 2023

[1]:
```
# This file is modified version of original:
# https://www.kaggle.com/code/leonidkulyk/eda-pfogp-interactive-visualisations
```

#

PFOGP - Data and problem investigation

Event detection from wearable sensor data

[ ]:

#

(_) Overview

In this competition, we aim to detect freezing of gait (FOG) using data collected from a wearable 3D lower back sensor. Accelerometer data can be pretty fun and unique to work with. It is important to understand that this data is a time series. We are looking for Turning, StartHesitation, and Walking events in these time series.

FOG is a debilitating symptom that affects many people with Parkinson's disease, leading to restricted independence and increased risk of falls.

By developing a machine learning model trained on this dataset, we can improve the ability of medical professionals to evaluate, monitor, and prevent FOG events.

There are multiple methods of evaluating FOG, but most involve FOG-provoking protocols, which are time-consuming and require specific expertise.

Wearable devices can help detect FOG more easily, but compliance and usability may be reduced.

The datasets used to train and test machine learning algorithms for detecting FOG have been relatively small, and generalizability is limited to date.

The competition host, the Center for the Study of Movement, Cognition, and Mobility (CMCM), Neurological Institute, Tel Aviv Sourasky Medical Center, aims to improve the personalized treatment of age-related movement, cognition, and mobility disorders and to alleviate the associated burden.

Submissions are evaluated by the Mean Average Precision of predictions for each event class. Average precision is computed on predicted confidence scores separately for each of the three event classes.

#

Table of contents

#

0. Import all dependencies

[2]:
```
import os
import random
import cv2
import pandas as pd
import numpy as np
import plotly.express as px
import matplotlib.pyplot as plt
```

[3]:
```
class color:
   PURPLE = '\033[95m'
   CYAN = '\033[96m'
   DARKCYAN = '\033[36m'
   BLUE = '\033[94m'
   GREEN = '\033[92m'
   YELLOW = '\033[93m'
   RED = '\033[91m'
   BOLD = '\033[1m'
   UNDERLINE = '\033[4m'
   END = '\033[0m'
```

#

1. Overview availiable directories

The data series include three datasets, collected under distinct circumstances:

- The tDCS FOG (tdcsfog) dataset, comprising data series collected in the lab, as subjects completed a FOG-provoking protocol.
- The DeFOG (defog) dataset, comprising data series collected in the subject's home, as subjects completed a FOG-provoking protocol.

- The Daily Living (daily) dataset, comprising one week of continuous 24/7 recordings from sixty-five subjects. Forty-five subjects exhibit FOG symptoms and also have series in the defog dataset, while the other twenty subjects do not exhibit FOG symptoms and do not have series elsewhere in the data.

Trials from the tdcsfog and defog datasets were videotaped and annotated by expert reviewers documented the freezing of gait episodes. That is, the start, end and type of each episode were marked by the experts. Series in the daily dataset are unannotated.

You will be detecting FOG episodes for the tdcsfog and defog series.

See this page for more on these datasets as well as video examples of freezing of gait events: Additional Data Documentation.

##

1.1 Overview train/ directory

train/ Folder containing the data series in the training set within three subfolders: tdcsfog/, defog/, and notype/. Series in the notype folder are from the defog dataset but lack event-type annotations. The fields present in these series vary by folder.

- Time An integer timestep. Series from the tdcsfog dataset are recorded at 128Hz (128 timesteps per second), while series from the defog and daily series are recorded at 100Hz (100 timesteps per second).
- AccV, AccML, and AccAP Acceleration in units of g, from a lower-back sensor on three axes: V - vertical, ML - mediolateral, AP - anteroposterior.
- StartHesitation, Turn, Walking Indicator variables for the occurrence of each of the event types.
- Event Indicator variable for the occurrence of any FOG-type event. Present only in the notype series, which lack type-level annotations.
- Valid Specifies whether the event annotations is unambiguous. There were cases during the video annotation that were hard for the annotator to decide if there was an Akinetic (i.e., essentially no movement) FoG or the subject stopped voluntarily. Only event annotations where the series is marked true should be considered as unambiguous.
- Task Specifies whether the event was annotated. Series were only annotated where this value is true. Portions marked false should be considered unannotated.

let's check what directories are in the train folder.

[4]:
```
os.listdir("/kaggle/input/tlvmc-parkinsons-freezing-gait-prediction/train")
```

[4]:
```
['defog', 'tdcsfog', 'notype']
```

How many files are in folder tdcsfog/.

[5]:
```
temp = len(os.listdir("/kaggle/input/tlvmc-parkinsons-freezing-gait-prediction/
 ↪train/tdcsfog"))
print(
```

```
    f"Number of files in folder tdcsfog/: {color.BLUE}{temp}{color.END}",
)
```

```
Number of files in folder tdcsfog/: 833
```

How the data looks like.

[6]:
```
train_tdcsfog_example_df = pd.read_csv("/kaggle/input/
  ↪tlvmc-parkinsons-freezing-gait-prediction/train/tdcsfog/003f117e14.csv")
```

[7]:
```
temp = len(train_tdcsfog_example_df)
print(
    f"Length of dataframe: {color.BLUE}{temp}{color.END}",
)
```

```
Length of dataframe: 4682
```

[8]:
```
train_tdcsfog_example_df.head()
```

[8]:

	Time	AccV	AccML	AccAP	StartHesitation	Turn	Walking
0	0	-9.533939	0.566322	-1.413525	0	0	0
1	1	-9.536140	0.564137	-1.440621	0	0	0
2	2	-9.529345	0.561765	-1.429332	0	0	0
3	3	-9.531239	0.564227	-1.415490	0	0	0
4	4	-9.540825	0.561854	-1.429471	0	0	0

[9]:
```
train_tdcsfog_example_df.describe()
```

[9]:

	Time	AccV	AccML	AccAP	StartHesitation	\
count	4682.00000	4682.000000	4682.000000	4682.000000	4682.0	
mean	2340.50000	-9.151214	0.753518	2.471637	0.0	
std	1351.72131	1.384390	1.102125	2.239906	0.0	
min	0.00000	-23.796051	-9.097370	-7.353417	0.0	
25%	1170.25000	-9.537719	0.322877	1.966646	0.0	
50%	2340.50000	-9.234702	0.580891	3.137857	0.0	
75%	3510.75000	-8.470460	1.368355	3.819931	0.0	
max	4681.00000	-3.915590	5.996704	10.281080	0.0	

	Turn	Walking
count	4682.000000	4682.0
mean	0.168304	0.0
std	0.374176	0.0
min	0.000000	0.0
25%	0.000000	0.0
50%	0.000000	0.0
75%	0.000000	0.0
max	1.000000	0.0

[10]:
```
for c in train_tdcsfog_example_df:
    print(c)
    break
```

```
Time
```

[11]:
```
# unique values in each column
pd.Series({c: train_tdcsfog_example_df[c].unique() for c in␣
 ↪train_tdcsfog_example_df})
```

[11]:
```
Time              [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,…
AccV              [-9.53393930253288, -9.53614029997918, -9.5293…
AccML             [0.566321631981499, 0.564136952175035, 0.56176…
AccAP             [-1.41352531246173, -1.4406209993301, -1.42933…
StartHesitation                                              [0]
Turn                                                      [0, 1]
Walking                                                      [0]
dtype: object
```

Is there any NaN values in the dataframe.

[12]:
```
train_tdcsfog_example_df.isnull().sum()
```

[12]:
```
Time              0
AccV              0
AccML             0
AccAP             0
StartHesitation   0
Turn              0
Walking           0
dtype: int64
```

How many files are in folder defog/.

[13]:
```
temp = len(os.listdir("/kaggle/input/tlvmc-parkinsons-freezing-gait-prediction/
 ↪train/defog"))
print(
    f"Number of files in folder defog/: {color.BLUE}{temp}{color.END}",
)
```

```
Number of files in folder defog/: 91
```

How many files are in folder notype/.

[14]:
```
temp = len(os.listdir("/kaggle/input/tlvmc-parkinsons-freezing-gait-prediction/
 ↪train/notype"))
print(
    f"Number of files in folder notype/: {color.BLUE}{temp}{color.END}",
```

```
)
```

```
Number of files in folder notype/: 46
```

How is the time sirieses for features 'AccV','AccML','AccAP' looks like.

[15]:
```
for column in ['AccV','AccML','AccAP']:
    fig = px.line(train_tdcsfog_example_df, x="Time", y=column,␣
 ↪color_discrete_sequence=['darkslateblue'])
    fig.update_layout(
        title={
            'text': f"{column} Time Series",
            'y':0.95,
            'x':0.5,
            'xanchor': 'center',
            'yanchor': 'top'
        }
    )
    fig.show()
```

How is the correlation matrix for the data looks like.

[16]:
```
corr_mat = np.round(train_tdcsfog_example_df.drop(
    ["StartHesitation", "Walking"], axis=1
).corr(), 3)

fig = px.imshow(
    corr_mat,
    x=corr_mat.columns,
    y=corr_mat.columns,
    text_auto=True
   )
fig.update_xaxes(side="bottom")
fig.update_layout(
    title={
        'text': "Correlation Matrix of tdcsfog train instance",
        'y':0.95,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    }
)

fig.show()
```

##

1.2 Overview test/ directory

test/ Only the Time, AccV, AccML, and AccAP fields are provided for the test series.

[17]:
```
os.listdir("/kaggle/input/tlvmc-parkinsons-freezing-gait-prediction/test")
```

[17]:
```
['defog', 'tdcsfog']
```

How many files are in folder tdcsfog/.

[18]:
```
temp = len(os.listdir("/kaggle/input/tlvmc-parkinsons-freezing-gait-prediction/
 ↪test/tdcsfog"))
print(
    f"Number of files in folder tdcsfog/: {color.BLUE}{temp}{color.END}",
)
```

```
Number of files in folder tdcsfog/: 1
```

How many files are in folder defog/.

[19]:
```
temp = len(os.listdir("/kaggle/input/tlvmc-parkinsons-freezing-gait-prediction/
 ↪test/defog"))
print(
    f"Number of files in folder defog/: {color.BLUE}{temp}{color.END}",
)
```

```
Number of files in folder defog/: 1
```

How the test data looks like.

[20]:
```
test_tdcsfog_example_df = pd.read_csv("/kaggle/input/
 ↪tlvmc-parkinsons-freezing-gait-prediction/test/tdcsfog/003f117e14.csv")
```

[21]:
```
temp = len(test_tdcsfog_example_df)
print(
    f"Length of dataframe: {color.BLUE}{temp}{color.END}",
)
```

```
Length of dataframe: 4682
```

[22]:
```
test_tdcsfog_example_df.head()
```

[22]:

	Time	AccV	AccML	AccAP
0	0	-9.533939	0.566322	-1.413525
1	1	-9.536140	0.564137	-1.440621
2	2	-9.529345	0.561765	-1.429332
3	3	-9.531239	0.564227	-1.415490
4	4	-9.540825	0.561854	-1.429471

[23]:
```
test_tdcsfog_example_df.describe()
```

[23]:

	Time	AccV	AccML	AccAP
count	4682.00000	4682.000000	4682.000000	4682.000000

```
mean   2340.50000    -9.151214     0.753518     2.471637
std    1351.72131     1.384390     1.102125     2.239906
min       0.00000   -23.796051    -9.097370    -7.353417
25%    1170.25000    -9.537719     0.322877     1.966646
50%    2340.50000    -9.234702     0.580891     3.137857
75%    3510.75000    -8.470460     1.368355     3.819931
max    4681.00000    -3.915590     5.996704    10.281080
```

How is the correlation matrix for the data looks like.

[24]:
```
corr_mat = np.round(test_tdcsfog_example_df.corr(), 3)

fig = px.imshow(
    corr_mat,
    x=corr_mat.columns,
    y=corr_mat.columns,
    text_auto=True
   )
fig.update_xaxes(side="bottom")
fig.update_layout(
    title={
        'text': "Correlation Matrix of tdcsfog test instance",
        'y':0.95,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    }
)

fig.show()
```

##

1.3 Overview unlabeled/ directory

unlabeled/ Folder containing the unannotated data series from the daily dataset, one series per subject. Forty-five of the subjects also have series in the defog dataset, some in the training split and some in the test split.

How many files are in folder unlabeled/.

[25]:
```
temp = len(os.listdir("/kaggle/input/tlvmc-parkinsons-freezing-gait-prediction/
 ↪unlabeled"))
print(
    f"Number of files in folder unlabeled/: {color.BLUE}{temp}{color.END}",
)
```

```
Number of files in folder unlabeled/: 65
```

How the data looks like.

[26]:
```
unlabeled_example_df = pd.read_parquet("/kaggle/input/
  ↪tlvmc-parkinsons-freezing-gait-prediction/unlabeled/00c4c9313d.parquet")
```

[27]:
```
unlabeled_example_df.head()
```

[27]:

	Time	AccV	AccML	AccAP
0	0	0.328125	-0.109375	0.671875
1	1	0.453108	-0.124721	0.811273
2	2	0.423042	-0.264046	0.921238
3	3	0.150015	-0.310241	0.937483
4	4	-0.202003	-0.545908	0.890842

[28]:
```
unlabeled_example_df.describe()
```

[28]:

	Time	AccV	AccML	AccAP
count	6.972239e+07	6.972239e+07	6.972239e+07	6.972239e+07
mean	3.486119e+07	-5.277971e-01	-8.455515e-02	-4.769068e-02
std	2.012712e+07	4.339047e-01	4.878394e-01	5.178868e-01
min	0.000000e+00	-7.035982e+00	-6.183435e+00	-6.459452e+00
25%	1.743060e+07	-9.531250e-01	-1.875000e-01	-3.281250e-01
50%	3.486119e+07	-5.937500e-01	-1.562500e-02	-6.250000e-02
75%	5.229179e+07	-7.835454e-02	1.253366e-01	2.291177e-01
max	6.972239e+07	2.936576e+00	6.082396e+00	7.138464e+00

How is the correlation matrix for the data looks like.

[29]:
```
corr_mat = np.round(unlabeled_example_df.corr(), 3)

fig = px.imshow(
    corr_mat,
    x=corr_mat.columns,
    y=corr_mat.columns,
    text_auto=True
   )
fig.update_xaxes(side="bottom")
fig.update_layout(
    title={
        'text': "Correlation Matrix of unlabeled instance",
        'y':0.95,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    }
)

fig.show()
```

#

2. Overview availiable csv files

##

2.1 Overview tdcsfog_metadata.csv file

tdcsfog_metadata.csv Identifies each series in the tdcsfog dataset by a unique Subject, Visit, Test, Medication condition.

- Visit Lab visits consist of a baseline assessment, two post-treatment assessments for different treatment stages, and one follow-up assessment.
- Test Specifies of which of three test types was performed, from easy (1) to hard (3) (for the tdcsfog_df dataframe).
- Medication Subjects may have been either off or on anti-parkinsonian medication during the recording.

How the data looks like.

[30]:
```
tdcsfog_metadata_df = pd.read_csv("/kaggle/input/
↪tlvmc-parkinsons-freezing-gait-prediction/tdcsfog_metadata.csv")
```

[31]:
```
tdcsfog_metadata_df.head()
```

[31]:

	Id	Subject	Visit	Test	Medication
0	003f117e14	4dc2f8	3	2	on
1	009ee11563	f62eec	4	2	on
2	011322847a	231c3b	2	2	on
3	01d0fe7266	231c3b	2	1	off
4	024418ba39	fa8764	19	3	on

[32]:
```
tdcsfog_metadata_df.describe()
```

[32]:

	Visit	Test
count	833.000000	833.000000
mean	6.460984	1.974790
std	6.171914	0.813402
min	2.000000	1.000000
25%	2.000000	1.000000
50%	4.000000	2.000000
75%	5.000000	3.000000
max	20.000000	3.000000

What is length of the dataframe.

[33]:
```
temp = len(tdcsfog_metadata_df)
print(
    f"Length of the tdcsfog_metadata.csv file is: {color.BLUE}{temp}{color.
↪END}",
)
```

```
Length of the tdcsfog_metadata.csv file is: 833
```

How many unique subjects the dataframe has.

[34]:
```
temp = len(tdcsfog_metadata_df.Subject.unique())
print(
    f"Number of unique subjects: {color.BLUE}{temp}{color.END}",
)
```

```
Number of unique subjects: 62
```

How the data for an unique subject looks like.

[35]:
```
unique_subject_id = "13abfd"
tdcsfog_metadata_df[tdcsfog_metadata_df.Subject == unique_subject_id]
```

[35]:
```
Empty DataFrame
Columns: [Id, Subject, Visit, Test, Medication]
Index: []
```

Is there any missing data in the dataframe.

[36]:
```
tdcsfog_metadata_df.isnull().sum()
```

[36]:
```
Id            0
Subject       0
Visit         0
Test          0
Medication    0
dtype: int64
```

How the bar chart looks like for categorical field Visit.

[37]:
```
tdcsfog_visit_counts = tdcsfog_metadata_df.Visit.value_counts()

fig = px.bar(x=tdcsfog_visit_counts.index, y=tdcsfog_visit_counts.values,␣
 ↪color_discrete_sequence=['darkgreen'])
fig.update_layout(xaxis_title="Visit", yaxis_title="Count")
fig.show()
```

How the bar chart looks like for categorical field Test.

[38]:
```
tdcsfog_test_counts = tdcsfog_metadata_df.Test.value_counts()

fig = px.bar(x=tdcsfog_test_counts.index, y=tdcsfog_test_counts.values,␣
 ↪color_discrete_sequence=['darkgreen'])
fig.update_layout(xaxis_title="Test", yaxis_title="Count")
fig.show()
```

How the bar chart looks like for categorical field Medication.

[39]:
```
tdcsfog_medication_counts = tdcsfog_metadata_df.Medication.value_counts()

fig = px.bar(x=tdcsfog_medication_counts.index, y=tdcsfog_medication_counts.
 ↪values, color_discrete_sequence=['darkgreen'])
fig.update_layout(xaxis_title="Medication", yaxis_title="Count")
fig.show()
```

##

2.2 Overview defog_metadata.csv file

defog_metadata.csv Identifies each series in the defog dataset by a unique Subject, Visit, Medication condition.

How the data looks like.

[40]:
```
defog_metadata_df = pd.read_csv("/kaggle/input/
 ↪tlvmc-parkinsons-freezing-gait-prediction/defog_metadata.csv")
```

[41]:
```
defog_metadata_df.head()
```

[41]:

	Id	Subject	Visit	Medication
0	02ab235146	e1f62e	2	on
1	02ea782681	ae2d35	2	on
2	06414383cf	8c1f5e	2	off
3	092b4c1819	2874c5	1	off
4	0a900ed8a2	0e3d49	2	on

Is there any missing data.

[42]:
```
defog_metadata_df.isnull().sum()
```

[42]:
```
Id            0
Subject       0
Visit         0
Medication    0
dtype: int64
```

What is length of the dataframe.

[43]:
```
temp = len(defog_metadata_df)
print(
    f"Length of the defog_metadata.csv file is: {color.BLUE}{temp}{color.END}",
)
```

```
Length of the defog_metadata.csv file is: 137
```

[44]:
```
temp = len(defog_metadata_df.Subject.unique())
print(
    f"Number of unique subjects: {color.BLUE}{temp}{color.END}",
```

```
)
```

```
Number of unique subjects: 45
```

How the data for an unique subject looks like.

[45]:
```
unique_subject_id = "bf608b"
defog_metadata_df[defog_metadata_df.Subject == unique_subject_id]
```

[45]:
```
Empty DataFrame
Columns: [Id, Subject, Visit, Medication]
Index: []
```

How the bar chart looks like for categorical field Visit.

[46]:
```
defog_visit_counts = defog_metadata_df.Visit.value_counts()

fig = px.bar(x=defog_visit_counts.index, y=defog_visit_counts.values,␣
 ↪color_discrete_sequence=['darkslateblue'])
fig.update_layout(xaxis_title="Visit", yaxis_title="Count")
fig.show()
```

How the bar chart looks like for categorical field Medication.

[47]:
```
defog_medication_counts = defog_metadata_df.Medication.value_counts()

fig = px.bar(x=defog_medication_counts.index, y=defog_medication_counts.values,␣
 ↪color_discrete_sequence=['darkslateblue'])
fig.update_layout(xaxis_title="Medication", yaxis_title="Count")
fig.show()
```

##

2.3 Overview daily_metadata.csv file

daily_metadata.csv Each series in the daily dataset is identified by the Subject id. This file also contains the time of day the recording begin.

How the data looks like.

[48]:
```
daily_metadata_df = pd.read_csv("/kaggle/input/
 ↪tlvmc-parkinsons-freezing-gait-prediction/daily_metadata.csv")
```

[49]:
```
daily_metadata_df.head()
```

[49]:

	Id	Subject	Visit	Beginning of recording [00:00-23:59]
0	00c4c9313d	fba3a3	1	10:19
1	07a96f89ec	7da72f	1	07:30
2	0d1bc672a8	056372	2	08:30
3	0e333c9833	b4bd22	1	11:30

```
4  164adaed7b  9f72eb      1                                  13:00
```

Is there any missing data.

[50]:
```
daily_metadata_df.isnull().sum()
```

[50]:
```
Id                                      0
Subject                                 0
Visit                                   0
Beginning of recording [00:00-23:59]    0
dtype: int64
```

What is length of the dataframe.

[51]:
```
temp = len(daily_metadata_df)
print(
    f"Length of the daily_metadata.csv file is: {color.BLUE}{temp}{color.END}",
)
```

```
Length of the daily_metadata.csv file is: 65
```

[52]:
```
temp = len(daily_metadata_df.Subject.unique())
print(
    f"Number of unique subjects: {color.BLUE}{temp}{color.END}",
)
```

```
Number of unique subjects: 65
```

How the bar chart looks like for categorical field Visit.

[53]:
```
daily_visit_counts = daily_metadata_df.Visit.value_counts()

fig = px.bar(x=daily_visit_counts.index, y=daily_visit_counts.values,␣
 ↪color_discrete_sequence=['cornflowerblue'])
fig.update_layout(xaxis_title="Visit", yaxis_title="Count")
fig.show()
```

How the bar chart looks like for the field Beginning of recording.

[54]:
```
daily_bor_counts = daily_metadata_df["Beginning of recording [00:00-23:59]"].
 ↪value_counts()

fig = px.bar(x=daily_bor_counts.index, y=daily_bor_counts.values,␣
 ↪color_discrete_sequence=['cornflowerblue'])
fig.update_layout(xaxis_title="Beginning of recording", yaxis_title="Count")
fig.show()
```

##

2.4 Overview subjects.csv file

subjects.csv Metadata for each Subject in the study, including their Age and Sex as well as:

- Visit Only available for subjects in the daily and defog datasets.
- YearsSinceDx Years since Parkinson's diagnosis.
- UPDRSIIIOn/UPDRSIIIOff Unified Parkinson's Disease Rating Scale score during on/off medication respectively.
- NFOGQ Self-report FoG questionnaire score.

How the data looks like.

[55]:
```
subjects_df = pd.read_csv("/kaggle/input/
↪tlvmc-parkinsons-freezing-gait-prediction/subjects.csv")
```

[56]:
```
subjects_df.head()
```

[56]:

	Subject	Visit	Age	Sex	YearsSinceDx	UPDRSIII_On	UPDRSIII_Off	NFOGQ
0	00f674	2.0	63	M	27.0	43.0	49.0	24
1	00f674	1.0	63	M	27.0	31.0	30.0	26
2	02bc69	NaN	69	M	4.0	21.0	NaN	22
3	040587	2.0	75	M	26.0	52.0	69.0	21
4	040587	1.0	75	M	26.0	47.0	75.0	24

Is there any missing data.

[57]:
```
subjects_df.isnull().sum()
```

[57]:
```
Subject          0
Visit           62
Age              0
Sex              0
YearsSinceDx     0
UPDRSIII_On      1
UPDRSIII_Off    41
NFOGQ            0
dtype: int64
```

What is length of the dataframe.

[58]:
```
temp = len(subjects_df)
print(
    f"Length of the subjects.csv file is: {color.BLUE}{temp}{color.END}",
)
```

```
Length of the subjects.csv file is: 173
```

[59]:
```
temp = len(subjects_df.Subject.unique())
print(
    f"Number of unique subjects: {color.BLUE}{temp}{color.END}",
```

```
)
```

```
Number of unique subjects: 136
```

How the bar chart looks like for categorical field Visit.

[60]:
```
subjects_visit_counts = subjects_df.Visit.value_counts()

fig = px.bar(x=subjects_visit_counts.index, y=subjects_visit_counts.values,␣
 ↪color_discrete_sequence=['goldenrod'])
fig.update_layout(xaxis_title="Visit", yaxis_title="Count")
fig.show()
```

What is the distibution of the field Age.

[61]:
```
fig = px.histogram(subjects_df, x="Age", nbins=30,␣
 ↪color_discrete_sequence=['goldenrod'])
fig.show()
```

How the bar chart looks like for categorical field Sex.

[62]:
```
subjects_sex_counts = subjects_df.Sex.value_counts()

fig = px.bar(x=subjects_sex_counts.index, y=subjects_sex_counts.values,␣
 ↪color_discrete_sequence=['goldenrod'])
fig.update_layout(xaxis_title="Sex", yaxis_title="Count")
fig.show()
```

What is the distibution of the field YearsSinceDx.

[63]:
```
fig = px.histogram(subjects_df, x="YearsSinceDx", nbins=30,␣
 ↪color_discrete_sequence=['goldenrod'])
fig.show()
```

What is the distibution of the field UPDRSIII_On.

[64]:
```
fig = px.histogram(subjects_df, x="UPDRSIII_On", nbins=30,␣
 ↪color_discrete_sequence=['goldenrod'])
fig.show()
```

What is the distibution of the field NFOGQ.

[65]:
```
fig = px.histogram(subjects_df, x="NFOGQ", nbins=20,␣
 ↪color_discrete_sequence=['goldenrod'])
fig.show()
```

How is the correlation matrix for the data looks like.

[66]:
```
corr_mat = np.round(subjects_df.corr(), 3)

fig = px.imshow(
    corr_mat,
    x=corr_mat.columns,
    y=corr_mat.columns,
    text_auto=True
   )
fig.update_xaxes(side="bottom")
fig.update_layout(
    title={
        'text': "Correlation Matrix of sybjects features",
        'y':0.95,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    }
)

fig.show()
```

##

2.5 Overview events.csv file

events.csv Metadata for each FoG event in all data series. The event times agree with the labels in the data series.

- Visit The data series the event occured in.
- Init Time (s) the event began.
- Completion Time (s) the event ended.
- Type Whether StartHesitation, Turn, or Walking.
- Kinetic Whether the event was kinetic (1) and involved movement, or akinetic (0) and static.

How the data looks like.

[67]:
```
events_df = pd.read_csv("/kaggle/input/
 ↪tlvmc-parkinsons-freezing-gait-prediction/events.csv")
```

[68]:
```
events_df.head()
```

[68]:

	Id	Init	Completion	Type	Kinetic
0	003f117e14	8.61312	14.7731	Turn	1.0
1	009ee11563	11.38470	41.1847	Turn	1.0
2	009ee11563	54.66470	58.7847	Turn	1.0
3	011322847a	28.09660	30.2966	Turn	1.0
4	01d0fe7266	30.31840	31.8784	Turn	1.0

What is length of the dataframe.

[69]:
```
temp = len(events_df)
print(
    f"Length of the events.csv file is: {color.BLUE}{temp}{color.END}",
)
```

```
Length of the events.csv file is: 3544
```

Is there any missing data.

[70]:
```
events_df.isnull().sum()
```

[70]:
```
Id              0
Init            0
Completion      0
Type         1042
Kinetic      1042
dtype: int64
```

What is the distibution of the field Init.

[71]:
```
fig = px.histogram(events_df, x="Init", nbins=50)
fig.show()
```

What is the distibution of the difference if fields Init and Completion.

[72]:
```
event_time = events_df.Completion - events_df.Init
fig = px.histogram(x=event_time, nbins=30)
fig.update_layout(
    xaxis_title="Event duration",
    title={
        'text': "Distribution of event durations",
        'y':0.95,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    }
)
fig.show()
```

How the bar chart looks like for categorical field Type.

[73]:
```
events_type_counts = events_df.Type.value_counts()

fig = px.bar(x=events_type_counts.index, y=events_type_counts.values)
fig.update_layout(xaxis_title="Type", yaxis_title="Count")
fig.show()
```

How the bar chart looks like for categorical field Kinetic.

[74]:
```
events_kinetic_counts = events_df.Kinetic.value_counts()

fig = px.bar(x=events_kinetic_counts.index, y=events_kinetic_counts.values)
fig.update_layout(xaxis_title="Kinetic", yaxis_title="Count")
fig.show()
```

How is the correlation matrix for the data looks like.

[75]:
```
corr_mat = np.round(events_df.corr(), 3)

fig = px.imshow(
    corr_mat,
    x=corr_mat.columns,
    y=corr_mat.columns,
    text_auto=True
   )
fig.update_xaxes(side="bottom")
fig.update_layout(
    title={
        'text': "Correlation Matrix of events features",
        'y':0.95,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    }
)

fig.show()
```

##

2.6 Overview tasks.csv file

tasks.csv Task metadata for series in the defog dataset. (Not relevant for the series in the fog or daily datasets.)

- Id The data series where the task was measured.
- Begin Time (s) the task began.
- End Time (s) the task ended.
- Task One of seven tasks types in the DeFOG protocol, described on this page.
- Description Description of the task.

How the data looks like.

[76]:
```
tasks_df = pd.read_csv("/kaggle/input/tlvmc-parkinsons-freezing-gait-prediction/
↪tasks.csv")
```

[77]:
```
tasks_df.head()
```

[77]:

	Id	Begin	End	Task
0	02ab235146	10.00	190.48	Rest1
1	02ab235146	211.24	271.56	Rest2
2	02ab235146	505.88	522.40	4MW
3	02ab235146	577.96	594.64	4MW-C
4	02ab235146	701.32	715.28	MB1

What is length of the dataframe.

[78]:
```
temp = len(tasks_df)
print(
    f"Length of the subjects.csv file is: {color.BLUE}{temp}{color.END}",
)
```

```
Length of the subjects.csv file is: 2817
```

Is there any missing data.

[79]:
```
tasks_df.isnull().sum()
```

[79]:
```
Id       0
Begin    0
End      0
Task     0
dtype: int64
```

What is the distibution of the difference between fields Begin and End.

[80]:
```
tasks_df["Duration"] = tasks_df.End - tasks_df.Begin
fig = px.histogram(x=tasks_df["Duration"], nbins=30,␣
 ↪color_discrete_sequence=['lightslategrey'])
fig.update_layout(
    xaxis_title="Task duration",
    title={
        'text': "Distribution of task durations",
        'y':0.95,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    }
)
fig.show()
```

How the bar chart looks like for categorical field Task.

[81]:
```
tasks_task_counts = tasks_df.Task.value_counts()

fig = px.bar(x=tasks_task_counts.index, y=tasks_task_counts.values,␣
 ↪color_discrete_sequence=['lightslategrey'])
```

```
fig.update_layout(xaxis_title="Task", yaxis_title="Count")
fig.show()
```

How the boxplot looks like for each task on duration.

[82]:
```
fig = px.box(
    tasks_df,
    x='Duration', y='Task',
    orientation='h', height=1000,
    color_discrete_sequence=['lightslategrey']
)

fig.update_layout(
    title={
        'text': "Task Duration Boxplot",
        'y':0.98,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'
    },
    margin=dict(l=50, r=0, t=50, b=50)
)
fig.show(autosize=True)
```

[83]:
```
# (   ) WORK STILL IN PROGRESS
```

#

Thank You!

Thank you for taking the time to read through my notebook. I hope you found it interesting and informative. If you have any feedback or suggestions for improvement, please don't hesitate to let me know in the comments. If you liked this notebook, please consider upvoting it so that others can discover it too. Your support means a lot to me, and it helps to motivate me to create more content in the future. Once again, thank you for your support, and I hope to see you again soon!