Homework 0 Part 2

This is an individual assignment.

Due: Thursday, September 8 @ 11:59pm

Custom Transformers

`scikit-learn` offers many useful transformers, but you will likely need to write your own transformers for tasks such as custom cleanup operations or combining specific attributes.

To have your transformer working seamlessly with `scikit-learn`, all you need to do is create a class and implement three methods: `fit()` (returning self), `transform()`, and `fit_transform()`.

You can get the last one for free by simply adding `TransformerMixin` as your base class. If you add `BaseEstimator` as a base class (and avoid `*args` and `**kargs` you will also get two extra methods (`get_params()` and `set_params()`) that will be useful for automatic hyperparameter tuning.

For example, for the dataset we discussed in lecture 3 (California housing prices), we can build a transformer to add the custom attributes as follows:

```
from sklearn.base import BaseEstimator, TransformerMixin

rooms_ix, bedrooms_ix, population_ix, households_ix = 3, 4, 5, 6

class CombinedAttributesAdder(BaseEstimator, TransformerMixin):
    def __init__ (self, add_bedrooms_per_room=True): # no *args or
**kargs
        self.add_bedrooms_per_room = add_bedrooms_per_room
    def fit(self, X, y=None):
        return self # nothing else to do
    def transform(self, X):
        rooms_per_household = X[:, rooms_ix] / X[:, households_ix]
        population_per_household = X[:, population_ix] / X[:,
households_ix]
        if self.add_bedrooms_per_room:
            bedrooms_per_room = X[:, bedrooms_ix] / X[:, rooms_ix]
        return np.c_[X, rooms_per_household,
population_per_household, bedrooms_per_room]
```

This transformer can later be added to a `scikit-learn` pipeline like this:

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
```

```
num_pipeline = Pipeline([('imputer',
SimpleImputer(strategy='median')),
                         ('attribs_adder',
CombinedAttributesAdder()),
                         ('std_scaler', StandardScaler())])

housing_num_tr = num_pipeline.fit_transform(housing_num)
```

Question 1 (7.5 points)

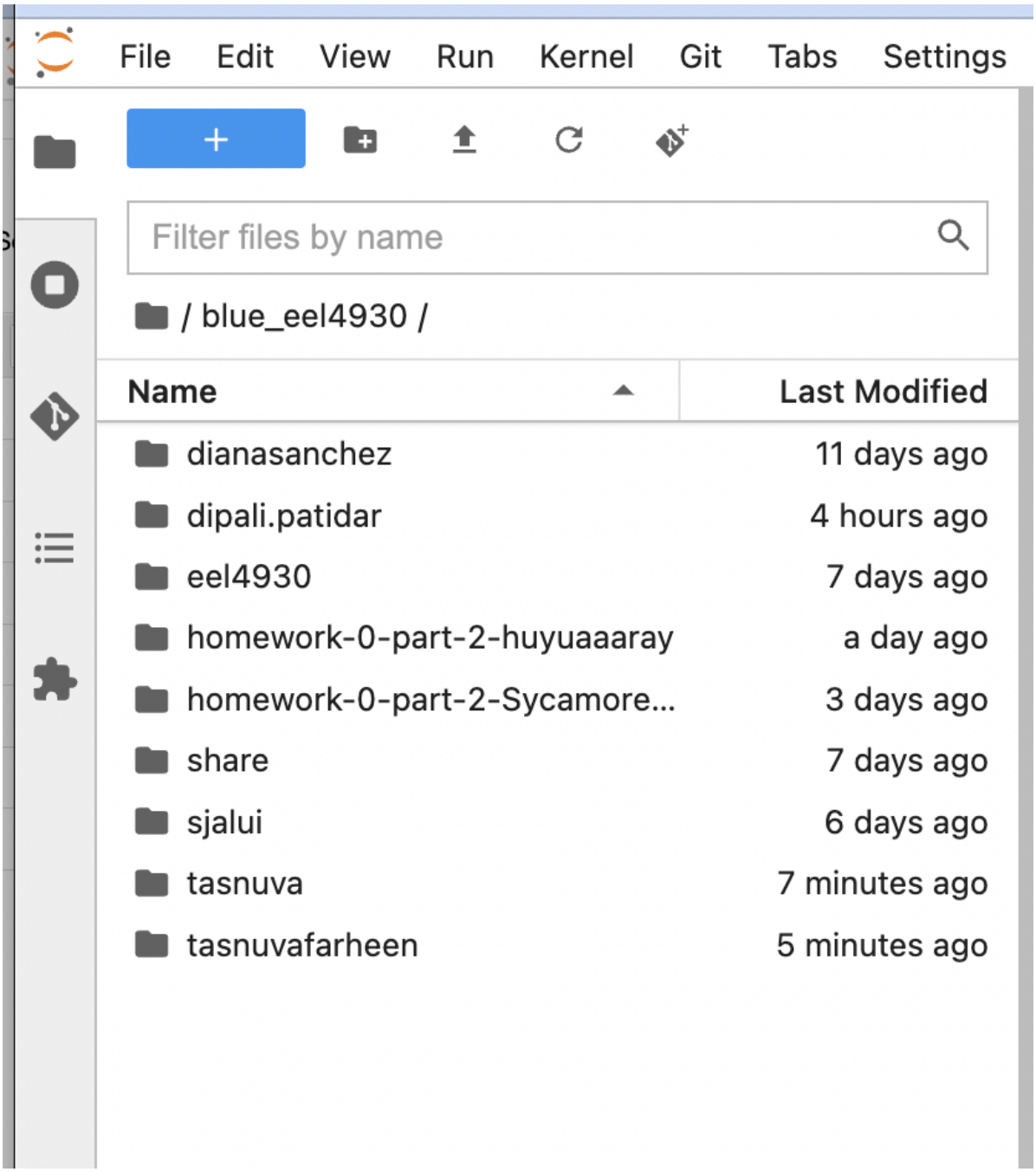
In this question, you will practice how to use HiPerGator and Git to maintain your code.

1. Open Open On-Demand ood.rc.ufl.edu and create an interactive jupyter session with the following specifications: 1 CPU, 1 GPU (type = 'a100') and 4 GB of RAM.

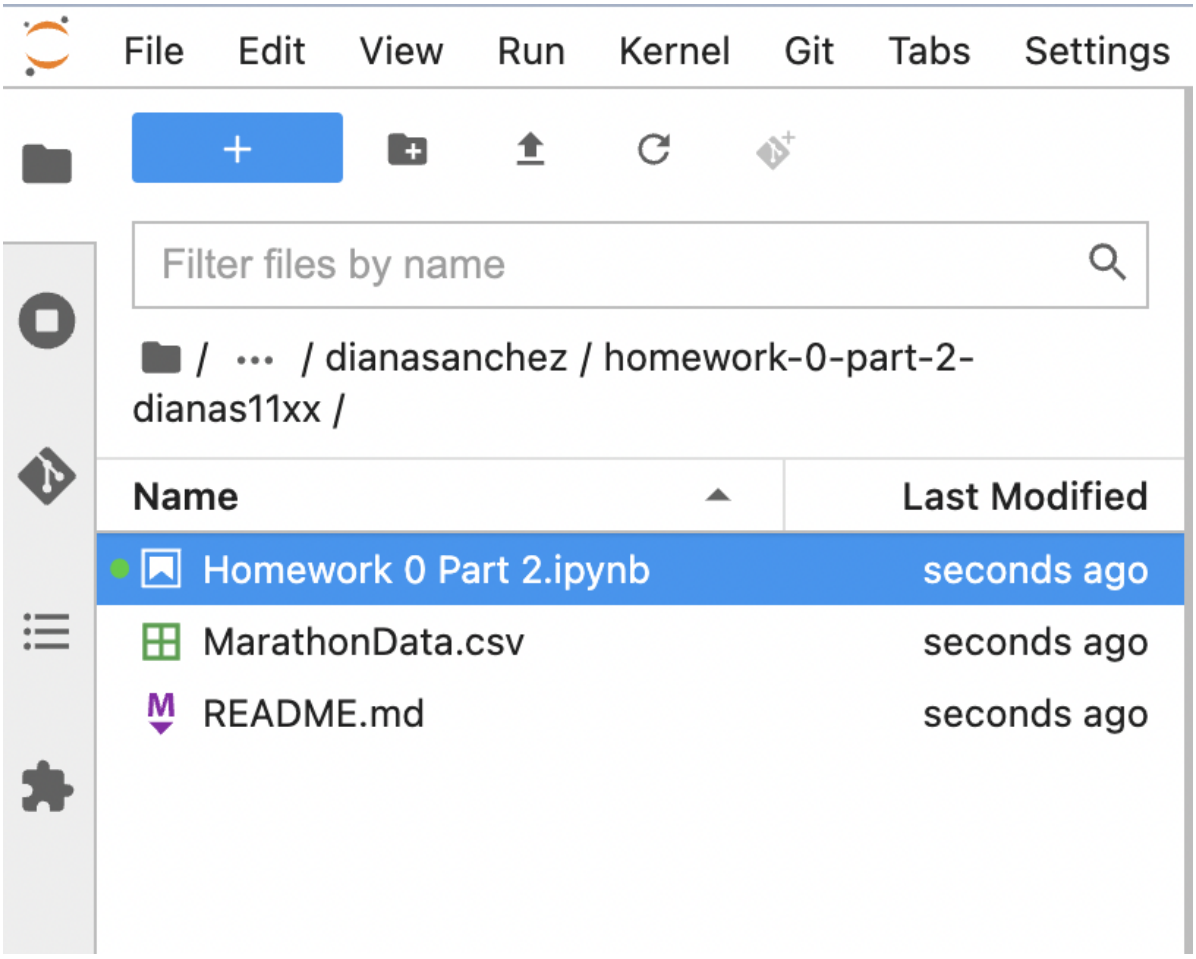
2. Create a symbolic link to map the class blue directory in your homepage. Attach a screenshot to show that this link has been created.

3. Navigate to your folder within the symbolic link you just created. Clone HW0-P2 in that folder. Attach a screenshot.

2.


File
Edit
View
Run
Kernel
Git
Tabs
Settings
Filter files by name
/ blue_eel4930 /
Name
Last Modified
dianasanchez
11 days ago
dipali.patidar
4 hours ago
eel4930
7 days ago
homework-0-part-2-huyuaaaray
a day ago
homework-0-part-2-Sycamore...
3 days ago
share
7 days ago
sjalui
6 days ago
tasnuva
7 minutes ago
tasnuvafarheen
5 minutes ago

3.



Question 2 (40 points)

In this question, you will be working with the marathon time predictions dataset.

Attributes

- **id**: simple counter
- **Marathon**: the Marathon name where the data were extracted.
- **Name**: The athlete's name.
- **Category**: the sex and age group of a runner.
  - MAM Male Athletes under 40 years
  - WAM Women under 40 Years
  - M40 Male Athletes between 40 and 45 years
- **km4week**: This is the total number of kilometers run in the last 4 weeks before the marathon, marathon included. If, for example, the km4week is 100, the athlete has run 400 km in the four weeks before the marathon.

- **sp4week**: This is the average speed of the athlete in the last 4 training weeks. The average counts all the kilometers done, included the slow kilometers done before and after the training. A typical running session can be of 2km of slow running, then 12-14km of fast running, and finally other 2km of slow running. The average of the speed is this number, and with time this is one of the numbers that has to be refined.
- **cross training**: If the runner is also a cyclist, or a triathlete.
- **Wall21**: To acknowledge a good performance, as a marathoner, the first half marathon should be run with the same split of the second half. If, for example, I run the first half marathon in 1h30m, I must finish the marathon in 3h (for doing a good job). If I finish in 3h20m, I started too fast and I hit "the wall". My training history is, therefore, less valid, since I was not estimating my result.
- **Marathon time**: This is target value.
- **Category**: Categorical encoding of the target value. It groups in:
  - A results under 3h
  - B results between 3h and 3h20m
  - C results between 3h20m and 3h40m
  - D results between 3h40 and 4h

Answer the following questions:

1. Load the data with `pandas`.
2. Are there missing samples? Determine which approach you are using to handle missing data. All justifications and reasoning must be included in a markdown cell.
3. Print the Pearson's correlation matrix and visualize the data. Which attribute has the largest predictive value to predict the marathon time?
4. Create a new attribute that computes the ratio between the `km4week` and `sp4week`. FBuild a custom transformer to automate this feature extraction (see example at the top of this notebook).
5. Partition the data into training and test sets. If you use stratified partition, justify your answer.
6. Build a transformation pipeline with `scikit-learn` to encode any categorical attributes.
7. Build a transformation pipeline with `scikit-learn` to encode the numerical attributes. This pipeline should include your custom transformer built in step 4.
8. Build a transformation pipeline that combines both categorical and numerical attributes.

P1)

In [596…
```
import pandas as pd
marathon_data = pd.read_csv("MarathonData.csv");
marathon_data
```

Out[596]:

	id	Marathon	Name	Category	km4week	sp4week	CrossTraining	Wall21	Maratho
0	1	Prague17	Blair MORGAN	MAM	132.8	14.434783	NaN	1.16	
1	2	Prague17	Robert Heczko	MAM	68.6	13.674419	NaN	1.23	
2	3	Prague17	Michon Jerome	MAM	82.7	13.520436	NaN	1.30	
3	4	Prague17	Daniel Or lek	M45	137.5	12.258544	NaN	1.32	
4	5	Prague17	Luk ? Mr zek	MAM	84.6	13.945055	NaN	1.36	
...	...	...	...	...	...	...	...	...	...
82	83	Prague17	Stefano Vegliani	M55	50.0	10.830325	NaN	2.02	
83	84	Prague17	Andrej Madliak	M40	33.6	10.130653	ciclista 3h	1.94	
84	85	Prague17	Yoi Ohsako	M40	55.4	11.043189	NaN	1.94	
85	86	Prague17	Simon Dunn	M45	33.2	11.066667	NaN	2.05	
86	87	Prague17	Pavel ? imek	M40	17.9	10.848485	ciclista 5h	2.05	

87 rows × 10 columns

In [597…
```
marathon_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 87 entries, 0 to 86
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   id             87 non-null     int64  
 1   Marathon       87 non-null     object 
 2   Name           87 non-null     object 
 3   Category       81 non-null     object 
 4   km4week        87 non-null     float64
 5   sp4week        87 non-null     float64
 6   CrossTraining  13 non-null     object 
 7   Wall21         87 non-null     object 
 8   MarathonTime   87 non-null     float64
 9   CATEGORY       87 non-null     object 
dtypes: float64(3), int64(1), object(6)
memory usage: 6.9+ KB
```

P2)

The CrossTraining attributes' value is missing for most of the samples, listing as
"NaN",and only 13 entires are non-empty. Since more than half of the samples are empty
for this attribute, the best approach would be to drop the column entirely. The
'Category' attribute only has 6 entries that are empty, so they best approach would be
to drop the samples that are empty and replace it with the most frequent value.

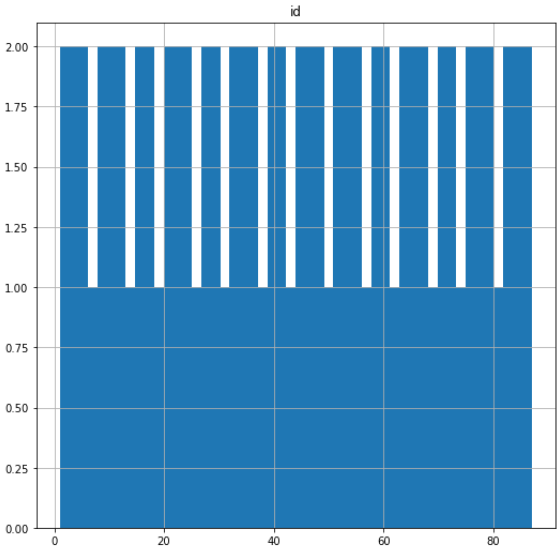
P3)

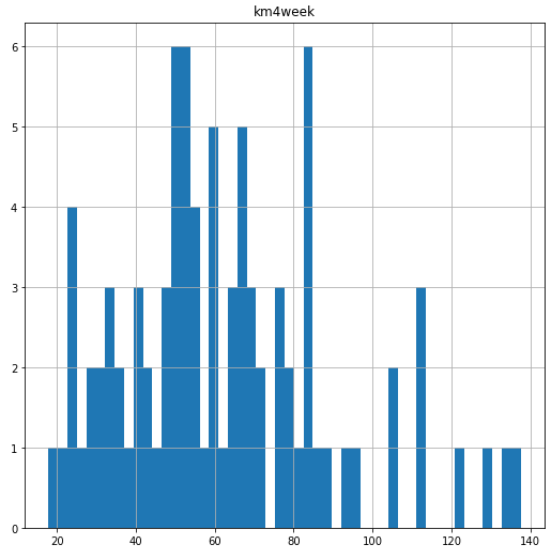
Note that the 'id' attribute cannot be considered since it is just a simple
counter of the samples, which does not affect the Marathon Time, so it could
be dropped later on.

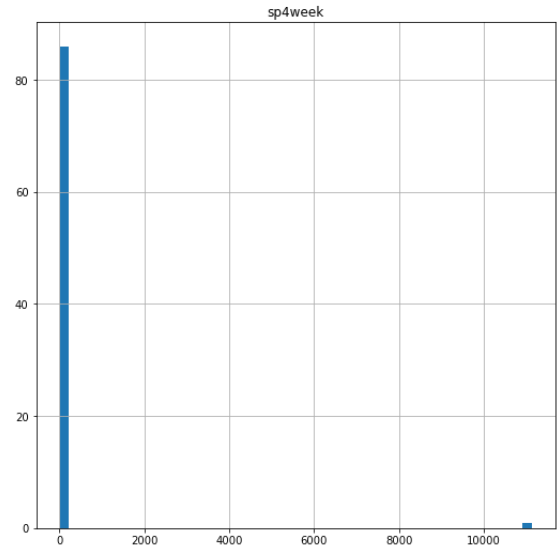
In [598…
```
corr_matrix = marathon_data.corr(method='pearson')
corr_matrix['MarathonTime'].sort_values(ascending=False)
```

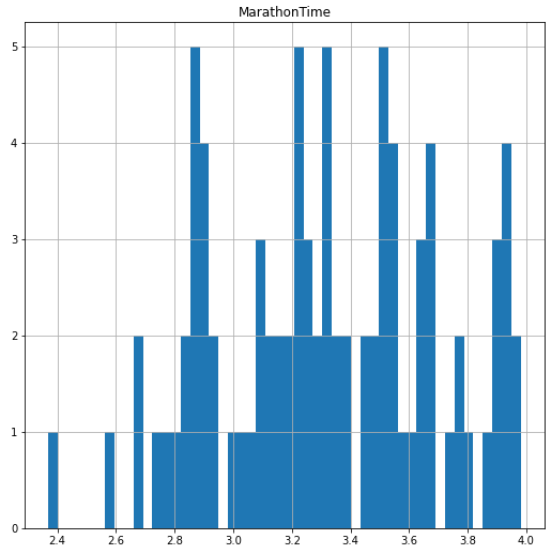
Out[598]:
```
MarathonTime    1.000000
id              0.991548
sp4week         0.172294
km4week        -0.606782
Name: MarathonTime, dtype: float64
```

In [599…
```
# visualize the data
marathon_data.hist(bins=50, figsize=(20,20));
```









- km4week has the largest predictive value to predict the marathon time.

P4) Custom Transformer

In [600…

```
from sklearn.base import BaseEstimator, TransformerMixin

km4week_ix, sp4week_ix = 0, 1

class CombinedAttributesAdder(BaseEstimator, TransformerMixin):
    def __init__(self, add_km4week_per_sp4week=True):
        self.add_km4week_per_sp4week = add_km4week_per_sp4week
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        if self.add_km4week_per_sp4week:
            km4week_per_sp4week = X[:,km4week_ix] / X[:,sp4week_ix]
        return np.c_[X, km4week_per_sp4week]
```

P5)

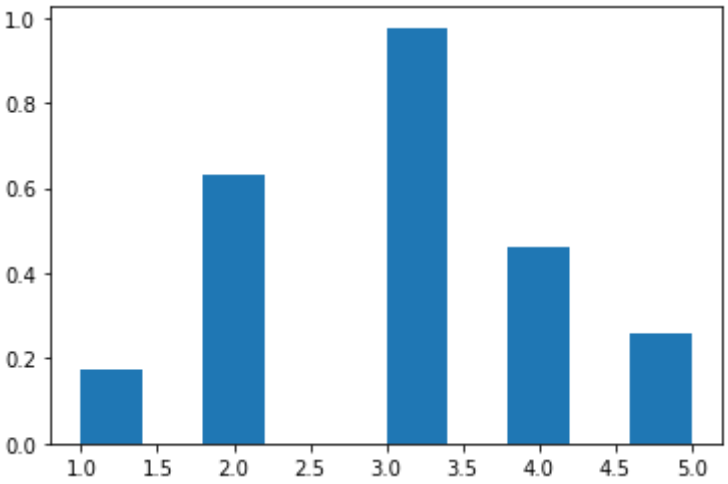
Since km4week is the most predictive attribute, it should be stratified
to maintain the prior probabilities of each category of km4week in the training
and test sets.

In [601…
```
import numpy as np

# stratified partition for km4week
km4week_cat = pd.cut(marathon_data['km4week'],
                     bins=[0.,25.,50.,75.,100.,np.inf],
                     labels=[1, 2, 3, 4, 5])
```

In [602…
```
import matplotlib.pyplot as plt

plt.hist(km4week_cat, density=True);
```



In [603…
```
from sklearn.model_selection import train_test_split

# Partition data into training and test sets
train_set, test_set, km4_cat_train, km4_cat_test = train_test_split(marathon_da
                                                                    km4week_cat
                                                                    test_size=(
                                                                    shuffle=Tru
                                                                    stratify=km
                                                                    random_stat
```

In [604…
```
train_set
```

Out[604]:

	id	Marathon	Name	Category	km4week	sp4week	CrossTraining	Wall21	Maratho
47	48	Prague17	Miguel Escribano	M45	39.6	12.247423	NaN	1.67	
43	44	Prague17	Filip Machart	MAM	32.2	12.710526	NaN	1.62	
58	59	Prague17	Mark Orton	MAM	22.7	12.728972	NaN	1.67	
59	60	Prague17	Hannes Lilljequist	M40	45.2	11.024390	NaN	1.77	
63	64	Prague17	Bradley Selmes	MAM	20.7	12.420000	ciclista 5h	1.69	
...	...	...	...	...	...	...	...	...	
70	71	Prague17	Roy Bruhn	M45	87.0	11.472527	NaN	1.81	
23	24	Prague17	Jiri Syrovatko	M45	66.1	12.128440	NaN	1.48	
15	16	Prague17	David Lehnen	MAM	76.1	14.970492	NaN	1.45	
76	77	Prague17	Michal Karhan	MAM	40.3	8.337931	ciclista 5h	1.94	
38	39	Prague17	Brian Parkinson	MAM	64.7	13.294521	NaN	1.50	

69 rows × 10 columns

In [605…

```
test_set
```

Out[605]:

	id	Marathon	Name	Category	km4week	sp4week	CrossTraining	Wall21	Mar
65	66	Prague17	Barry Sacher	M45	60.3	11.708738	NaN	1.88	
73	74	Prague17	Pavel Szappanos	NaN	53.6	12.711462	NaN	-	
71	72	Prague17	Martin Bo Meyer	M50	24.2	11.523810	ciclista 3h	1.76	
30	31	Prague17	Luka Slap?ak Pelliccioni	MAM	52.5	12.549801	ciclista 3h	1.62	
55	56	Prague17	Jakub Ka?par	MAM	26.9	13.121951	NaN	1.67	
64	65	Prague17	Frederic Bonningues	M40	54.2	11.782609	NaN	1.69	
82	83	Prague17	Stefano Vegliani	M55	50.0	10.830325	NaN	2.02	
31	32	Prague17	Ondrej Barta	MAM	79.4	13.344538	NaN	1.60	
83	84	Prague17	Andrej Madliak	M40	33.6	10.130653	ciclista 3h	1.94	
62	63	Prague17	Ale?Kuchynka	M45	48.8	11.665339	NaN	1.66	
14	15	Prague17	Denis Wachtl	MAM	76.8	12.943820	NaN	1.44	
3	4	Prague17	Daniel Or lek	M45	137.5	12.258544	NaN	1.32	
81	82	Prague17	Nathan Khan	MAM	35.6	11125.000000	NaN	2.05	
52	53	Prague17	Luk ? Charv t	M40	121.7	9.907734	NaN	1.65	
79	80	Prague17	Martin Werner	MAM	53.9	11.802920	NaN	1.98	
27	28	Prague17	Pavel Hlo?ek	MAM	82.4	11.771429	NaN	1.52	
56	57	Prague17	Mickael Rihouey	MAM	56.2	12.086022	NaN	1.71	
68	69	Prague17	Joost Saanen	MAM	59.1	10.910769	NaN	1.75	

In [606…

```
# Prepare data for ML algorithms
# training labels
t_train = train_set['MarathonTime'].copy()

# input data
X_train = train_set.drop(labels='MarathonTime', axis=1)
t_train.shape, X_train.shape
```

Out[606]: ((69,), (69, 9))

In [607…
```
# test labels
t_test = test_set['MarathonTime'].copy()

# input data
X_test = test_set.drop(labels='MarathonTime', axis=1)
t_test.shape, X_test.shape
```

Out[607]:
```
((18,), (18, 9))
```

In [608…
```
# Clean Data

X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 69 entries, 47 to 38
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             69 non-null     int64
 1   Marathon       69 non-null     object
 2   Name           69 non-null     object
 3   Category       64 non-null     object
 4   km4week        69 non-null     float64
 5   sp4week        69 non-null     float64
 6   CrossTraining  10 non-null     object
 7   Wall21         69 non-null     object
 8   CATEGORY       69 non-null     object
dtypes: float64(2), int64(1), object(6)
memory usage: 5.4+ KB
```

I noticed on the data sheet that id values 26, 35, 52,54, 74, and 81 have '- 'values for Wall21 attributes, which aren't identified as NaN, causing issues when pipelining by category/numerical attributes later on. In order to fix this, it is best to convert the data type of Wall21 to float and then replace '- ' with NaN before using SimpleImputer to replace all NaN values.

In [609…
```
X_train['Wall21'] = pd.to_numeric(X_train['Wall21'], errors='coerce')
X_test['Wall21'] = pd.to_numeric(X_test['Wall21'], errors='coerce')
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 69 entries, 47 to 38
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             69 non-null     int64
 1   Marathon       69 non-null     object
 2   Name           69 non-null     object
 3   Category       64 non-null     object
 4   km4week        69 non-null     float64
 5   sp4week        69 non-null     float64
 6   CrossTraining  10 non-null     object
 7   Wall21         64 non-null     float64
 8   CATEGORY       69 non-null     object
dtypes: float64(3), int64(1), object(5)
memory usage: 5.4+ KB
```

As mentioned before, CrossTraining is missing more than half of its samples so it should be dropped. The 'id','Marathon', and 'Name' attributes don't contribute to predicting the Marathon Time, so they should also be dropped

In [610…
```
X_train = X_train.drop(['id','CrossTraining','Marathon', 'Name'], axis=1);


X_test = X_test.drop(['id','CrossTraining','Marathon', 'Name'], axis=1);
```

Since the 'Category' attribute is only missing a few samples, it would be best to replace the empty samples with the most frequent value using SimpleImputer

In [611…
```
from sklearn.impute import SimpleImputer

# learns the most frequent value for all columns with NaN
imputer = SimpleImputer(strategy='most_frequent')
```

In [612…
```
Xtr = imputer.fit_transform(X_train)
Xts = imputer.transform(X_test)
```

In [613…
```
imputer.statistics_
```

Out[613]:
```
array(['MAM', 50.1, 8.031413613, 1.67, 'B'], dtype=object)
```

In [614…
```
marathon_data_tr = pd.DataFrame(Xtr, columns=X_train.columns,
                                index=X_train.index)
marathon_data_tr
```

Out[614]:

	Category	km4week	sp4week	Wall21	CATEGORY
47	M45	39.6	12.247423	1.67	B
43	MAM	32.2	12.710526	1.62	B
58	MAM	22.7	12.728972	1.67	C
59	M40	45.2	11.02439	1.77	C
63	MAM	20.7	12.42	1.69	C
...	...	...	...	...	...
70	M45	87.0	11.472527	1.81	D
23	M45	66.1	12.12844	1.48	B
15	MAM	76.1	14.970492	1.45	A
76	MAM	40.3	8.337931	1.94	D
38	MAM	64.7	13.294521	1.5	B

69 rows × 5 columns

In [615…
```
# Convert data types of km4week, sp4week and Wall21 to float
marathon_data_tr['km4week'] = marathon_data_tr['km4week'].astype(float)
marathon_data_tr['sp4week'] = marathon_data_tr['sp4week'].astype(float)
marathon_data_tr['Wall21'] = marathon_data_tr['Wall21'].astype(float)
```

```
marathon_data_tr.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 69 entries, 47 to 38
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Category  69 non-null     object 
 1   km4week   69 non-null     float64
 2   sp4week   69 non-null     float64
 3   Wall21    69 non-null     float64
 4   CATEGORY  69 non-null     object 
dtypes: float64(3), object(2)
memory usage: 3.2+ KB
```

In [616…

```
marathon_data_test = pd.DataFrame(Xts, columns=X_test.columns,
                                  index=X_test.index)
marathon_data_test
```

Out[616]:

	Category	km4week	sp4week	Wall21	CATEGORY
65	M45	60.3	11.708738	1.88	C
73	MAM	53.6	12.711462	1.67	D
71	M50	24.2	11.52381	1.76	D
30	MAM	52.5	12.549801	1.62	B
55	MAM	26.9	13.121951	1.67	C
64	M40	54.2	11.782609	1.69	C
82	M55	50.0	10.830325	2.02	D
31	MAM	79.4	13.344538	1.6	B
83	M40	33.6	10.130653	1.94	D
62	M45	48.8	11.665339	1.66	C
14	MAM	76.8	12.94382	1.44	A
3	M45	137.5	12.258544	1.32	A
81	MAM	35.6	11125.0	2.05	D
52	M40	121.7	9.907734	1.65	C
79	MAM	53.9	11.80292	1.98	D
27	MAM	82.4	11.771429	1.52	B
56	MAM	56.2	12.086022	1.71	C
68	MAM	59.1	10.910769	1.75	C

In [617…

```
# Convert data types of km4week, sp4week and Wall21 to float
marathon_data_test['km4week'] = marathon_data_test['km4week'].astype(float)
marathon_data_test['sp4week'] = marathon_data_test['sp4week'].astype(float)
marathon_data_test['Wall21'] = marathon_data_test['Wall21'].astype(float)

marathon_data_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 18 entries, 65 to 68
Data columns (total 5 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Category  18 non-null     object
 1   km4week   18 non-null     float64
 2   sp4week   18 non-null     float64
 3   Wall21    18 non-null     float64
 4   CATEGORY  18 non-null     object
dtypes: float64(3), object(2)
memory usage: 864.0+ bytes
```

6) Transformation Pipeline to encode categorical attributes

In [618…
```
from sklearn.preprocessing import OneHotEncoder
# Collect all categorical attributes by dropping numerical attributes
X_train_cat = marathon_data_tr.drop(labels=['km4week','sp4week','Wall21'], axis
X_test_cat = marathon_data_test.drop(labels=['km4week','sp4week','Wall21'], axi
```

In [619…
```
X_test_cat
```

Out[619]:

	Category	CATEGORY
65	M45	C
73	MAM	D
71	M50	D
30	MAM	B
55	MAM	C
64	M40	C
82	M55	D
31	MAM	B
83	M40	D
62	M45	C
14	MAM	A
3	M45	A
81	MAM	D
52	M40	C
79	MAM	D
27	MAM	B
56	MAM	C
68	MAM	C

In [620…
```
cat_encoder = OneHotEncoder()
```

```
X_train_cat_1hot = cat_encoder.fit_transform(X_train_cat)
X_test_cat_1hot = cat_encoder.transform(X_test_cat)
```

In [621…
```
X_train_cat_1hot
```

Out[621]:
```
<69x10 sparse matrix of type '<class 'numpy.float64'>'
        with 138 stored elements in Compressed Sparse Row format>
```

In [622…
```
X_test_cat_1hot
```

Out[622]:
```
<18x10 sparse matrix of type '<class 'numpy.float64'>'
        with 36 stored elements in Compressed Sparse Row format>
```

In [623…
```
X_train_cat_1hot.toarray()
```

Out[623]:
```
array([[0., 1., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0., 0., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1., 0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1., 0., 1., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 1., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [0., 1., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1., 0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 1., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.],
       [0., 0., 1., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1., 0., 0., 1., 0., 0.],
       [1., 0., 0., 0., 0., 0., 1., 0., 0., 0.],
```

```
       [0., 0., 0., 0., 1., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 1.],
       [0., 1., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1., 0., 0., 1., 0., 0.]])
```

In [624…
```
X_test_cat_1hot.toarray()
```

Out[624]:
```
array([[0., 1., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1., 0., 0., 1., 0., 0.],
       [1., 0., 0., 0., 0., 0., 0., 0., 0., 1.],
       [0., 1., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1., 0., 1., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 1., 0.]])
```

In [625…
```
cat_encoder.categories_
```

Out[625]:
```
[array(['M40', 'M45', 'M50', 'M55', 'MAM', 'WAM'], dtype=object),
 array(['A', 'B', 'C', 'D'], dtype=object)]
```

7) Transformation Pipeline to encode numerical attributes

In [626…
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

# Drop all non-numerical attributes for training and test sets

marathon_data_num_tr = marathon_data_tr.drop(labels=['Category', 'CATEGORY'], a
marathon_data_num_test = marathon_data_test.drop(labels=['Category', 'CATEGORY'

num_pipeline = Pipeline([('imputer', SimpleImputer(strategy='median')),
                         ('attribs_adder', CombinedAttributesAdder()),
                         ('std_scaler', StandardScaler())])

marathon_num_tr = num_pipeline.fit_transform(marathon_data_num_tr)
marathon_num_test = num_pipeline.transform(marathon_data_num_test)
marathon_num_tr
```

Out[626]:
```
array([[-0.87592143,  0.05825006,  0.2958742 , -0.9043226 ],
       [-1.15807744,  0.43192303,  0.04445401, -1.23534784],
       [-1.52030474,  0.44680661,  0.2958742 , -1.59001775],
       [-0.66239797, -0.92860068,  0.7987146 , -0.49448181],
       [-1.59656312,  0.1975007 ,  0.39644228, -1.64518863],
       [-0.39168072, -0.37623311,  1.65354327, -0.32108764],
       [ 1.89988359,  0.08788734, -0.15668215,  1.8936289 ],
       [ 1.65585677,  0.41908097, -1.01151082,  1.51531433],
       [ 1.89607067,  0.33821601, -0.8606587 ,  1.78328715],
       [-0.34592569,  1.53608616, -1.21264698, -0.63634978],
       [ 0.83607919,  0.30199675, -0.81037466,  0.75080364],
       [-0.93311522, -0.91319878,  0.2958742 , -0.8018624 ],
       [ 0.28320594,  1.28719693, -1.16236294, -0.02947016],
       [-0.77678554,  1.16004053, -1.46406718, -0.96737503],
       [-1.21908414, -0.89967315,  1.80439539, -1.12500609],
       [ 0.76744665, -0.4255081 ,  0.84899864,  0.92419782],
       [-1.00174776, -0.4261766 ,  0.5472944 , -0.95949347],
       [-1.07800614, -0.70000752,  0.64786248, -0.99890124],
       [ 0.53867151, -3.34359893, -1.01151082,  2.08278618],
       [-0.14384099,  0.52759143,  0.34615824, -0.26591676],
       [ 0.1154375 , -0.2291507 , -0.50867043,  0.17545023],
       [ 0.25270259, -1.71218737, -0.05611407,  0.82173762],
       [-0.39930656, -0.00952741, -0.00583003, -0.40778472],
       [-1.32965879, -0.44609158,  0.2958742 , -1.30628183],
       [ 0.83989211,  1.42804991, -1.26293102,  0.4355415 ],
       [-0.43743575, -0.76207586,  0.2958742 , -0.28167987],
       [-1.70332485, -1.07053671,  2.2066677 , -1.65307018],
       [-0.41837115,  0.67227657, -0.55895447, -0.55753424],
       [ 0.20694756, -0.30929987,  0.2958742 ,  0.29367354],
       [-0.47556494, -0.00418874, -0.25725023, -0.48660026],
       [ 0.22982507,  1.20967657, -1.91662353, -0.06099637],
       [-0.49081661,  1.74393343, -0.8606587 , -0.79398085],
       [ 2.55570565,  0.0103734 , -0.35781831,  2.59508716],
       [-1.31822003, -0.7869032 ,  1.80439539, -1.25111095],
       [ 0.11925042,  0.95816401, -1.06179486, -0.10828569],
       [ 1.21737109, -0.23273831, -0.81037466,  1.32615705],
       [ 1.1792419 ,  0.82687369, -0.96122678,  0.91631626],
       [ 0.18407005,  0.56784708, -0.45838639,  0.04146383],
       [-1.47454971, -0.10070899,  1.20098691, -1.49543911],
       [-0.27348023, -0.91343188,  1.65354327, -0.06099637],
       [ 0.18025713,  0.85862012, -0.55895447, -0.02947016],
       [-0.04851801,  0.16832231, -0.10639811, -0.09252258],
       [-0.32304818, -0.85432812,  0.09473805, -0.13193035],
       [-0.53275872,  0.06202481,  0.24559016, -0.55753424],
       [-0.53657164, -0.92994252,  1.60325923, -0.35261385],
       [ 0.5958653 , -0.14139244,  0.14502208,  0.64834345],
       [-0.73865635, -0.99934978,  0.74843056, -0.5654158 ],
       [-0.09427304,  0.00581018, -0.30753427, -0.10040414],
       [ 0.83226627,  1.34010764, -1.01151082,  0.45130461],
       [ 2.67771906,  1.8232055 , -2.26861181,  1.91727356],
       [-1.11994825, -0.89448832,  2.2066677 , -1.01466435],
       [-0.79966305,  0.04846366,  0.94956672, -0.82550706],
       [ 1.61391467, -0.36678028, -0.20696619,  1.79905026],
       [-1.47836263,  1.04610375,  0.2958742 , -1.59789931],
       [ 0.82464044,  0.96007432, -1.31321506,  0.54588325],
       [ 1.89225775, -0.72527152,  0.2958742 ,  2.27194347],
       [ 0.30989637,  0.39332169,  0.2958742 ,  0.20697645],
       [ 0.30989637, -0.31621828,  0.7987146 ,  0.40401529],
       [-0.47556494, -0.23707092,  0.04445401, -0.43931094],
       [ 1.00766055,  0.33817333, -1.16236294,  0.90843471],
```

```
       [ 0.76744665,  1.08542977, -1.56463526,  0.45918616],
       [-0.4679391 , -1.62476312,  0.2958742 , -0.09252258],
       [ 0.15737961, -0.49119957,  1.45240711,  0.29367354],
       [-0.12096347,  0.37364273, -0.55895447, -0.21074589],
       [ 0.93140217, -0.56700393,  0.99985075,  1.15276287],
       [ 0.1345021 , -0.03775539, -0.65952255,  0.14392402],
       [ 0.515794  ,  2.25546306, -0.81037466, -0.02947016],
       [-0.849231  , -3.09627357,  1.65354327, -0.14769346],
       [ 0.08112123,  0.90314123, -0.55895447, -0.13193035]])
```

In [627…
```
marathon_num_test
```

Out[627]:
```
array([[-8.66472025e-02, -3.76408499e-01,  1.35183903e+00,
         2.05605817e-03],
       [-3.42112775e-01,  4.32678384e-01,  2.95874203e-01,
        -4.39310939e-01],
       [-1.46311096e+00, -5.25625039e-01,  7.48430558e-01,
        -1.44026823e+00],
       [-3.84054883e-01,  3.02235467e-01,  4.44540059e-02,
        -4.55074047e-01],
       [-1.36016214e+00,  7.63897028e-01,  2.95874203e-01,
        -1.46391290e+00],
       [-3.19235261e-01, -3.16802972e-01,  3.96442282e-01,
        -2.58035209e-01],
       [-4.79377858e-01, -1.08518976e+00,  2.05581558e+00,
        -2.50153655e-01],
       [ 6.41620324e-01,  9.43499583e-01, -5.61140730e-02,
         3.80370625e-01],
       [-1.10469657e+00, -1.64974672e+00,  1.65354327e+00,
        -8.64914830e-01],
       [-5.25132886e-01, -4.11426820e-01,  2.45590163e-01,
        -4.55074047e-01],
       [ 5.42484430e-01,  6.20165183e-01, -8.60658704e-01,
         3.72489073e-01],
       [ 2.85692625e+00,  6.72235841e-02, -1.46406718e+00,
         2.87094154e+00],
       [-1.02843819e+00,  8.96680987e+03,  2.20666770e+00,
        -2.43183072e+00],
       [ 2.25448505e+00, -1.82961765e+00,  1.95306124e-01,
         3.37536096e+00],
       [-3.30674018e-01, -3.00414252e-01,  1.85467942e+00,
        -2.73798316e-01],
       [ 7.56007893e-01, -3.25824089e-01, -4.58386388e-01,
         8.76908498e-01],
       [-2.42976881e-01, -7.19826820e-02,  4.97010361e-01,
        -2.34390549e-01],
       [-1.32402230e-01, -1.02028016e+00,  6.98146518e-01,
         1.28160914e-01]])
```

8) Transformation pipeline that combines both categorical and numerical attributes

In [628…
```
from sklearn.compose import ColumnTransformer

# numerical columns
marathon_data_num_tr.columns
```

Out[628]:
```
Index(['km4week', 'sp4week', 'Wall21'], dtype='object')
```

In [629…
```
# categorial columns
X_train_cat.columns
```

Out[629]:
```
Index(['Category', 'CATEGORY'], dtype='object')
```

In [630…
```
num_attribs = list(marathon_data_num_tr.columns)
cat_attribs = list(X_train_cat.columns)

full_pipeline = ColumnTransformer([('num', num_pipeline, num_attribs),
                                   ('cat', OneHotEncoder(), cat_attribs)])

X_train_prepared = full_pipeline.fit_transform(X_train)

X_test_prepared = full_pipeline.transform(X_test)
```

In [631…
```
X_train_prepared.shape, X_test_prepared.shape
```

Out[631]:
```
((69, 15), (18, 15))
```

In [632…
```
attribute_labels = np.hstack((marathon_data_tr.columns, cat_encoder.categories_
attribute_labels
```

Out[632]:
```
array(['Category', 'km4week', 'sp4week', 'Wall21', 'CATEGORY', 'M40',
       'M45', 'M50', 'M55', 'MAM', 'WAM', 'A', 'B', 'C', 'D'],
      dtype=object)
```

In [633…
```
marathon_data_train_prepared = pd.DataFrame(X_train_prepared,
                                            columns=attribute_labels,
                                            index=marathon_data_tr.index)
marathon_data_train_prepared
```

Out[633]:

	Category	km4week	sp4week	Wall21	CATEGORY	M40	M45	M50	M55	MAM
47	-0.875921	0.058250	0.324276	-0.904323	0.0	1.0	0.0	0.0	0.0	0.0
43	-1.158077	0.431923	0.072021	-1.235348	0.0	0.0	0.0	0.0	1.0	0.0
58	-1.520305	0.446807	0.324276	-1.590018	0.0	0.0	0.0	0.0	1.0	0.0
59	-0.662398	-0.928601	0.828787	-0.494482	1.0	0.0	0.0	0.0	0.0	0.0
63	-1.596563	0.197501	0.425179	-1.645189	0.0	0.0	0.0	0.0	1.0	0.0
...	...	...	...	...	...	...	...	...	...	...
70	0.931402	-0.567004	1.030592	1.152763	0.0	1.0	0.0	0.0	0.0	0.0
23	0.134502	-0.037755	-0.634295	0.143924	0.0	1.0	0.0	0.0	0.0	0.0
15	0.515794	2.255463	-0.785648	-0.029470	0.0	0.0	0.0	0.0	1.0	0.0
76	-0.849231	-3.096274	1.686456	-0.147693	0.0	0.0	0.0	0.0	1.0	0.0
38	0.081121	0.903141	-0.533392	-0.131930	0.0	0.0	0.0	0.0	1.0	0.0

69 rows × 15 columns

In [634…
```
marathon_data_test_prepared = pd.DataFrame(X_test_prepared,
                                           columns=attribute_labels,
```

```
                                                index=marathon_data_test.index)
marathon_data_test_prepared
```

Out[634]:

	Category	km4week	sp4week	Wall21	CATEGORY	M40	M45	M50	M55	MAI
65	-0.086647	-0.376408	1.383750	0.002056	0.0	1.0	0.0	0.0	0.0	0.(
73	-0.342113	0.432678	-0.054107	-0.439311	0.0	0.0	0.0	0.0	0.0	0.(
71	-1.463111	-0.525625	0.778336	-1.440268	0.0	0.0	1.0	0.0	0.0	0.(
30	-0.384055	0.302235	0.072021	-0.455074	0.0	0.0	0.0	0.0	1.0	0.(
55	-1.360162	0.763897	0.324276	-1.463913	0.0	0.0	0.0	0.0	1.0	0.(
64	-0.319235	-0.316803	0.425179	-0.258035	1.0	0.0	0.0	0.0	0.0	0.(
82	-0.479378	-1.085190	2.090065	-0.250154	0.0	0.0	0.0	1.0	0.0	0.(
31	0.641620	0.943500	-0.028881	0.380371	0.0	0.0	0.0	0.0	1.0	0.(
83	-1.104697	-1.649747	1.686456	-0.864915	1.0	0.0	0.0	0.0	0.0	0.(
62	-0.525133	-0.411427	0.273825	-0.455074	0.0	1.0	0.0	0.0	0.0	0.(
14	0.542484	0.620165	-0.836099	0.372489	0.0	0.0	0.0	0.0	1.0	0.(
3	2.856926	0.067224	-1.441512	2.870942	0.0	1.0	0.0	0.0	0.0	0.(
81	-1.028438	8966.809868	2.241418	-2.431831	0.0	0.0	0.0	0.0	1.0	0.(
52	2.254485	-1.829618	0.223374	3.375361	1.0	0.0	0.0	0.0	0.0	0.(
79	-0.330674	-0.300414	1.888261	-0.273798	0.0	0.0	0.0	0.0	1.0	0.(
27	0.756008	-0.325824	-0.432490	0.876908	0.0	0.0	0.0	0.0	1.0	0.(
56	-0.242977	-0.071983	0.526081	-0.234391	0.0	0.0	0.0	0.0	1.0	0.(
68	-0.132402	-1.020280	0.727885	0.128161	0.0	0.0	0.0	0.0	1.0	0.(

Submit Your Solution

Confirm that you've successfully completed the assignment.

Along with the Notebook, include a PDF of the notebook with your solutions.

`add` and `commit` the final version of your work, and `push` your code to your GitHub repository.

Submit the URL of your GitHub Repository as your assignment submission on Canvas.

In [ ]: