Feature selection of molecular descriptors

Álvaro Román Gómez

5/29/23

Table of contents

1 FILTER METHODS **2**
1.1 LOW VARIANCE FILTER . . . 2
1.2 CORRELATION FILTER . . . 3
1.2.1 LINEAR CORRELATED VARIABLES . . . 3
1.2.2 NON-LINEAR CORRELATED VARIABLES . . . 3

2 PCA SELECTION **3**
2.1 GET NUMBER OF COMPONENTS TO EXPLAIN 95% OF VARIANCE . . 3

3 GENETIC ALGORITH FEATURE SELECTION **6**

4 SCALE DATA **10**

5 CONCATENATE X AND Y **10**

6 PREPARE TEST DATA **11**

7 SAVE DATA **12**

```python
# PATH TO CUSTOM MODULES
import sys

sys.path.append("../src")

# IMPORT LIBRARIES
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```python
from sklearn.tree import DecisionTreeClassifier
from genetic_selection import GeneticSelectionCV
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA

# IMPORT CUSTOM MODULES
import utils.statisticsUtils as st
```

```python
# DIRECTORIES
input_path = "../data/raw/"
input_train_path = "../data/raw/train_data/"
input_test_path = "../data/raw/test_data/"

output_train_path = "../data/processed/train_data/"
output_test_path = "../data/processed/test_data/"
```

```python
# FILES
molecular_descriptors_training_file = "molecular_descriptors_training.csv"
molecular_descriptors_test_file = "molecular_descriptors_test.csv"
```

```python
dataset_name = "molecular_descriptors"
```

```python
# LOAD DATA
molecular_descriptors = pd.read_csv(
    input_train_path + molecular_descriptors_training_file
)
X = molecular_descriptors.drop(columns=["activity"])
Y = molecular_descriptors["activity"]
```

1 FILTER METHODS

1.1 LOW VARIANCE FILTER

```python
# REMOVE VARIABLES WITH LOW VARIANCE
molecules_low_var_filter = st.low_variance_filter(X, 0.1)
X_filter = X[molecules_low_var_filter]
```

1.2 CORRELATION FILTER

First of all, we are going to remove the variables that are highly correlated with each other. We will use the Spearman correlation coefficient to measure the correlation between the variables. When two variables are highly correlated, we will remove the one that has the lowest correlation with the target variable.

1.2.1 LINEAR CORRELATED VARIABLES

```
# REMOVE VARIABLES WITH HIGH PEARSON CORRELATION
molecules_corr_filter_pearson = st.pearson_corr_filter(X_filter, 0.8, Y)
```

```
X_filter = X_filter.drop(columns=molecules_corr_filter_pearson)
```

1.2.2 NON-LINEAR CORRELATED VARIABLES

```
# REMOVE VARIABLES WITH HIGH SPEARMAN CORRELATION
molecules_corr_filter_spearman = st.spearman_corr_filter(X_filter, 0.8, Y)
```

```
X_filter = X_filter.drop(columns=molecules_corr_filter_spearman)
```

2 PCA SELECTION

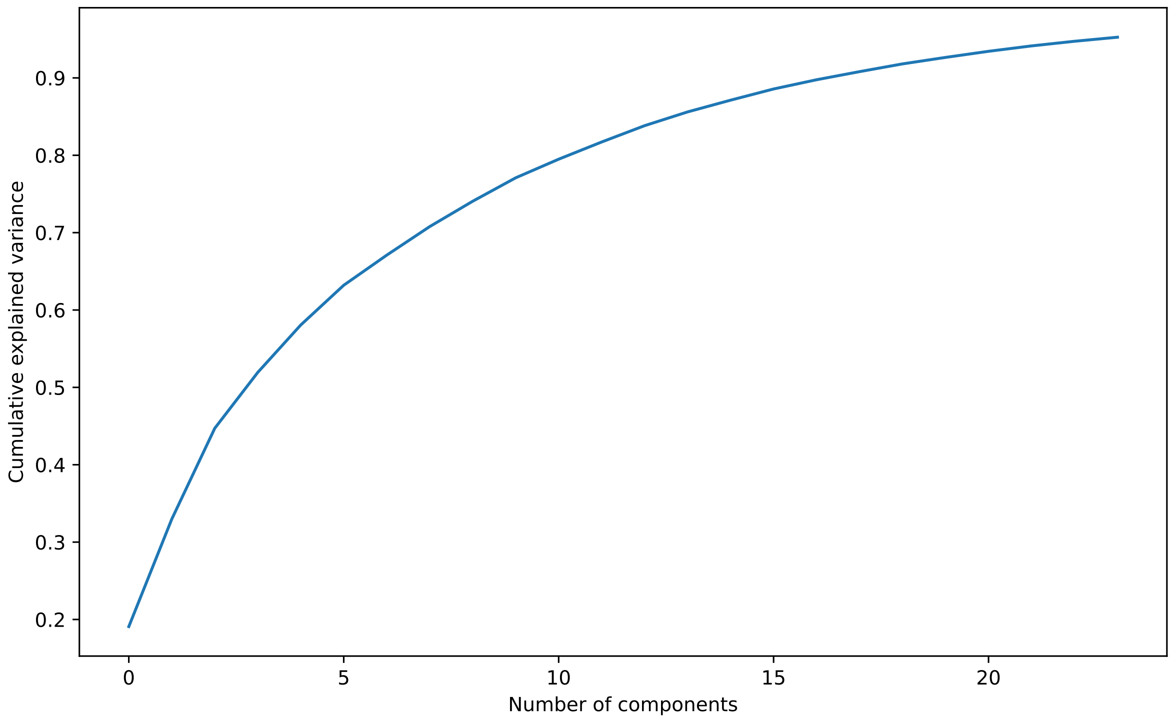
2.1 GET NUMBER OF COMPONENTS TO EXPLAIN 95% OF VARIANCE

```
# SCALE DATA FOR PCA
scaler = MinMaxScaler()
X_filter_scaled = pd.DataFrame(scaler.fit_transform(X_filter), columns=X_filter.columns)
```

```
# PCA 95% VARIANCE
pca = PCA(n_components=0.95)
# DATAFRAME
X_pca = pd.DataFrame(pca.fit_transform(X_filter_scaled))
```

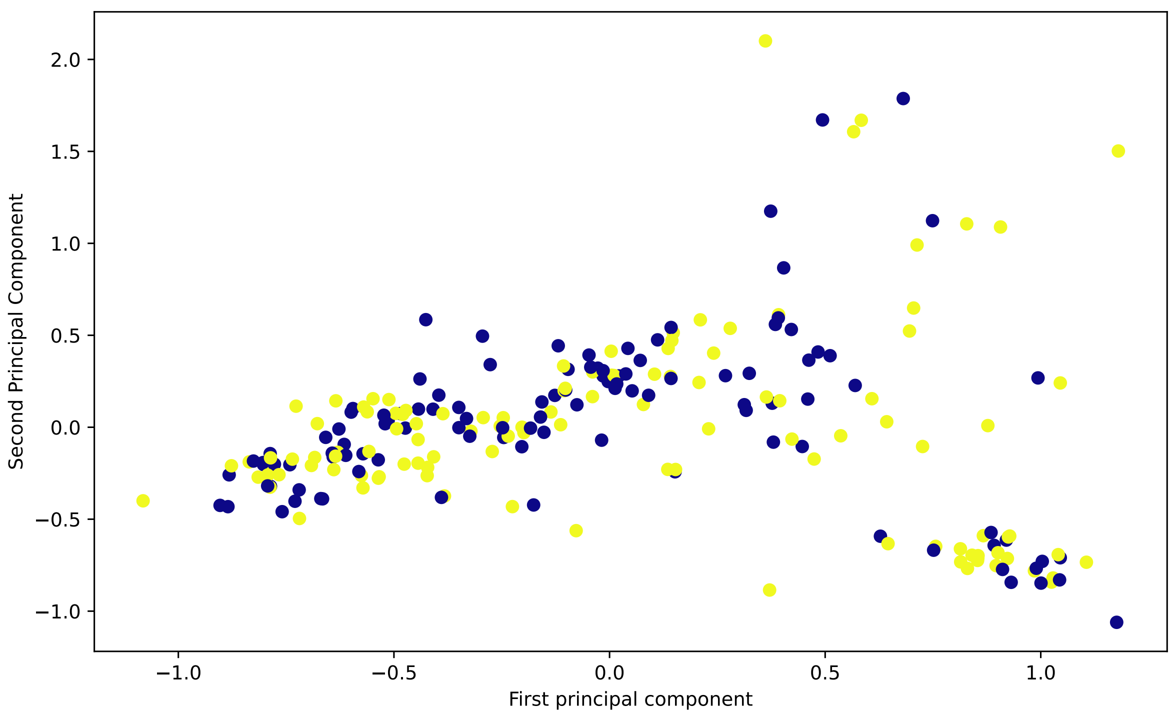
```
# PLOT PCA
pca.fit(X_filter_scaled)
plt.figure(figsize=(10, 6))
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel("Number of components")
plt.ylabel("Cumulative explained variance")
```

```
Text(0, 0.5, 'Cumulative explained variance')
```



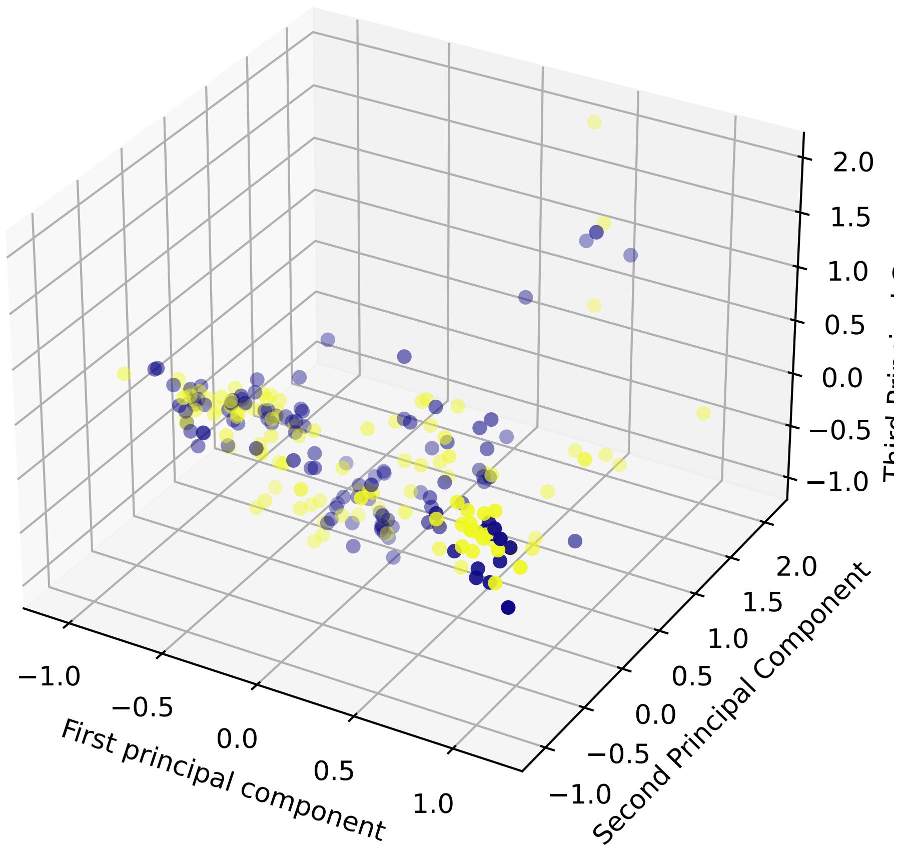
```
# PLOT TWO FIRST COMPONENTS
plt.figure(figsize=(10, 6))
plt.scatter(X_pca[0], X_pca[1], c=Y, cmap="plasma")
plt.xlabel("First principal component")
plt.ylabel("Second Principal Component")
```

```
Text(0, 0.5, 'Second Principal Component')
```



```
# PLOT THREE FIRST COMPONENTS 3D
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection="3d")
ax.scatter(X_pca[0], X_pca[1], X_pca[2], c=Y, cmap="plasma")
ax.set_xlabel("First principal component")
ax.set_ylabel("Second Principal Component")
ax.set_zlabel("Third Principal Component")
```

```
Text(0.5, 0, 'Third Principal Component')
```



3 GENETIC ALGORITH FEATURE SELECTION

```
# GENETIC ALGORITHM FEATURE SELECTION
from genetic_selection import GeneticSelectionCV
from sklearn.metrics import roc_auc_score
from sklearn.ensemble import RandomForestClassifier
```

```
# TRAIN MODEL
model = RandomForestClassifier(random_state=100)
```

```
# SCALED VARIABLES
scaler = MinMaxScaler()
```

```
X_scaled = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
```

```
model.fit(X_scaled, Y)
```

```
RandomForestClassifier(random_state=100)
```

```
# SELECT FEATURES
selector = GeneticSelectionCV(
    model,
    cv=5,
    verbose=1,
    scoring="roc_auc",
    crossover_proba=0.5,
    mutation_proba=0.2,
    n_generations=50,
    crossover_independent_proba=0.5,
    mutation_independent_proba=0.05,
    tournament_size=3,
    n_gen_no_change=20,
    caching=True,
    n_jobs=-1,
)
selector = selector.fit(X_scaled, Y)
```

```
Selecting features with genetic algorithm.

gen nevals  avg                                     std                                 min
0   300     [    0.573265   128.873333    0.092016]   [  0.018993   74.47839     0.014836]  [ 0.

1   184     [    0.58419    136.536667    0.09442 ]   [  0.015855   60.05205     0.013189]  [ 0.

2   176     [    0.590926   124.78        0.097293]   [  0.014292   48.874925    0.011463]  [  0

3   183     [    0.597088   110.743333    0.099676]   [  0.014172   41.634891    0.011473]  [  0

4   172     [  0.603395   97.483333    0.100622]      [  0.013932   30.713999    0.012123]  [  0
```

```
5   187     [  0.606323  88.243333   0.101197]      [  0.015546  22.267408   0.013112]  [  0
6   172     [  0.611837  87.543333   0.101945]      [  0.015105  16.729857   0.012711]  [  0
7   176     [  0.617656  84.91       0.104028]      [  0.013429  14.790602   0.014061]  [  0
8   187     [  0.621171  83.346667   0.103753]      [  0.014518  11.941517   0.013333]  [  0
9   159     [  0.626471  80.88       0.102773]      [  0.014646  11.206201   0.012394]  [  0
10  159     [  0.631809  79.68       0.102946]      [ 0.013999  9.479677  0.011901]     [  0
11  185     [  0.634524  77.71       0.103872]      [ 0.014935  9.277171  0.011813]     [  0
12  183     [  0.638116  77.036667   0.104608]      [ 0.014048  8.570219  0.010897]     [  0
13  185     [  0.642037  75.453333   0.1033  ]      [ 0.014724  7.866881  0.011702]     [  0
14  178     [  0.646757  75.         0.10212 ]      [ 0.013558  7.038939  0.012101]     [  0
15  188     [  0.649964  74.516667   0.102001]      [ 0.013822  7.040103  0.012729]     [  0
16  148     [  0.654273  74.333333   0.101356]      [ 0.013325  7.077821  0.012061]     [  0
17  175     [  0.656859  73.33       0.100979]      [ 0.014006  7.219956  0.011449]     [  0
18  175     [  0.65874   71.78       0.100204]      [ 0.013677  6.800853  0.010531]     [  0
19  180     [  0.660418  72.04       0.100266]      [ 0.013935  6.984631  0.011668]     [  0
20  179     [  0.661462  70.766667   0.09877 ]      [ 0.014368  6.81363   0.011846]     [  0
21  184     [  0.662965  70.27       0.098818]      [ 0.014483  6.076493  0.01104 ]     [  0
22  181     [  0.664708  70.56       0.099474]      [ 0.014438  5.702023  0.010117]     [  0
23  193     [  0.66597   71.573333   0.098559]      [ 0.015313  6.025332  0.010889]     [  0
```

```
24  177     [  0.667745  71.35        0.097099]      [ 0.014582  5.921219  0.009789]      [  0
25  189     [  0.670357  70.51        0.09775 ]      [ 0.014746  5.986365  0.01097 ]      [  0
26  177     [  0.673995  69.283333    0.097842]      [ 0.014345  6.23028   0.010393]      [  0
27  164     [  0.676947  68.253333    0.097579]      [ 0.014798  5.509007  0.010795]      [  0
28  196     [  0.678373  67.846667    0.097864]      [ 0.014701  4.91289   0.010842]      [  0
29  183     [  0.680024  68.216667    0.095833]      [ 0.014718  4.92237   0.010833]      [  0
30  192     [  0.679777  68.403333    0.09632 ]      [ 0.01532   4.79451   0.010881]      [  0
31  183     [  0.681877  68.123333    0.095653]      [ 0.013854  4.726675  0.010424]      [  0
32  163     [  0.682006  67.663333    0.09544 ]      [ 0.015592  4.843895  0.010679]      [  0
33  188     [  0.68259   67.616667    0.095714]      [ 0.017148  5.084262  0.01078 ]      [  0
34  194     [  0.683459  67.26        0.094917]      [ 0.017192  4.740506  0.01007 ]      [  0
35  190     [  0.685088  67.003333    0.093564]      [ 0.016554  5.17848   0.010562]      [  0
36  178     [  0.688469  65.69        0.094038]      [ 0.01628   5.286514  0.01074 ]      [  0
37  185     [  0.687644  65.54        0.094165]      [ 0.016646  5.167372  0.010711]      [  0
38  170     [  0.689826  64.486667    0.093035]      [ 0.017134  5.004314  0.01163 ]      [  0
39  198     [  0.688494  64.54        0.092361]      [ 0.017534  5.084788  0.011161]      [  0
40  176     [  0.68911   64.206667    0.093254]      [ 0.01855   5.059376  0.010896]      [  0
41  187     [  0.692418  62.876667    0.093929]      [ 0.018293  5.403837  0.009809]      [  0
42  178     [  0.695126  61.903333    0.094527]      [ 0.018711  4.964607  0.01013 ]      [  0
```

```
43  161      [  0.699744  60.753333   0.095559]      [ 0.019064  4.569371  0.010088]      [  0
44  171      [  0.702384  59.586667   0.097205]      [ 0.019461  3.899892  0.009401]      [  0
45  188      [  0.703221  59.126667   0.09687 ]      [ 0.020518  3.816275  0.009162]      [  0
46  190      [  0.707132  58.54       0.09736 ]      [ 0.018435  3.396331  0.008577]      [  0
47  182      [  0.704882  59.056667   0.097366]      [ 0.020474  3.740783  0.009494]      [  0
48  190      [  0.706111  58.703333   0.096083]      [ 0.019928  3.390475  0.010232]      [  0
49  166      [  0.706648  58.813333   0.094813]      [ 0.020248  3.640305  0.010795]      [  0
50  181      [  0.703603  58.86       0.092685]      [ 0.020489  3.425454  0.011632]      [  0
```

```
# SELECT GENETIC SELECTION VARIABLES
X_genetic_selection = X[X.columns[selector.support_]]
```

4 SCALE DATA

```
# MIN-MAX SCALER
scaler = MinMaxScaler()
# SCALE FILTER SELECTION DATA
X_filter = pd.DataFrame(scaler.fit_transform(X_filter), columns=X_filter.columns)
# SCALE PCA SELECTION DATA
X_pca = pd.DataFrame(scaler.fit_transform(X_pca), columns=X_pca.columns)
# SCALE GENETIC SELECTION DATA
X_genetic_selection = pd.DataFrame(
    scaler.fit_transform(X_genetic_selection), columns=X_genetic_selection.columns
)
```

5 CONCATENATE X AND Y

```
# CONCATENATE X AND Y
## FILTER SELECTION
molecular_descriptors_filter = pd.concat([X_filter, Y], axis=1)
## PCA SELECTION
molecular_descriptors_pca = pd.concat([X_pca, Y], axis=1)
## GENETIC SELECTION
molecular_descriptors_genetic_selection = pd.concat([X_genetic_selection, Y], axis=1)
```

6 PREPARE TEST DATA

```
# WE DO THE SAME WITH THE TEST DATA
molecular_descriptors_test = pd.read_csv(
    input_test_path + molecular_descriptors_test_file
)
## X TEST
X_test = molecular_descriptors_test.drop(columns=["activity"])
## Y TEST
Y_test = molecular_descriptors_test["activity"]

# TEST DATA FOR FILTER SELECTION
X_filter_test = X_test[X_filter.columns]

## MIN-MAX SCALER FOR X TEST
X_filter_test = pd.DataFrame(
    scaler.fit_transform(X_filter_test), columns=X_filter_test.columns
)

## CONCATENATE X AND Y
molecular_descriptors_filter_test = pd.concat([X_filter_test, Y_test], axis=1)

# TEST DATA FOR PCA SELECTION
## PCA TRANSFORMATION DATAFRAME
X_pca_test = pd.DataFrame(pca.transform(X_filter_test))

## MIN-MAX SCALER FOR X TEST
X_pca_test = pd.DataFrame(scaler.fit_transform(X_pca_test), columns=X_pca_test.columns)

## CONCATENATE X AND Y
molecular_descriptors_pca_test = pd.concat([X_pca_test, Y_test], axis=1)
```

```
# TEST DATA FOR GENETIC SELECTION
X_genetic_selection_test = X_test[X_genetic_selection.columns]

## MIN-MAX SCALER FOR X TEST
X_genetic_selection_test = pd.DataFrame(
    scaler.fit_transform(X_genetic_selection_test),
    columns=X_genetic_selection_test.columns,
)

## CONCATENATE X AND Y
molecular_descriptors_genetic_selection_test = pd.concat(
    [X_genetic_selection_test, Y_test], axis=1
)
```

7 SAVE DATA

```
# SAVE DATA FILTER SELECTION
## SAVE TRAIN DATA
molecular_descriptors_filter.to_csv(
    output_train_path + dataset_name + "_filter_training.csv", index=False
)
## SAVE TEST DATA
molecular_descriptors_filter_test.to_csv(
    output_test_path + dataset_name + "_filter_test.csv", index=False
)

# SAVE DATA PCA SELECTION
## SAVE TRAIN DATA
molecular_descriptors_pca.to_csv(
    output_train_path + dataset_name + "_pca_training.csv", index=False
)
## SAVE TEST DATA
molecular_descriptors_pca_test.to_csv(
    output_test_path + dataset_name + "_pca_test.csv", index=False
)

# SAVE DATA GENETIC SELECTION
## SAVE TRAIN DATA
molecular_descriptors_genetic_selection.to_csv(
    output_train_path + dataset_name + "_genetic_selection_training.csv", index=False
)
## SAVE TEST DATA
molecular_descriptors_genetic_selection_test.to_csv(
    output_test_path + dataset_name + "_genetic_selection_test.csv", index=False
)
```