S2 - HOPE kNN

November 30, 2020

0.1 Import data from DB.

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
dfOrg = pd.read_csv('hope_dataset_cleaned.csv')

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

```
1243
```

[3]:
```
dfOrg.head(10)
```

[3]:
```
   pedido.data.attributes.age pedido.data.attributes.diagnostic_main  \
0                        75.0                     FISTULA PERITONEAL
1                        75.0                     FISTULA PERITONEAL
2                        75.0                     FISTULA PERITONEAL
3                        75.0                     FISTULA PERITONEAL
4                        75.0                     FISTULA PERITONEAL
5                        75.0                     FISTULA PERITONEAL
6                        75.0                     FISTULA PERITONEAL
7                        75.0                     FISTULA PERITONEAL
8                        75.0                     FISTULA PERITONEAL
9                        75.0                     FISTULA PERITONEAL

  pedido.data.attributes.gender  articulo  respuesta.articlesRevisedYear  \
0                          male  27395425                           2018
1                          male  28560554                           2018
2                          male  28641726                           2017
3                          male  26245344                           2016
4                          male  28942543                           2018
5                          male  24782153                           2014
6                          male  28002229                           2018
7                          male  27505109                           2017
8                          male  24850546                           2015
9                          male  29371050                           2019

   respuesta.articlesRevisedMonth  \
0                               1
1                               4
2                              12
3                              12
4                               6
5                               6
6                               9
7                               4
8                               1
9                               4

                             respuesta.pubmed_keys  utilidad
0  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       1.0
1  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
2  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
3  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
4  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
5  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
6  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
7  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
8  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
9  Abdomen,Adenocarcinoma,Antiemetics,Blood Cultu…       NaN
```

0.2 Transform (factorice) from Categories to continuous atributes

Transform ‘pedido.data.attributes.diagnostic_main’ atribute

[4]:

```python
dfKNN = dfOrg

categoriesORGDiagnosticMain = dfKNN['pedido.data.attributes.diagnostic_main'].
 ↪value_counts()

print("total: " + str(categoriesORGDiagnosticMain.size))

categoriesORGDiagnosticMain
```

```
total: 31
```

[4]:

```
CETOACIDOSIS DIABETICA         250
REHABILITACION NEUROLOGICA     180
INFECCION DE PARTES BLANDAS    170
DOLOR ABDOMINAL                131
INSUFICIENCIA RESPIRATORIA      90
FISTULA PERITONEAL              40
REACCION ALERGICA               30
INFECCION URINARIA              30
DIFICULTAD RESPIRATORIA         30
```

```
SINDROME FEBRIL                 20
LEGRADO                         20
PROLAPSO                        20
ACV.ISQUEMICO                   20
CEFALEA INTENSA                 20
CA GASTRICO                     20
TORACOTOMIA                     11
ABDOMEN AGUDO                   11
DISNEA                          10
DERMOLIPECTOMIA                 10
ARTRITIS SEPTICA                10
DOLOR                           10
TEP                             10
DIABETES                        10
LUXACION COLUMNA CERVICAL       10
METRORRAGIA                     10
POLITRAUMATISMO                 10
HEMORRAGIA DIGESTIVA            10
ANEMIA                          10
NEUMONIA                        10
INSUFICIENCIA CARDIACA          10
ADENOMA DE PROSTATA             10
Name: pedido.data.attributes.diagnostic_main, dtype: int64
```

[5]:

```
dataDiagnosticMain, categoriesDiagnosticMain = pd.factorize(dfKNN['pedido.data.
 ↪attributes.diagnostic_main'])

dfKNN['pedido.data.attributes.diagnostic_main'] = dataDiagnosticMain
```

Transform 'gender' atribute

[6]:

```
dataGender, categoriesGender = pd.factorize(dfKNN['pedido.data.attributes.
 ↪gender'])

dfKNN['pedido.data.attributes.gender'] = dataGender
```

Transform 'respuesta.pubmed_keys' atribute

[7]:

```
categoriesORGPubMedKeys = dfKNN['respuesta.pubmed_keys'].value_counts()

print("total: " + str(categoriesORGPubMedKeys.size))
```

```
total: 80
```

[8]:

```
dataPubMedKeys, categoriesPubMedKeys = pd.factorize(dfKNN['respuesta.
 ↪pubmed_keys'])

dfKNN['respuesta.pubmed_keys'] = dataPubMedKeys
```

[9]:
```
dfKNN.head(10)
```

[9]:
```
   pedido.data.attributes.age  pedido.data.attributes.diagnostic_main  \
0                        75.0                                       0
1                        75.0                                       0
2                        75.0                                       0
3                        75.0                                       0
4                        75.0                                       0
5                        75.0                                       0
6                        75.0                                       0
7                        75.0                                       0
8                        75.0                                       0
9                        75.0                                       0

   pedido.data.attributes.gender  articulo  respuesta.articlesRevisedYear  \
0                              0  27395425                           2018
1                              0  28560554                           2018
2                              0  28641726                           2017
3                              0  26245344                           2016
4                              0  28942543                           2018
5                              0  24782153                           2014
6                              0  28002229                           2018
7                              0  27505109                           2017
8                              0  24850546                           2015
9                              0  29371050                           2019

   respuesta.articlesRevisedMonth  respuesta.pubmed_keys  utilidad
0                               1                      0       1.0
1                               4                      0       NaN
2                              12                      0       NaN
3                              12                      0       NaN
4                               6                      0       NaN
5                               6                      0       NaN
6                               9                      0       NaN
7                               4                      0       NaN
8                               1                      0       NaN
9                               4                      0       NaN
```

[10]:
```
print("age NaN => " + str(dfKNN[pd.isnull(dfKNN['pedido.data.attributes.age'])].
 ↪shape[0]))
print("diagnostic_main NaN => " + str(dfKNN[pd.isnull(dfKNN['pedido.data.
 ↪attributes.diagnostic_main'])].shape[0]))
print("gender NaN => " + str(dfKNN[pd.isnull(dfKNN['pedido.data.attributes.
 ↪gender'])].shape[0]))
print("articulo NaN => " + str(dfKNN[pd.isnull(dfKNN['articulo'])].shape[0]))
print("articlesRevisedYear NaN => " + str(dfKNN[pd.isnull(dfKNN['respuesta.
 ↪articlesRevisedYear'])].shape[0]))
```

```
print("articlesRevisedMonth NaN => " + str(dfKNN[pd.isnull(dfKNN['respuesta.
 ↪articlesRevisedMonth'])].shape[0]))
print("pubmed_keys NaN => " + str(dfKNN[pd.isnull(dfKNN['respuesta.
 ↪pubmed_keys'])].shape[0]))
print("utilidad NaN => " + str(dfKNN[pd.isnull(dfKNN['utilidad'])].shape[0]))
```

```
age NaN => 10
diagnostic_main NaN => 0
gender NaN => 0
articulo NaN => 0
articlesRevisedYear NaN => 0
articlesRevisedMonth NaN => 0
pubmed_keys NaN => 0
utilidad NaN => 1192
```

Remove row with age eq NaN

[11]:
```
dfKNN = dfKNN[pd.notnull(dfKNN['pedido.data.attributes.age'])]
```

0.3 Separe data by utilidad is defined

[12]:
```
dfDataSetComplete = dfKNN[pd.notnull(dfKNN['utilidad'])]

print(dfDataSetComplete.shape[0])

dfDataSetToPredict = dfKNN[pd.isnull(dfKNN['utilidad'])]

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

```
51
1182
```

[13]:
```
dfDataSetComplete.head(10)
```

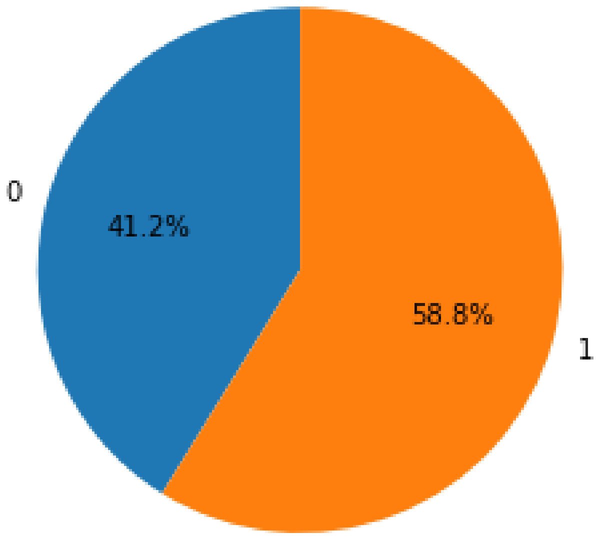
[13]:
```
     pedido.data.attributes.age  pedido.data.attributes.diagnostic_main  \
0                          75.0                                       0
32                         75.0                                       0
230                        36.0                                       6
290                        51.0                                      10
299                        51.0                                      10
300                        18.0                                      11
303                        18.0                                      11
304                        18.0                                      11
305                        18.0                                      11
311                        76.0                                      12

     pedido.data.attributes.gender  articulo  respuesta.articlesRevisedYear  \
0                                0  27395425                           2018
32                               0  28694230                           2017
230                              0  28805236                           2011
290                              0  27537587                           2011
299                              0  28148670                           2019
300                              0  25055513                           2019
303                              0  29279563                           2017
304                              0  29279563                           2017
305                              0  28065368                           2017
311                              0  30762794                           2019

     respuesta.articlesRevisedMonth  respuesta.pubmed_keys  utilidad
0                                 1                      0       1.0
32                               12                      3       1.0
230                               3                     21       0.0
290                               3                     23       0.0
299                               3                     23       1.0
300                               3                     24       1.0
303                               2                     24       0.0
304                               2                     24       0.0
305                              11                     24       1.0
311                               3                     25       1.0
```

0.4 Check distribution of “utilidad” attribute

[14]:

```
utilityValues = dfDataSetComplete['utilidad'].value_counts()

print(utilityValues)

import matplotlib.pyplot as plt

labels = '0', '1'
sizes = [utilityValues.get(0.0), utilityValues.get(1.0)]

fig1, ax1 = plt.subplots()
ax1.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
ax1.axis('equal')

plt.show()
```

```
1.0    30
0.0    21
Name: utilidad, dtype: int64
```



0.5 k-NN

[15]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from matplotlib.colors import ListedColormap
```

[16]:

```
# Discard 'articulo' because it is a identifier

X = dfDataSetComplete[['pedido.data.attributes.age',
       'pedido.data.attributes.diagnostic_main',
       'pedido.data.attributes.gender',
       'respuesta.articlesRevisedYear',
       'respuesta.articlesRevisedMonth',
       'respuesta.pubmed_keys']].values

y = dfDataSetComplete['utilidad'].values

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

[17]:

```
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[18]:

```
k_range = range(1, 20)
accuracy_weights_uniform = []
error_weights_uniform = []
```

```python
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k, weights='uniform', n_jobs=4)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy_weights_uniform.append(knn.score(X_test, y_test))
    error_weights_uniform.append(np.mean(y_pred != y_test))
```

[19]:

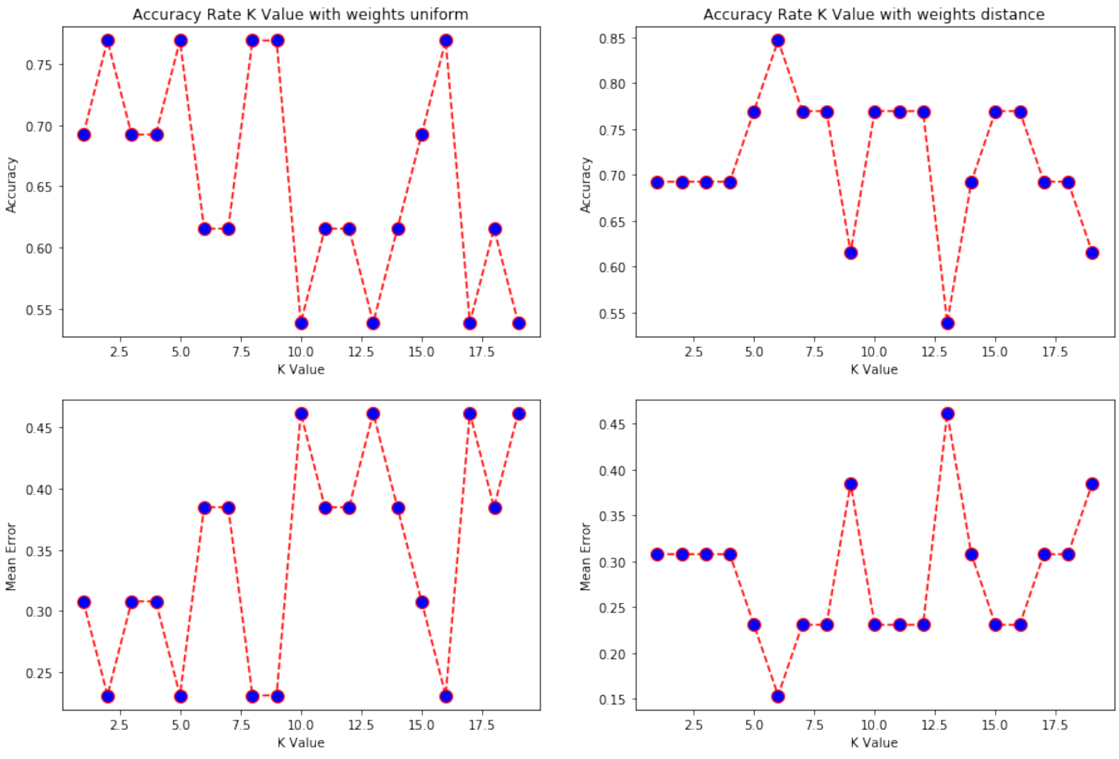
```python
k_range = range(1, 20)
accuracy_weights_distance = []
error_weights_distance = []
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k, weights='distance', n_jobs=4)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    accuracy_weights_distance.append(knn.score(X_test, y_test))
    error_weights_distance.append(np.mean(y_pred != y_test))
```

[20]:

```python
fig, axs = plt.subplots(2, 2,figsize=(15, 10))
axs[0, 0].plot(range(1, 20), accuracy_weights_uniform, color='red',␣
 ↪linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
axs[0, 0].set_title('Accuracy Rate K Value with weights uniform')
axs[0, 0].set_xlabel('K Value')
axs[0, 0].set_ylabel('Accuracy')
axs[0, 1].plot(range(1, 20), accuracy_weights_distance, color='red',␣
 ↪linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
axs[0, 1].set_title('Accuracy Rate K Value with weights distance')
axs[0, 1].set_xlabel('K Value')
axs[0, 1].set_ylabel('Accuracy')
axs[1, 0].plot(range(1, 20), error_weights_uniform, color='red',␣
 ↪linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
axs[1, 0].set_xlabel('K Value')
axs[1, 0].set_ylabel('Mean Error')
axs[1, 1].plot(range(1, 20), error_weights_distance, color='red',␣
 ↪linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
axs[1, 1].set_xlabel('K Value')
axs[1, 1].set_ylabel('Mean Error')
```

[20]:

```
Text(0, 0.5, 'Mean Error')
```



[21]:

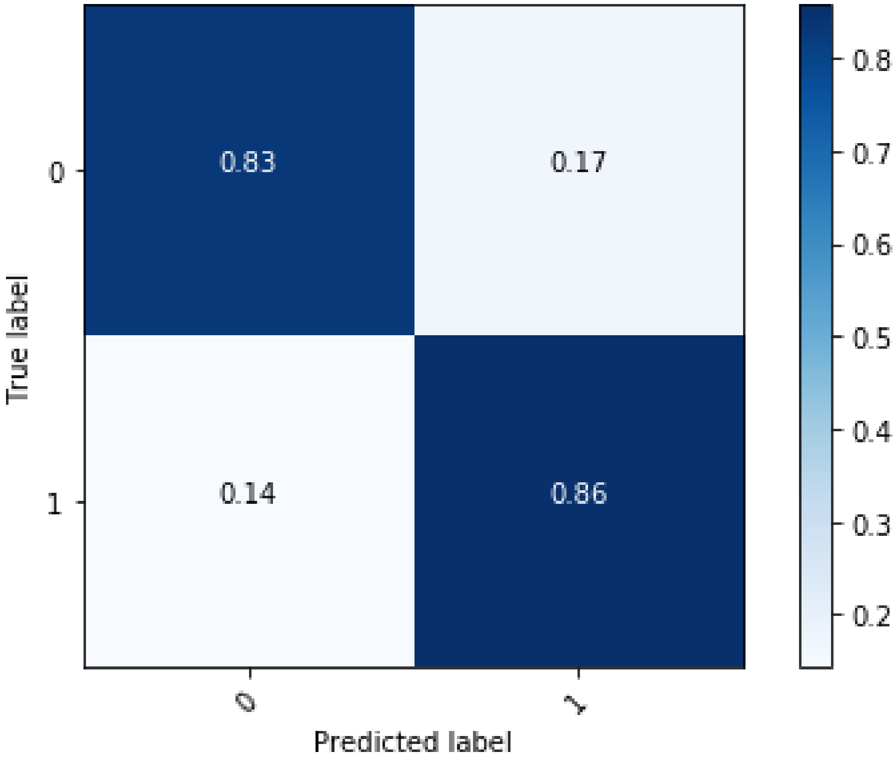
```
n_neighbors = 6

knn = KNeighborsClassifier(n_neighbors, weights='distance', n_jobs=4)
knn.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set: {:.2f}'
     .format(knn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set: {:.2f}'
     .format(knn.score(X_test, y_test)))
```

```
Accuracy of K-NN classifier on training set: 0.97
Accuracy of K-NN classifier on test set: 0.85
```

Show confusion matrix:

[22]:

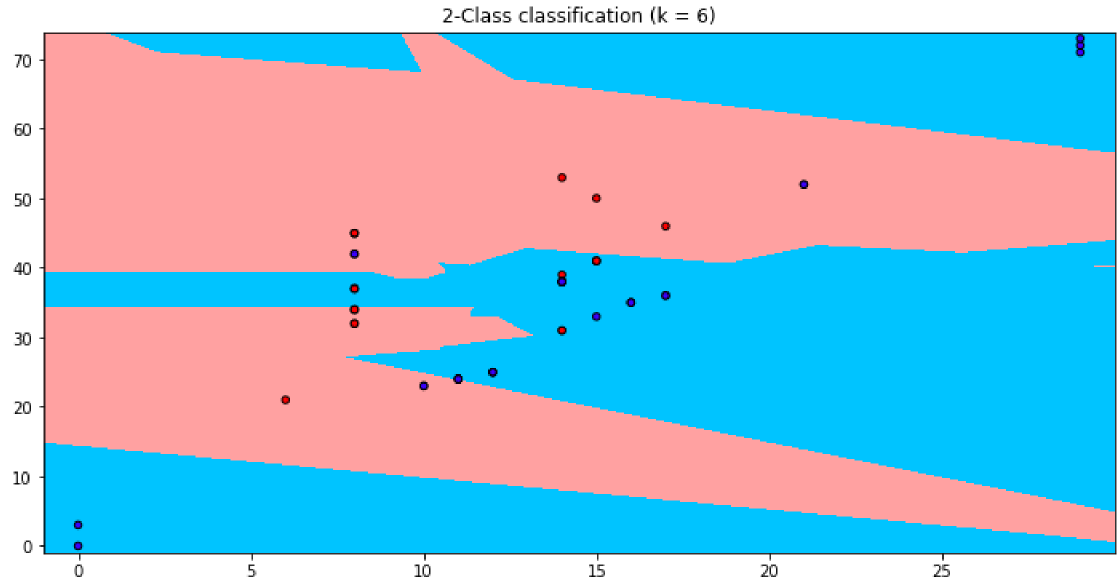
```
import itertools
from sklearn.metrics import confusion_matrix

preds = knn.predict(X_test)
cnf_matrix = confusion_matrix(y_test, preds)

def plot_confusion_matrix(cm, classes):
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    cmap=plt.cm.Blues
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], ".2f"),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

n_classes=["0","1"]
plot_confusion_matrix(cnf_matrix, classes=n_classes)
```



0.6 Print the K-NN classification only with the attributes “diagnostic_main” and “pubmed_keys”

[23]:

```
X_plot = dfDataSetComplete[['pedido.data.attributes.diagnostic_main',
       'respuesta.pubmed_keys']].values
y_plot = dfDataSetComplete['utilidad'].values

h = .02  # step size in the mesh

# Create color maps
cmap_light = ListedColormap(['#ffa1a1', '#00c4ff'])
cmap_bold = ListedColormap(['#ff0000', '#3a00ff'])

# we create an instance of Neighbours Classifier and fit the data.
clf = KNeighborsClassifier(n_neighbors)
clf.fit(X_plot, y_plot)

# Plot the decision boundary. For that, we will assign a color to each
# point in the mesh [x_min, x_max]x[y_min, y_max].
x_min, x_max = X_plot[:, 0].min() - 1, X_plot[:, 0].max() + 1
y_min, y_max = X_plot[:, 1].min() - 1, X_plot[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))
Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])

# Put the result into a color plot
Z = Z.reshape(xx.shape)
plt.figure(figsize=(12, 6))
plt.pcolormesh(xx, yy, Z, cmap=cmap_light)

# Plot also the training points
plt.scatter(X_plot[:, 0], X_plot[:, 1], c=y_plot, cmap=cmap_bold,
            edgecolor='k', s=20)
plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())
plt.title("2-Class classification (k = 6)")

plt.show()
```



0.7 Run Prediction

[24]:
```
def runPrediction(row):
    valuesrow = np.array([row.get(['pedido.data.attributes.age',
       'pedido.data.attributes.diagnostic_main',
       'pedido.data.attributes.gender',
       'respuesta.articlesRevisedYear',
       'respuesta.articlesRevisedMonth',
       'respuesta.pubmed_keys']).values])
    return knn.predict(valuesrow)

dfDataSetToPredict.apply(runPrediction, axis=1)
```

[24]:
```
1        [1.0]
2        [1.0]
3        [1.0]
4        [1.0]
5        [1.0]
         …
1238     [1.0]
1239     [1.0]
1240     [1.0]
1241     [1.0]
1242     [1.0]
Length: 1182, dtype: object
```

[ ]: