zad1

November 13, 2024

[1]:
```
import polars as pl
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split, KFold
from sklearn.linear_model import LinearRegression
```

[2]:
```
def RMSE(y_hat: pl.Series, y_obs: pl.DataFrame):

    return (((y_hat - y_obs.to_series())**2).sum()/y_obs.shape[0])**0.5
```

[3]:
```
input_matrix = pl.read_excel(
    source="dane_leki.xlsx"
)
```

[4]:
```
input_matrix.head()
```

[4]:
shape: (5, 8)

__UNNAMED__	Nazwa	logK HSA	logKCTAB	CATS3D_00_D	CATS3D_09_A	CATS3D_00_	Zbiór
0	---	---	---	D	L	AA	---
---	str	f64	f64	---	---	---	str
i64				i64	i64	i64	
1	acetaminoph en	-0.79	-0.63	2	0	2	t
2	acetylsalic ylic acid	-0.23	1.22	1	0	4	t
3	bromazepam	0.38	0.57	1	0		

```
3           t
 4             carbamazepi   0.69       0.68       0            0
3           t
              ne

 5             chlorpromaz   1.18       1.5        0            0
2           t
              ine
```

[5]:
```
Y_obs = input_matrix.select(
    pl.nth(2)
)
```

[6]:
```
Y_obs.head()
```

[6]:
```
shape: (5, 1)

 logK HSA
 ---
 f64

 -0.79
 -0.23
 0.38
 0.69
 1.18
```

[7]:
```
descriptors = input_matrix.select(
    pl.nth([3,4,5,6])
)
```

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

[8]:
```
shape: (5, 4)

 logKCTAB   CATS3D_00_DD   CATS3D_09_AL   CATS3D_00_AA
 ---        ---            ---            ---
 f64        i64            i64            i64

 -0.63      2              0              2
 1.22       1              0              4
 0.57       1              0              3
 0.68       0              0              3
```

```
1.5        0              0              2
```

[9]:
```
pca_model = PCA(n_components=4)
```

[10]:
```
pca_model.fit(descriptors)
```

[10]:
```
PCA(n_components=4)
```

[11]:
```
pca_model.explained_variance_
```

[11]:
```
array([4.67581967, 3.27064572, 0.96644472, 0.27022295])
```

[12]:
```
pca_model.components_
```

[12]:
```
array([[ 0.05905699,  0.02081031,  0.64914805,  0.75808048],
       [ 0.30296835, -0.55414496,  0.58475165, -0.50911595],
       [-0.22498561,  0.74897495,  0.4718871 , -0.40711247],
       [ 0.92417743,  0.36266627, -0.11829997,  0.01934869]])
```

[19]:
```
PC = pl.DataFrame(
    pca_model.fit_transform(descriptors),
    schema=[f"PC{i+1}" for i in range(pca_model.n_components_)])
```

[20]:
```
PC
```

[20]:
```
shape: (27, 4)
```

PC1	PC2	PC3	PC4
f64	f64	f64	f64
-2.300697	-0.649955	0.949429	-0.779364
-0.696091	-0.553551	-1.029994	0.606395
-1.492558	-0.241364	-0.476641	-0.013669
-1.506872	0.346107	-1.250365	-0.274675
-2.216526	1.103657	-1.02774	0.463801
…	…	…	…
0.98374	3.79413	1.504934	-0.839315
2.682795	1.400557	0.358198	0.531457
1.407812	-0.971157	0.982119	1.206894
0.557913	0.388726	0.082519	-0.323338
1.522327	-3.689086	-0.382156	-0.135119

[17]:
```
X_training, X_validation, Y_training, Y_validation = train_test_split(
    PC,
```

```
    Y_obs,
    test_size=0.33,
    random_state=42
)
```

[18]:
```
X_training
```

[18]:
shape: (18, 2)

PC1	PC2
f64	f64
1.407812	-0.971157
-0.696091	-0.553551
-2.216526	1.103657
-3.006217	0.128802
-1.492558	-0.241364
…	…
2.739027	0.697231
-0.251225	2.288619
0.711546	-2.789428
0.783145	-3.083021
-1.47321	0.518799

[19]:
```
Y_training
```

[19]:
shape: (18, 1)

logK HSA
f64
0.08
-0.23
1.18
-0.42
0.38
…
0.06
2.05
-1.25
-1.25
1.08

[ ]:
```
def principal_component_plot(n: int):

    pca_model = PCA(n_components=n)

    PC = pl.DataFrame(
        pca_model.fit_transform(descriptors),
        schema=[f"PC{i+1}" for i in range(pca_model.n_components_)]
    )

    KFold_model = KFold(
        n_splits=10,
        shuffle=True,
        random_state=0
    )

    X_training, _, Y_training, _ = train_test_split(
        PC,
        Y_obs,
        test_size=0.33,
        random_state=42
    )

    validation_sets = [
        validation_set for (_, validation_set) in KFold_model.split(X_training,␣
↪Y_training)
    ]

    X=[
        X_training.with_row_index().filter(
            ~pl.col("index").is_in(validation_set)
        ).drop(
            pl.col("index")
        ) for validation_set in validation_sets
    ]

    Y=[
        Y_training.with_row_index().filter(
            ~pl.col("index").is_in(validation_set)
        ).drop(
            pl.col("index")
        ) for validation_set in validation_sets
    ]

    PCR_models = [
        LinearRegression().fit(
            X=x,
            y=y
```

```
        ) for (x,y) in zip(X,Y)
    ]


    residues = [
        (PCR_models[idx].predict(
            X_training.with_row_index().filter(
            pl.col("index").is_in(validation_sets[idx])
        ).drop(
            pl.col("index")
        )
        ).reshape(-1) - Y_training.with_row_index().filter(
            ~pl.col("index").is_in(validation_sets[idx])
        ).drop(
            pl.col("index")
        )) for idx in range(len(validation_sets))
    ]

    print(residues)
    print(residues[0].shape)
```

[31]:
```
principal_component_plot(4)
```

```
[array([[ 0.30838227,  2.12834894],
       [-0.79161773,  1.02834894],
       [ 0.80838227,  2.62834894],
       [ 0.00838227,  1.82834894],
       [ 0.98838227,  2.80834894],
       [-0.30161773,  1.51834894],
       [ 0.18838227,  2.00834894],
       [-1.43161773,  0.38834894],
       [-1.90161773, -0.08165106],
       [ 1.63838227,  3.45834894],
       [-1.30161773,  0.51834894],
       [ 0.32838227,  2.14834894],
       [-1.66161773,  0.15834894],
       [ 1.63838227,  3.45834894],
       [ 1.63838227,  3.45834894],
       [-0.69161773,  1.12834894]]), array([[ 0.41455754,  1.47717266],
       [ 0.72455754,  1.78717266],
       [-0.68544246,  0.37717266],
       [ 0.91455754,  1.97717266],
       [ 0.11455754,  1.17717266],
       [ 1.09455754,  2.15717266],
       [-1.34544246, -0.28282734],
```

```
[-0.19544246,  0.86717266],
[-1.32544246, -0.26282734],
[ 1.74455754,  2.80717266],
[-1.19544246, -0.13282734],
[ 0.43455754,  1.49717266],
[-1.55544246, -0.49282734],
[ 1.74455754,  2.80717266],
[ 1.74455754,  2.80717266],
[-0.58544246,  0.47717266]]), array([[ 0.06667332,  1.41311439],
[ 0.37667332,  1.72311439],
[-1.03332668,  0.31311439],
[ 0.56667332,  1.91311439],
[ 0.74667332,  2.09311439],
[-1.69332668, -0.34688561],
[-0.54332668,  0.80311439],
[-0.05332668,  1.29311439],
[-1.67332668, -0.32688561],
[-2.14332668, -0.79688561],
[ 1.39667332,  2.74311439],
[-1.54332668, -0.19688561],
[ 0.08667332,  1.43311439],
[ 1.39667332,  2.74311439],
[ 1.39667332,  2.74311439],
[-0.93332668,  0.41311439]]), array([[ 0.9129948 , -1.13408795],
[ 1.2229948 , -0.82408795],
[ 1.4129948 , -0.63408795],
[ 0.6129948 , -1.43408795],
[ 1.5929948 , -0.45408795],
[-0.8470052 , -2.89408795],
[ 0.3029948 , -1.74408795],
[ 0.7929948 , -1.25408795],
[-0.8270052 , -2.87408795],
[-1.2970052 , -3.34408795],
[ 2.2429948 ,  0.19591205],
[-0.6970052 , -2.74408795],
[ 0.9329948 , -1.11408795],
[-1.0570052 , -3.10408795],
[ 2.2429948 ,  0.19591205],
[-0.0870052 , -2.13408795]]), array([[ 1.32514028,  0.5920098 ],
[ 1.63514028,  0.9020098 ],
[ 0.22514028, -0.5079902 ],
[ 1.82514028,  1.0920098 ],
[ 1.02514028,  0.2920098 ],
[ 2.00514028,  1.2720098 ],
[-0.43485972, -1.1679902 ],
[ 0.71514028, -0.0179902 ],
[ 1.20514028,  0.4720098 ],
[-0.88485972, -1.6179902 ],
```

```
[ 2.65514028,  1.9220098 ],
[-0.28485972, -1.0179902 ],
[ 1.34514028,  0.6120098 ],
[-0.64485972, -1.3779902 ],
[ 2.65514028,  1.9220098 ],
[ 2.65514028,  1.9220098 ]]), array([[ 0.50128563,  1.31061869],
[ 0.81128563,  1.62061869],
[-0.59871437,  0.21061869],
[ 1.00128563,  1.81061869],
[ 0.20128563,  1.01061869],
[ 1.18128563,  1.99061869],
[-1.25871437, -0.44938131],
[ 0.38128563,  1.19061869],
[-1.23871437, -0.42938131],
[-1.70871437, -0.89938131],
[ 1.83128563,  2.64061869],
[-1.10871437, -0.29938131],
[-1.46871437, -0.65938131],
[ 1.83128563,  2.64061869],
[ 1.83128563,  2.64061869],
[-0.49871437,  0.31061869]]), array([[ 0.49594767, -1.61405204],
[ 0.80594767, -1.30405204],
[-0.60405233, -2.71405204],
[ 0.19594767, -1.91405204],
[ 1.17594767, -0.93405204],
[-1.26405233, -3.37405204],
[-0.11405233, -2.22405204],
[ 0.37594767, -1.73405204],
[-1.24405233, -3.35405204],
[-1.71405233, -3.82405204],
[-1.11405233, -3.22405204],
[ 0.51594767, -1.59405204],
[-1.47405233, -3.58405204],
[ 1.82594767, -0.28405204],
[ 1.82594767, -0.28405204],
[-0.50405233, -2.61405204]]), array([[ 0.67502147, -1.09861092],
[-0.73497853, -2.50861092],
[ 0.86502147, -0.90861092],
[ 0.06502147, -1.70861092],
[-1.39497853, -3.16861092],
[-0.24497853, -2.01861092],
[ 0.24502147, -1.52861092],
[-1.37497853, -3.14861092],
[-1.84497853, -3.61861092],
[ 1.69502147, -0.07861092],
[-1.24497853, -3.01861092],
[ 0.38502147, -1.38861092],
[-1.60497853, -3.37861092],
```

```
       [ 1.69502147, -0.07861092],
       [ 1.69502147, -0.07861092],
       [-0.63497853, -2.40861092]]), array([[-1.37804952],
       [-1.06804952],
       [-2.47804952],
       [-0.87804952],
       [-1.67804952],
       [-0.69804952],
       [-3.13804952],
       [-1.98804952],
       [-1.49804952],
       [-3.11804952],
       [-3.58804952],
       [-0.04804952],
       [-2.98804952],
       [-1.35804952],
       [-3.34804952],
       [-0.04804952],
       [-2.37804952]]), array([[ 1.20805674],
       [ 1.51805674],
       [ 0.10805674],
       [ 1.70805674],
       [ 0.90805674],
       [ 1.88805674],
       [-0.55194326],
       [ 0.59805674],
       [ 1.08805674],
       [-0.53194326],
       [-1.00194326],
       [ 2.53805674],
       [ 1.22805674],
       [-0.76194326],
       [ 2.53805674],
       [ 2.53805674],
       [ 0.20805674]])]
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[31], line 1
----> 1 principal_component_plot(4)

Cell In[30], line 66, in principal_component_plot(n)
     51 residues = [
     52     (PCR_models[idx].predict(
     53         X_training.with_row_index().filter(
   (…)
     62     )) for idx in range(len(validation_sets))
```

```
     63 ]
     65 print(residues)
---> 66 print(residues.shape)

AttributeError: 'list' object has no attribute 'shape'
```

[20]:
```
KFold_model = KFold(
    n_splits=10,
    shuffle=True,
    random_state=0
)
```

[21]:
```
validation_sets = []
for training_set, validation_set in KFold_model.split(X_training, Y_training):
    validation_sets.append(validation_set)
    print(f"Training: {training_set}\nValidation: {validation_set}")
```

```
Training: [ 0  2  3  4  5  7  8  9 10 11 12 13 14 15 16 17]
Validation: [1 6]
Training: [ 0  1  2  3  4  5  6  7  9 11 12 13 14 15 16 17]
Validation: [ 8 10]
Training: [ 0  1  2  3  5  6  7  8  9 10 11 12 13 15 16 17]
Validation: [ 4 14]
Training: [ 0  1  3  4  5  6  7  8  9 10 11 12 13 14 15 17]
Validation: [ 2 16]
Training: [ 0  1  2  3  4  5  6  7  8 10 11 12 13 14 15 16]
Validation: [ 9 17]
Training: [ 0  1  2  3  4  5  6  8  9 10 11 12 14 15 16 17]
Validation: [ 7 13]
Training: [ 0  1  2  4  5  6  7  8  9 10 12 13 14 15 16 17]
Validation: [ 3 11]
Training: [ 1  2  3  4  6  7  8  9 10 11 12 13 14 15 16 17]
Validation: [0 5]
Training: [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 16 17]
Validation: [15]
Training: [ 0  1  2  3  4  5  6  7  8  9 10 11 13 14 15 16 17]
Validation: [12]
```

[22]:
```
descriptors
```

[22]: shape: (27, 4)

logKCTAB	CATS3D_00_DD	CATS3D_09_AL	CATS3D_00_AA
f64	i64	i64	i64
-0.63	2	0	2

```
1.22        1           0           4
0.57        1           0           3
0.68        0           0           3
1.5         0           0           2
…           …           …           …
0.73        0           5           2
1.63        1           4           5
1.32        3           2           5
0.47        1           2           4
-0.43       3           0           7
```

[71]:
```
LinearRegression().fit(
            X=X_training.with_row_index().filter(~pl.col("index").
 ↪is_in(validation_set)).drop(pl.col("index")),
            y=Y_training.with_row_index().filter(~pl.col("index").
 ↪is_in(validation_set)).drop(pl.col("index"))
   )
```

[71]:
```
LinearRegression()
```

[86]:
```
pl.Series(prediction[i][0] for i in range(len(prediction)))
```

[86]:
```
shape: (2,)
Series: '' [f64]
[
        0.159224
        2.714788
]
```

[76]:
```
Y_training.with_row_index().filter(pl.col("index").is_in(validation_set)).
 ↪drop(pl.col("index"))
```

[76]:
```
shape: (2, 1)

 logK HSA
 ---
 f64

 -0.23
 1.84
```

[88]:
```
#predictions = pl.DataFrame()
predictions = []
residues = []
RMSEs = []
```

```
for idx, validation_set in enumerate(validation_sets):
    x=X_training.with_row_index().filter(~pl.col("index").
 ↪is_in(validation_set)).drop(pl.col("index"))
    y=Y_training.with_row_index().filter(~pl.col("index").
 ↪is_in(validation_set)).drop(pl.col("index"))
    PCR_model = LinearRegression().fit(
            X=x,
            y=y
    )
    prediction = PCR_model.predict(X_training.with_row_index().filter(pl.
 ↪col("index").is_in(validation_set)).drop(pl.col("index")))


    RMSEs.append(
        RMSE(
            pl.Series(prediction[i][0] for i in range(len(prediction))),
            Y_training.with_row_index().filter(pl.col("index").
 ↪is_in(validation_set)).drop(pl.col("index"))
        )
    )

print(sum(RMSEs))
```

```
5.181391908438678
```

[70]:
```
x
```

[70]:
```
shape: (18,)
Series: '' [array[f64, 1]]
[
        [0.159224]
        [2.714788]
        [0.655465]
        [1.780267]
        [0.185102]
        …
        [-1.522106]
        [-0.054193]
        [-1.08324]
        [-0.931429]
        [1.030242]
]
```

[68]:
```
predictions
```

```
[68]: [shape: (2,)
       Series: '' [array[f64, 1]]
       [
              [0.159224]
              [2.714788]
       ],
       shape: (2,)
       Series: '' [array[f64, 1]]
       [
              [0.655465]
              [1.780267]
       ],
       shape: (2,)
       Series: '' [array[f64, 1]]
       [
              [0.185102]
              [1.548124]
       ],
       shape: (2,)
       Series: '' [array[f64, 1]]
       [
              [0.845459]
              [-1.10008]
       ],
       shape: (2,)
       Series: '' [array[f64, 1]]
       [
              [1.208883]
              [0.583907]
       ],
       shape: (2,)
       Series: '' [array[f64, 1]]
       [
              [0.51657]
              [1.110712]
       ],
       shape: (2,)
       Series: '' [array[f64, 1]]
       [
              [0.588338]
              [-1.522106]
       ],
       shape: (2,)
       Series: '' [array[f64, 1]]
       [
              [-0.054193]
              [-1.08324]
```

```
],
shape: (1,)
Series: '' [array[f64, 1]]
[
        [-0.931429]
],
shape: (1,)
Series: '' [array[f64, 1]]
[
        [1.030242]
]]
```

[46]:
```
residues.head(2)
```

[46]:
```
shape: (2,)
Series: '' [o][object]
[
        shape: (2, 1)

 logK HSA
 ---
 f64

 -0.389224
 -0.874788

        shape: (2, 1)

 logK HSA
 ---
 f64

 -0.455465
 0.509733

]
```

[ ]:
```

```