PYTHON CODE USED TO PRODUCE FIGURE 7

Contents

#IMPORTING LIBRARIES

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

%matplotlib inline
```

#IMPORTING DATA

```
data = r'C:\Users\HP\Desktop\Dr. Auwal\MAN1\Supporting information_Unsupervised ML
data.csv'
```

```
df = pd.read_csv(data)

df
```

#COLUMN INDEX

```
df.columns
```

#HISTOGRAM PLOT OF DATA IN EACH COLUMN

```
df[cols].hist(layout=(1, len(cols)), figsize=(3 * len(cols), 3.5));

plt.rcParams['figure.dpi'] = 200
```

#PAIRWISE CORRELATIONS OF COLUMN DATA

```
df.corr()
```

#HEATMAP OF PAIRWISE CORRELATIONS

```
dcorr=df[cols].corr()

#dcorr

mask = np.zeros_like(dcorr)

#mask.shape

mask[np.triu_indices_from(mask)] = True

fig, ax = plt.subplots(figsize=(7,5))

sns.heatmap(dcorr, cmap=sns.diverging_palette(10, 145, n=100),

        vmin=-1, vmax=1, center=0, linewidths=1, annot=True, mask=mask, ax=ax);
```

#PRINCIPAL COMPONENT ANALYSIS (PCA)

```
#Defining dataframe columns as X

X=df[cols]
```

#STANDARDIZE DATA USING StandardScalar (applied for columns not rows)

```
scalar = StandardScaler()

X_t=scalar.fit_transform(X)

scalar.mean_

scalar.var_
```

```
X_t.round(4) #round up to 4 digits
```

#VIEW DATA Z-SCORES PER COLUMN

```
dz=pd.DataFrame(X_t.round(4), columns=[f'z_{c}' for c in cols])

dz
```

#CONCATENATE ORIGINAL DATA TO Z-SCORE DATA

```
pd.concat([df, dz], axis='columns')
```

#STANDARDIZE (CONTINUATION)

```
(df['% AGEs formed]-scalar.mean_[0])/np.sqrt(scalar.var_[0])

X_t[:, 0]

X_t[:, 0]

X_t[:, 0].mean().round(4)

np.sqrt(X_t[:, 0].var())

X.head()

X_t[:5]

X_t.shape

X_t.shape[1]

#pca = PCA(n_components=X_t.shape[1])

pca = PCA(n_components=4)

pca.fit_transform(X_t)
```

#SCREE PLOT (Shows the % of each PC)

```
def scree_plot(X, n_components, with_cumulative=False, show_data_label=False, figsize=(10, 7)):

  '''

  PCA scree plot with cumulative

  '''

  scalar = StandardScaler()
```

```
X_t=scalar.fit_transform(X)

max_components = min(X.shape)
x=np.arange(1, n_components+1)
pca = PCA(n_components=max_components)
pca.fit_transform(X_t)
y1=pca.explained_variance_ratio_[:n_components]
y2=np.cumsum(pca.explained_variance_ratio_)[:n_components]

plt.figure(figsize=figsize)

if n_components > 20:
   marker = None
else:
   marker = 'o'
if with_cumulative:
   plt.plot(x, y2, linestyle='--', marker=marker, label='cumulative')

   plt.plot(x, y1, linestyle='-', marker=marker, label='individual')
   plt.title('Explained variance ratio')
   plt.xlabel('Number of components')
   plt.ylabel('Proportion of variance explained')
   plt.legend()
   if with_cumulative:
      [plt.axhline(y=x1, color='.7', linestyle='--',) for x1 in [.8, .9, .95, 1]]
   plt.grid(axis='x')
```

```
    if show_data_label:
        for n, v, cv in zip(np.nditer(x, flags=['refs_ok']),
                            np.nditer(y1, flags=['refs_ok']),
                            np.nditer(y2, flags=['refs_ok'])):
            plt.text(n+.02, v+.02, f'{v*100:.2f}%', fontsize=10)
            if with_cumulative:
                plt.text(n+.02, cv+.02, f'{cv*100:.2f}%', fontsize=10)
scree_plot(X, 4, True, True)


pca.components_ #Eigenvectors
dpc=pd.DataFrame(pca.components_.T,
            index=cols,
            columns=[f'PC{n+1}' for n in range(pca.components_.shape[0])]).round(4)
dpc
```

#HEATMAPS OF PCs

```
sns.heatmap(dpc, cmap=sns.diverging_palette(10, 145, n=100), linewidths=1,
      center=0, annot=True, vmin=-1, vmax=1);


X_t[:5]
X_t.shape
pca.components_.T.shape
#multiply matrix
np.dot(X_t, pca.components_.T)[:5]
pca.transform(X_t[:5])
df.head()
X_t[:1]
pca.components_[:1]
```

```
np.sum(X_t[:1] * pca.components_[:1])

pca.n_components_

dd=pd.concat([pd.DataFrame(pca.transform(X_t),
                    columns=[f'PC{n}' for n in range(1, pca.n_components_ +1)]),
        df[['peptide']]], axis = 'columns')

dd.head()
```

#PCA PLOT (2D)

```
# pca = PCA(n_components=X.shape[1])

pca = PCA(n_components=2)


X_pca=pca.fit_transform(X_t)

X_pca.shape

X_pca[:5]


# pca = PCA(n_components=X.shape[1])

pca = PCA(n_components=2)


X_pca=pca.fit_transform(X_t)

X_pca.shape

X_pca[:5]


principalDf = pd.DataFrame(data = X_pca
        , columns = ['principal component 1', 'principal component 2'])


finalDf = pd.concat([principalDf, df[['peptide']]], axis = 1)


fig = plt.figure(figsize = (8,8))
```

```
ax = fig.add_subplot(1,1,1)
ax.set_xlabel('PC 1 48.05%', fontsize = 15)
ax.set_ylabel('PC 21.24%', fontsize = 15)
ax.set_xlim(-3, 3)
ax.set_ylim(-2, 2)
#ax.set_title('2 component PCA', fontsize = 20)
peptides = ['AVIAIMF', 'GPADPF', ‘NGDF’, ‘NGNW’, ‘GSH’]
colors = ['y', 'g',]
for peptide, color in zip(peptides,colors):
  indicesToKeep = finalDf['peptide'] == peptide
  ax.scatter(finalDf.loc[indicesToKeep, 'principal component 1']
           , finalDf.loc[indicesToKeep, 'principal component 2']
           , c = color
           , s = 50,)

ax.legend(peptides)
ax.grid()
```