OPEN SOURCE SOFTWARE LAB (15B17CI575)

Lab Assignment 7 (Practice Lab)

Odd 2021

Week 11: 20 Nov- 26 Nov

Topic Coverage: Scikit-learn: Machine Learning in Python

Patil Amit Gurusidhappa

19104004

B11

Practice

import pandas as pd

COLUMNS = ['age', 'workclass', 'fnlwgt', 'education',

           'education\_num', 'marital', 'occupation', 'relationship',

           'race', 'sex', 'capital\_gain', 'capital\_loss', 'hours\_week',

           'native\_country', 'label']

features = ['age', 'workclass', 'fnlwgt', 'education',

            'education\_num', 'marital', 'occupation', 'relationship',

            'race', 'sex', 'capital\_gain', 'capital\_loss', 'hours\_week',

            'native\_country']

PATH = "https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data"

df\_train = pd.read\_csv(PATH, skipinitialspace=True,

                       names=COLUMNS, index\_col=False)

#List continuous features

CONTI\_FEATURES = df\_train.\_get\_numeric\_data()

# print(CONTI\_FEATURES)

CONTI\_FEATURES = df\_train.\_get\_numeric\_data()

# print(CONTI\_FEATURES)

"""

       age  fnlwgt  education\_num  capital\_gain  capital\_loss  hours\_week

0       39   77516             13          2174             0          40

1       50   83311             13             0             0          13

2       38  215646              9             0             0          40

3       53  234721              7             0             0          40

4       28  338409             13             0             0          40

...    ...     ...            ...           ...           ...         ...

32556   27  257302             12             0             0          38

32557   40  154374              9             0             0          40

32558   58  151910              9             0             0          40

32559   22  201490              9             0             0          20

32560   52  287927              9         15024             0          40

 """

df\_train[CONTI\_FEATURES.columns] = df\_train[CONTI\_FEATURES.columns].astype('float64')

print(df\_train.describe())

"""

                age        fnlwgt  education\_num  capital\_gain  capital\_loss    hours\_week

count  32561.000000  3.256100e+04   32561.000000  32561.000000  32561.000000  32561.000000

mean      38.581647  1.897784e+05      10.080679   1077.648844     87.303830     40.437456

std       13.640433  1.055500e+05       2.572720   7385.292085    402.960219     12.347429

min       17.000000  1.228500e+04       1.000000      0.000000      0.000000      1.000000

25%       28.000000  1.178270e+05       9.000000      0.000000      0.000000     40.000000

50%       37.000000  1.783560e+05      10.000000      0.000000      0.000000     40.000000

75%       48.000000  2.370510e+05      12.000000      0.000000      0.000000     45.000000

max       90.000000  1.484705e+06      16.000000  99999.000000   4356.000000     99.000000

"""

print(df\_train.describe(include='all'))

"""

 age workclass        fnlwgt education  ...  capital\_loss    hours\_week native\_country  label

count   32561.000000     32561  3.256100e+04     32561  ...  32561.000000  32561.000000          32561  32561

unique           NaN         9           NaN        16  ...           NaN           NaN             42      2

top              NaN   Private           NaN   HS-grad  ...           NaN           NaN  United-States  <=50K

freq             NaN     22696           NaN     10501  ...           NaN           NaN          29170  24720

mean       38.581647       NaN  1.897784e+05       NaN  ...     87.303830     40.437456            NaN    NaN

std        13.640433       NaN  1.055500e+05       NaN  ...    402.960219     12.347429            NaN    NaN

min        17.000000       NaN  1.228500e+04       NaN  ...      0.000000      1.000000            NaN    NaN

25%        28.000000       NaN  1.178270e+05       NaN  ...      0.000000     40.000000            NaN    NaN

50%        37.000000       NaN  1.783560e+05       NaN  ...      0.000000     40.000000            NaN    NaN

75%        48.000000       NaN  2.370510e+05       NaN  ...      0.000000     45.000000            NaN    NaN

max        90.000000       NaN  1.484705e+06       NaN  ...   4356.000000     99.000000            NaN    NaN

 """

conti\_features = []

for i in CONTI\_FEATURES:

    position = df\_train.columns.get\_loc(i)

    conti\_features.append(position)

print(conti\_features)

"""

[0, 2, 4, 10, 11, 12]

 """

Questions:

Q1.

import numpy as np

from scipy import sparse

eye = np.eye(4)

print("NumPy array:\n", eye)

sparse\_matrix = sparse.csr\_matrix(eye)

print("\nSciPy sparse CSR matrix:\n", sparse\_matrix)

Output

"""

NumPy array:

 [[1. 0. 0. 0.]

 [0. 1. 0. 0.]

 [0. 0. 1. 0.]

 [0. 0. 0. 1.]]

SciPy sparse CSR matrix:

   (0, 0)       1.0

  (1, 1)        1.0

  (2, 2)        1.0

  (3, 3)        1.0

 """

Q2.

import pandas as pd

data = pd.read\_csv("OpenSourceLab/assign-7/iris.csv")

print(data.describe())

Output

"""

       150\t4\tsetosa\tversicolor\tvirginica

count                                    150

unique                                   149

top                    5.8\t2.7\t5.1\t1.9\t2

freq                                       2

 """

Q3.

import pandas as pd

from sklearn.model\_selection import train\_test\_split

iris = pd.read\_csv("OpenSourceLab/assign-7/iris.csv")

#Drop id column

iris = iris.drop('setosa', axis=1)

X = iris.iloc[:, :-1].values

y = iris.iloc[:, 4].values

#Split arrays or matrices into random train and test subsets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

print("\n70% train data:")

print(X\_train)

print(y\_train)

print("\n30% test data:")

print(X\_test)

print(y\_test)

Q4.

# Import necessary modules

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

iris = pd.read\_csv("OpenSourceLab/assign-7/iris.csv")

#Drop id column

iris = iris.drop('Id', axis=1)

X = iris.iloc[:, :-1].values

y = iris.iloc[:, 4].values

#Split arrays or matrices into train and test subsets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

knn = KNeighborsClassifier(n\_neighbors=7)

knn.fit(X\_train, y\_train)

a\_index = list(range(1, 11))

a = pd.Series()

# Calculate the accuracy of the model for different values of k

for i in np.arange(1, 10):

    knn2 = KNeighborsClassifier(n\_neighbors=i)

    knn2.fit(X\_train, y\_train)

    print("For k = %d accuracy is" % i, knn2.score(X\_test, y\_test))

# Visual presentation: Various values of n for K-Nearest nerighbours

print("\nVisual presentation: Various values of n for K-Nearest nerighbours:")

for i in list(range(1, 11)):

    model = KNeighborsClassifier(n\_neighbors=i)

    model.fit(X\_train, y\_train)

    prediction = model.predict(X\_test)

    a = a.append(pd.Series(metrics.accuracy\_score(prediction, y\_test)))

plt.plot(a\_index, a)

Q5.

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn import metrics

from sklearn.linear\_model import LogisticRegression

iris = pd.read\_csv("iris.csv")

#Drop id column

iris = iris.drop('Id', axis=1)

X = iris.iloc[:, :-1].values

y = iris.iloc[:, 4].values

#Split arrays or matrices into train and test subsets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.20)

model = LogisticRegression(

    random\_state=0, solver='lbfgs', multi\_class='multinomial').fit(X, y)

model.fit(X\_train, y\_train)

prediction = model.predict(X\_test)

print('The accuracy of the Logistic Regression is',

      metrics.accuracy\_score(prediction, y\_test))