PS4

January 15, 2024

0.1 Data organization

0.2 Data loading

[1]:

```
pip install shap
```

```
Requirement already satisfied: shap in c:\programdata\anaconda3\lib\site-
packages (0.41.0)
Requirement already satisfied: slicer==0.0.7 in
c:\programdata\anaconda3\lib\site-packages (from shap) (0.0.7)
Requirement already satisfied: packaging>20.9 in
c:\programdata\anaconda3\lib\site-packages (from shap) (21.0)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-
packages (from shap) (1.7.1)
Requirement already satisfied: numpy in c:\programdata\anaconda3\lib\site-
packages (from shap) (1.20.3)
Requirement already satisfied: numba in c:\programdata\anaconda3\lib\site-
packages (from shap) (0.54.1)
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-
packages (from shap) (1.3.4)
Requirement already satisfied: cloudpickle in c:\programdata\anaconda3\lib\site-
packages (from shap) (2.0.0)
Requirement already satisfied: scikit-learn in
c:\programdata\anaconda3\lib\site-packages (from shap) (0.24.2)
Requirement already satisfied: tqdm>4.25.0 in c:\programdata\anaconda3\lib\site-
packages (from shap) (4.62.3)
Requirement already satisfied: pyparsing>=2.0.2 in
c:\programdata\anaconda3\lib\site-packages (from packaging>20.9->shap) (3.0.4)
Requirement already satisfied: colorama in c:\programdata\anaconda3\lib\site-
packages (from tqdm>4.25.0->shap) (0.4.4)
Requirement already satisfied: llvmlite<0.38,>=0.37.0rc1 in
c:\programdata\anaconda3\lib\site-packages (from numba->shap) (0.37.0)
Requirement already satisfied: setuptools in c:\programdata\anaconda3\lib\site-
packages (from numba->shap) (58.0.4)
Requirement already satisfied: python-dateutil>=2.7.3 in
c:\programdata\anaconda3\lib\site-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in
c:\programdata\anaconda3\lib\site-packages (from pandas->shap) (2021.3)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-
```

```
packages (from python-dateutil>=2.7.3->pandas->shap) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn->shap) (2.2.0)
Requirement already satisfied: joblib>=0.11 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn->shap) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

```
WARNING: Ignoring invalid distribution -cikit-learn
(c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn
(c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn
(c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn
(c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn
(c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -cikit-learn
(c:\programdata\anaconda3\lib\site-packages)
```

[1]:

```python
from shap.datasets import adult

X, y = adult()
print(X.shape)
print(y.shape)
print(type(X))
print(type(y))
```

```
(32561, 12)
(32561,)
<class 'pandas.core.frame.DataFrame'>
<class 'numpy.ndarray'>
```

[2]:

```python
print(X)
print(y)
```

```
        Age  Workclass  Education-Num  Marital Status  Occupation  \
0      39.0          7           13.0               4           1
1      50.0          6           13.0               2           4
2      38.0          4            9.0               0           6
3      53.0          4            7.0               2           6
4      28.0          4           13.0               2          10
…     …        …            …             …         …
32556  27.0          4           12.0               2          13
32557  40.0          4            9.0               2           7
32558  58.0          4            9.0               6           1
32559  22.0          4            9.0               4           1
32560  52.0          5            9.0               2           4
```

```
       Relationship  Race  Sex  Capital Gain  Capital Loss  Hours per week  \
0                 0     4    1        2174.0           0.0            40.0
1                 4     4    1           0.0           0.0            13.0
2                 0     4    1           0.0           0.0            40.0
3                 4     2    1           0.0           0.0            40.0
4                 5     2    0           0.0           0.0            40.0
…             …   …  …           …           …             …
32556             5     4    0           0.0           0.0            38.0
32557             4     4    1           0.0           0.0            40.0
32558             1     4    0           0.0           0.0            40.0
32559             3     4    1           0.0           0.0            20.0
32560             5     4    0       15024.0           0.0            40.0

       Country
0           39
1           39
2           39
3           39
4            5
…        …
32556       39
32557       39
32558       39
32559       39
32560       39

[32561 rows x 12 columns]
[False False False … False False  True]
```

[3]:

```python
print(X['Age'])
print(X['Age'].shape)
print(X['Age'].values)
print(type(X['Age'].values))
print(X['Age'].values.reshape(-1,1))
print(X['Age'].values.reshape(-1,1).shape)
```

```
0        39.0
1        50.0
2        38.0
3        53.0
4        28.0
         …
32556    27.0
32557    40.0
32558    58.0
32559    22.0
32560    52.0
Name: Age, Length: 32561, dtype: float32
```

```
(32561,)
[39. 50. 38. … 58. 22. 52.]
<class 'numpy.ndarray'>
[[39.]
 [50.]
 [38.]
 …
 [58.]
 [22.]
 [52.]]
(32561, 1)
```

[4]:
```
numerical_columns = ['Age','Education-Num','Capital Gain','Capital Loss','Hours␣
 ↪per week']
categorical_columns = ['Workclass','Marital␣
 ↪Status','Occupation','Relationship','Race','Sex','Country']
```

0.3 Conversion of categorical data

[5]:
```
import pandas as pd # for one-hot encoding
from sklearn.preprocessing import StandardScaler # for normalization
```

[6]:
```
# Normalization of numerical data
for column in numerical_columns:
    scaler = StandardScaler()
    X[column] = scaler.fit_transform(X[column].values.reshape(-1,1))
    #X[column] = scaler.fit_transform(X[column])


print(X)
print(type(X))
```

```
            Age  Workclass  Education-Num  Marital Status  Occupation  \
0      0.030671          7       1.134739               4           1
1      0.837109          6       1.134739               2           4
2     -0.042642          4      -0.420060               0           6
3      1.057047          4      -1.197459               2           6
4     -0.775768          4       1.134739               2          10
…         …        …            …             …         …
32556 -0.849080          4       0.746039               2          13
32557  0.103983          4      -0.420060               2           7
32558  1.423610          4      -0.420060               6           1
32559 -1.215643          4      -0.420060               4           1
32560  0.983734          5      -0.420060               2           4

       Relationship  Race  Sex  Capital Gain  Capital Loss  Hours per week  \
0                 0     4    1      0.148453      -0.21666       -0.035429
1                 4     4    1     -0.145920      -0.21666       -2.222153
```

```
2                  0     4    1     -0.145920      -0.21666       -0.035429
3                  4     2    1     -0.145920      -0.21666       -0.035429
4                  5     2    0     -0.145920      -0.21666       -0.035429
…               …    …   …           …            …             …
32556              5     4    0     -0.145920      -0.21666       -0.197409
32557              4     4    1     -0.145920      -0.21666       -0.035429
32558              1     4    0     -0.145920      -0.21666       -0.035429
32559              3     4    1     -0.145920      -0.21666       -1.655225
32560              5     4    0      1.888424      -0.21666       -0.035429

       Country
0           39
1           39
2           39
3           39
4            5
…          …
32556       39
32557       39
32558       39
32559       39
32560       39

[32561 rows x 12 columns]
<class 'pandas.core.frame.DataFrame'>
```

[7]:

```
# Data type change of categorical data
#   categorical data

for column in categorical_columns:
    X[column] = X[column].astype('category')

print(X)
print(X['Country'].values)
```

```
            Age Workclass  Education-Num Marital Status Occupation  \
0      0.030671         7       1.134739              4          1
1      0.837109         6       1.134739              2          4
2     -0.042642         4      -0.420060              0          6
3      1.057047         4      -1.197459              2          6
4     -0.775768         4       1.134739              2         10
…          …        …            …             …         …
32556 -0.849080         4       0.746039              2         13
32557  0.103983         4      -0.420060              2          7
32558  1.423610         4      -0.420060              6          1
32559 -1.215643         4      -0.420060              4          1
32560  0.983734         5      -0.420060              2          4
```

```
      Relationship Race Sex  Capital Gain  Capital Loss  Hours per week  \
0                0    4   1      0.148453      -0.21666       -0.035429
1                4    4   1     -0.145920      -0.21666       -2.222153
2                0    4   1     -0.145920      -0.21666       -0.035429
3                4    2   1     -0.145920      -0.21666       -0.035429
4                5    2   0     -0.145920      -0.21666       -0.035429
…            …  …  ..           …           …             …
32556            5    4   0     -0.145920      -0.21666       -0.197409
32557            4    4   1     -0.145920      -0.21666       -0.035429
32558            1    4   0     -0.145920      -0.21666       -0.035429
32559            3    4   1     -0.145920      -0.21666       -1.655225
32560            5    4   0      1.888424      -0.21666       -0.035429

      Country
0          39
1          39
2          39
3          39
4           5
…       …
32556      39
32557      39
32558      39
32559      39
32560      39

[32561 rows x 12 columns]
[39, 39, 39, 39, 5, …, 39, 39, 39, 39, 39]
Length: 32561
Categories (42, int64): [0, 1, 2, 3, …, 38, 39, 40, 41]
```

[8]:

```
X = pd.get_dummies(X)
print(X)
```

```
            Age  Education-Num  Capital Gain  Capital Loss  Hours per week  \
0      0.030671       1.134739      0.148453      -0.21666       -0.035429
1      0.837109       1.134739     -0.145920      -0.21666       -2.222153
2     -0.042642      -0.420060     -0.145920      -0.21666       -0.035429
3      1.057047      -1.197459     -0.145920      -0.21666       -0.035429
4     -0.775768       1.134739     -0.145920      -0.21666       -0.035429
…         …            …           …           …             …
32556 -0.849080       0.746039     -0.145920      -0.21666       -0.197409
32557  0.103983      -0.420060     -0.145920      -0.21666       -0.035429
32558  1.423610      -0.420060     -0.145920      -0.21666       -0.035429
32559 -1.215643      -0.420060     -0.145920      -0.21666       -1.655225
32560  0.983734      -0.420060      1.888424      -0.21666       -0.035429

       Workclass_0  Workclass_1  Workclass_2  Workclass_3  Workclass_4  …  \
0                0            0            0            0            0  …
1                0            0            0            0            0  …
2                0            0            0            0            1  …
3                0            0            0            0            1  …
4                0            0            0            0            1  …
…            …          …          …          …          …  …
32556            0            0            0            0            1  …
32557            0            0            0            0            1  …
32558            0            0            0            0            1  …
32559            0            0            0            0            1  …
32560            0            0            0            0            0  …

       Country_32  Country_33  Country_34  Country_35  Country_36  Country_37  \
0               0           0           0           0           0           0
1               0           0           0           0           0           0
2               0           0           0           0           0           0
3               0           0           0           0           0           0
4               0           0           0           0           0           0
…           …         …         …         …         …         …
32556           0           0           0           0           0           0
32557           0           0           0           0           0           0
32558           0           0           0           0           0           0
32559           0           0           0           0           0           0
32560           0           0           0           0           0           0

       Country_38  Country_39  Country_40  Country_41
0               0           1           0           0
1               0           1           0           0
2               0           1           0           0
3               0           1           0           0
4               0           0           0           0
…           …         …         …         …
32556           0           1           0           0
32557           0           1           0           0
32558           0           1           0           0
32559           0           1           0           0
32560           0           1           0           0

[32561 rows x 91 columns]
```

[9]:

```
# One-hot encoding of categorical data
X = pd.get_dummies(X)

# Conversion of data frame to numpy
X = X.values

# Converision: {False, True} --> {0., 1.}
```

```
y = y.astype(float)
```

[10]:

```
print(X.shape)
print(y.shape)
#print(X)
#print(y)
print(max(X[:,42]))
```

```
(32561, 91)
(32561,)
1.0
```

0.4 train-val-test split

[11]:

```
from sklearn.model_selection import train_test_split

X_,X_test,y_,y_test = train_test_split(X,y,test_size=1/10,stratify=y)
#X_,X_test,y_,y_test = train_test_split(X,y,test_size=1/10)

X_train,X_val,y_train,y_val = train_test_split(X_,y_,test_size=1/9,stratify=y_)
#X_train,X_val,y_train,y_val = train_test_split(X_,y_,test_size=1/9)

print(X_train.shape)
print(X_val.shape)
print(X_test.shape)
print(sum(y_train)/y_train.shape)
print(sum(y_val)/y_val.shape)
print(sum(y_test)/y_test.shape)
```

```
(26048, 91)
(3256, 91)
(3257, 91)
[0.24082463]
[0.24078624]
[0.24071231]
```

0.5 Logistic regression

[12]:

```
from sklearn.linear_model import LogisticRegression
```

[13]:

```
model_LR = LogisticRegression()

# training
model_LR.fit(X_train, y_train)

# evaulation
val_acc = model_LR.score(X_val, y_val)
```

```
print(val_acc)
```

```
0.8507371007371007
```

```
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
```

[14]:

```
from joblib import dump

dump(model_LR, 'LR_sample.joblib')
```

[14]:

```
['LR_sample.joblib']
```

[15]:

```
from joblib import load

loaded_model_LR = load('LR_sample.joblib')
```

[ ]:

```

```