chap7-cleaningData

June 19, 2022

[1]:

```python
# import pandas
import pandas as pd

# read the data using csv
data = pd.read_csv('employee.csv')

# see initial 5 records
data.head()
```

[1]:

```
            name   age   income gender  department grade  performance_score
0    Allen Smith  45.0      NaN    NaN  Operations    G3                723
1        S Kumar   NaN  16000.0      F     Finance    G0                520
2    Jack Morgan  32.0  35000.0      M     Finance    G2                674
3      Ying Chin  45.0  65000.0      F       Sales    G3                556
4  Dheeraj Patel  30.0  42000.0      F  Operations    G2                711
```

[2]:

```python
# see last 5 records
data.tail()
```

[2]:

```
            name   age   income gender  department grade  performance_score
4  Dheeraj Patel  30.0  42000.0      F  Operations    G2                711
5  Satyam Sharma   NaN  62000.0    NaN       Sales    G3                649
6   James Authur  54.0      NaN      F  Operations    G3                 53
7     Josh Wills  54.0  52000.0      F     Finance    G3                901
8       Leo Duck  23.0  98000.0      M       Sales    G4                709
```

[3]:

```python
# print list of columns in the data
print(data.columns)
```

```
Index(['name', 'age', 'income', 'gender', 'department', 'grade',
       'performance_score'],
      dtype='object')
```

[4]:

```python
# print the shape of a DataFrame
print(data.shape)
```

```
(9, 7)
```

[6]:

```
# check the information of DataFrame
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   name               9 non-null      object
 1   age                7 non-null      float64
 2   income             7 non-null      float64
 3   gender             7 non-null      object
 4   department         9 non-null      object
 5   grade              9 non-null      object
 6   performance_score  9 non-null      int64
dtypes: float64(2), int64(1), object(4)
memory usage: 632.0+ bytes
```

[7]:

```
# check the descriptive statistics
data.describe()
```

[7]:

```
             age        income  performance_score
count   7.000000      7.000000           9.000000
mean   40.428571  52857.142857         610.666667
std    12.204605  26028.372797         235.671912
min    23.000000  16000.000000          53.000000
25%    31.000000  38500.000000         556.000000
50%    45.000000  52000.000000         674.000000
75%    49.500000  63500.000000         711.000000
max    54.000000  98000.000000         901.000000
```

[8]:

```
# filter columns
data.filter(['name', 'department'])
```

[8]:

```
             name  department
0     Allen Smith  Operations
1         S Kumar     Finance
2     Jack Morgan     Finance
3       Ying Chin       Sales
4   Dheeraj Patel  Operations
5   Satyam Sharma       Sales
6    James Authur  Operations
7      Josh Wills     Finance
8        Leo Duck       Sales
```

[9]:

```
# filter column "name"
data['name']
```

```
[9]: 0      Allen Smith
     1          S Kumar
     2      Jack Morgan
     3        Ying Chin
     4    Dheeraj Patel
     5    Satyam Sharma
     6     James Authur
     7       Josh Wills
     8         Leo Duck
     Name: name, dtype: object
```

```
[10]: # filter column "name"
      data[['name']]
```

```
[10]:              name
      0    Allen Smith
      1        S Kumar
      2    Jack Morgan
      3      Ying Chin
      4  Dheeraj Patel
      5  Satyam Sharma
      6   James Authur
      7     Josh Wills
      8       Leo Duck
```

```
[11]: # filter two columns: name and department
      data[['name', 'department']]
```

```
[11]:              name  department
      0    Allen Smith  Operations
      1        S Kumar     Finance
      2    Jack Morgan     Finance
      3      Ying Chin       Sales
      4  Dheeraj Patel  Operations
      5  Satyam Sharma       Sales
      6   James Authur  Operations
      7     Josh Wills     Finance
      8       Leo Duck       Sales
```

```
[12]: # select rows for the specific index
      data.filter([0,1,2], axis = 0)
```

```
[12]:            name   age   income gender  department grade  performance_score
      0  Allen Smith  45.0      NaN    NaN  Operations    G3                723
      1      S Kumar   NaN  16000.0      F     Finance    G0                520
      2  Jack Morgan  32.0  35000.0      M     Finance    G2                674
```

[13]:

```
# filter using slicing
data[2:5]
```

[13]:

```
            name   age   income gender  department grade  performance_score
2    Jack Morgan  32.0  35000.0      M     Finance    G2                674
3      Ying Chin  45.0  65000.0      F       Sales    G3                556
4  Dheeraj Patel  30.0  42000.0      F  Operations    G2                711
```

[14]:

```
# filter data for specific value
data[data.department=='Sales']
```

[14]:

```
            name   age   income gender department grade  performance_score
3      Ying Chin  45.0  65000.0      F      Sales    G3               556
5  Satyam Sharma   NaN  62000.0    NaN      Sales    G3               649
8       Leo Duck  23.0  98000.0      M      Sales    G4               709
```

[16]:

```
# select data from multiple values
data[data.department.isin(['Sales', 'Finance'])]
```

[16]:

```
            name   age   income gender department grade  performance_score
1        S Kumar   NaN  16000.0      F    Finance    G0               520
2    Jack Morgan  32.0  35000.0      M    Finance    G2               674
3      Ying Chin  45.0  65000.0      F      Sales    G3               556
5  Satyam Sharma   NaN  62000.0    NaN      Sales    G3               649
7     Josh Wills  54.0  52000.0      F    Finance    G3               901
8       Leo Duck  23.0  98000.0      M      Sales    G4               709
```

[17]:

```
# filter employee who has more than 700 performance score
data[(data.performance_score >= 700)]
```

[17]:

```
            name   age   income gender  department grade  performance_score
0    Allen Smith  45.0      NaN    NaN  Operations    G3                723
4  Dheeraj Patel  30.0  42000.0      F  Operations    G2                711
7     Josh Wills  54.0  52000.0      F     Finance    G3                901
8       Leo Duck  23.0  98000.0      M       Sales    G4                709
```

[18]:

```
# filter employee who has more than 500 and less than 700 perfomance score
data[(data.performance_score >= 500) & (data.performance_score < 700)]
```

[18]:

```
            name   age   income gender department grade  performance_score
1        S Kumar   NaN  16000.0      F    Finance    G0               520
2    Jack Morgan  32.0  35000.0      M    Finance    G2               674
3      Ying Chin  45.0  65000.0      F      Sales    G3               556
5  Satyam Sharma   NaN  62000.0    NaN      Sales    G3               649
```

[19]:

```
# filter employee who has performance score of less than 500
data.query('performance_score<500')
```

[19]:
```
             name   age  income gender  department grade  performance_score
6  James Authur  54.0     NaN      F  Operations    G3                 53
```

[20]:
```
# drop missing value rows using dropna() func
# read the data

data = pd.read_csv('employee.csv')
data = data.dropna()
data
```

[20]:
```
              name   age   income gender  department grade  performance_score
2     Jack Morgan  32.0  35000.0      M     Finance    G2                674
3       Ying Chin  45.0  65000.0      F       Sales    G3                556
4   Dheeraj Patel  30.0  42000.0      F  Operations    G2                711
7      Josh Wills  54.0  52000.0      F     Finance    G3                901
8        Leo Duck  23.0  98000.0      M       Sales    G4                709
```

[21]:
```
# read the data
data = pd.read_csv('employee.csv')

# fill all the missing value in the age column with mean
# of the age column
data['age'] = data.age.fillna(data.age.mean())
```

[22]:
```
data
```

[22]:
```
              name        age   income gender  department grade  \
0     Allen Smith  45.000000      NaN    NaN  Operations    G3
1         S Kumar  40.428571  16000.0      F     Finance    G0
2     Jack Morgan  32.000000  35000.0      M     Finance    G2
3       Ying Chin  45.000000  65000.0      F       Sales    G3
4   Dheeraj Patel  30.000000  42000.0      F  Operations    G2
5   Satyam Sharma  40.428571  62000.0    NaN       Sales    G3
6    James Authur  54.000000      NaN      F  Operations    G3
7      Josh Wills  54.000000  52000.0      F     Finance    G3
8        Leo Duck  23.000000  98000.0      M       Sales    G4

   performance_score
0                723
1                520
2                674
3                556
4                711
5                649
6                 53
7                901
8                709
```

[23]:
```
# fill all the missing values in the income column with
# median of the income column
data['income'] = data.income.fillna(data.income.median())
data
```

[23]:
```
            name        age   income gender  department grade  \
0    Allen Smith  45.000000  52000.0    NaN  Operations    G3
1        S Kumar  40.428571  16000.0      F     Finance    G0
2    Jack Morgan  32.000000  35000.0      M     Finance    G2
3      Ying Chin  45.000000  65000.0      F       Sales    G3
4  Dheeraj Patel  30.000000  42000.0      F  Operations    G2
5  Satyam Sharma  40.428571  62000.0    NaN       Sales    G3
6   James Authur  54.000000  52000.0      F  Operations    G3
7     Josh Wills  54.000000  52000.0      F     Finance    G3
8       Leo Duck  23.000000  98000.0      M       Sales    G4

   performance_score
0                723
1                520
2                674
3                556
4                711
5                649
6                 53
7                901
8                709
```

[24]:
```
# fill all the missing values in the gender column
# (category column) with the mode of the gender column
data['gender'] = data['gender'].fillna(data['gender'].mode()[0])
data
```

[24]:
```
            name        age   income gender  department grade  \
0    Allen Smith  45.000000  52000.0      F  Operations    G3
1        S Kumar  40.428571  16000.0      F     Finance    G0
2    Jack Morgan  32.000000  35000.0      M     Finance    G2
3      Ying Chin  45.000000  65000.0      F       Sales    G3
4  Dheeraj Patel  30.000000  42000.0      F  Operations    G2
5  Satyam Sharma  40.428571  62000.0      F       Sales    G3
6   James Authur  54.000000  52000.0      F  Operations    G3
7     Josh Wills  54.000000  52000.0      F     Finance    G3
8       Leo Duck  23.000000  98000.0      M       Sales    G4

   performance_score
0                723
1                520
2                674
```

```
3                 556
4                 711
5                 649
6                  53
7                 901
8                 709
```

```
[26]: # handling outliers

      # dropping the outliers using Standard Deviation
      # read the data
      data = pd.read_csv('employee.csv')

      # dropping the outliers using Standard Deviation
      upper_limit = data['performance_score'].mean() + 3 * data['performance_score'].
       ↪std()
      lower_limit = data['performance_score'].mean() - 3 * data['performance_score'].
       ↪std()
```

```
[27]: data = data[(data['performance_score'] < upper_limit) &␣
       ↪(data['performance_score'] > lower_limit)]
      data
```

```
[27]:             name   age   income gender  department grade  performance_score
      0    Allen Smith  45.0      NaN    NaN  Operations    G3                723
      1        S Kumar   NaN  16000.0      F     Finance    G0                520
      2    Jack Morgan  32.0  35000.0      M     Finance    G2                674
      3      Ying Chin  45.0  65000.0      F       Sales    G3                556
      4  Dheeraj Patel  30.0  42000.0      F  Operations    G2                711
      5  Satyam Sharma   NaN  62000.0    NaN       Sales    G3                649
      6   James Authur  54.0      NaN      F  Operations    G3                 53
      7     Josh Wills  54.0  52000.0      F     Finance    G3                901
      8       Leo Duck  23.0  98000.0      M       Sales    G4                709
```

```
[28]: # read the data
      data = pd.read_csv('employee.csv')

      # drop the outlier observations using Percentiles
      upper_limit = data['performance_score'].quantile(.99)
      lower_limit = data['performance_score'].quantile(.01)
      data = data[(data['performance_score'] < upper_limit) &␣
       ↪(data['performance_score'] > lower_limit)]
      data
```

```
[28]:             name   age   income gender  department grade  performance_score
      0    Allen Smith  45.0      NaN    NaN  Operations    G3                723
      1        S Kumar   NaN  16000.0      F     Finance    G0                520
```

```
2    Jack Morgan  32.0  35000.0      M     Finance    G2                674
3      Ying Chin  45.0  65000.0      F       Sales    G3                556
4  Dheeraj Patel  30.0  42000.0      F  Operations    G2                711
5  Satyam Sharma   NaN  62000.0    NaN       Sales    G3                649
8       Leo Duck  23.0  98000.0      M       Sales    G4                709
```

[29]:

```
## feature encoding techniques

# one-hot encoding

# import one hot encoder
from sklearn.preprocessing import OneHotEncoder

# initialize the one-hot encoder object
onehotencoder = OneHotEncoder()

# fill all the missing values in income column
# (category column) with mode of age column
data['gender'] = data['gender'].fillna(data['gender'].mode()[0])

# fit and transforms the gender column
onehotencoder.fit_transform(data[['gender']]).toarray()
```

[29]:

```
array([[1., 0.],
       [1., 0.],
       [0., 1.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [0., 1.]])
```

[30]:

```
# label encoding

# import pandas
import pandas as pd

# read the data
data = pd.read_csv('employee.csv')

# import LabelEncoder
from sklearn.preprocessing import LabelEncoder

# instantiate the Label Encoder Object
label_encoder = LabelEncoder()

# fit and transform the column
encoded_data = label_encoder.fit_transform(data['department'])
```

```python

# print the encoded
print(encoded_data)
```

```
[1 0 0 2 1 2 1 0 2]
```

[31]:

```python
# perfomance inverse encoding
inverse_encode = label_encoder.inverse_transform([0,0,1,2])
```

[32]:

```python
# print inverse encode
print(inverse_encode)
```

```
['Finance' 'Finance' 'Operations' 'Sales']
```

[33]:

```python
# ordinal encoder

# import pandas and OrdinalEncoder
import pandas as pd
from sklearn.preprocessing import OrdinalEncoder

# load the data
data = pd.read_csv('employee.csv')

# initialize OrginalEncoder with order
order_encoder = OrdinalEncoder(categories=['G0', 'G1', 'G2', 'G3', 'G4'])

# fit and transform the grade
data['grade_encoded'] = label_encoder.fit_transform(data['grade'])

# check top-5 records of the dataframe
data.head()
```

[33]:

```
            name   age   income gender  department grade  performance_score  \
0    Allen Smith  45.0      NaN    NaN  Operations    G3                723
1       S Kumar    NaN  16000.0      F     Finance    G0                520
2     Jack Morgan  32.0  35000.0      M     Finance    G2                674
3      Ying Chin  45.0  65000.0      F       Sales    G3                556
4  Dheeraj Patel  30.0  42000.0      F  Operations    G2                711

   grade_encoded
0              2
1              0
2              1
3              2
4              1
```

[34]:

```
## methods for feature scaling

# import StandardScaler (or z-score normalization)
from sklearn.preprocessing import StandardScaler

# initialize the StandardScaler
scaler = StandardScaler()

# to scale data
scaler.fit(data['performance_score'].values.reshape(-1,1))
data['performance_std_scaler'] = scaler.transform(data['performance_score'].
 ↪values.reshape(-1,1))
data.head()
```

[34]:

```
            name   age   income gender  department grade  performance_score  \
0    Allen Smith  45.0      NaN    NaN  Operations    G3                723
1        S Kumar   NaN  16000.0      F     Finance    G0                520
2    Jack Morgan  32.0  35000.0      M     Finance    G2                674
3      Ying Chin  45.0  65000.0      F       Sales    G3                556
4  Dheeraj Patel  30.0  42000.0      F  Operations    G2                711

   grade_encoded  performance_std_scaler
0              2                0.505565
1              0               -0.408053
2              1                0.285037
3              2               -0.246032
4              1                0.451558
```

[36]:

```
# min-max scaling

# import MinMaxScaler
from sklearn.preprocessing import MinMaxScaler

# initialise the MinMaxScaler
scaler = MinMaxScaler()

# to scale data
scaler.fit(data['performance_score'].values.reshape(-1,1))
data['performance_minmax_scaler'] = scaler.transform(data['performance_score'].
 ↪values.reshape(-1,1))
data.head()
```

[36]:

```
            name   age   income gender  department grade  performance_score  \
0    Allen Smith  45.0      NaN    NaN  Operations    G3                723
1        S Kumar   NaN  16000.0      F     Finance    G0                520
2    Jack Morgan  32.0  35000.0      M     Finance    G2                674
3      Ying Chin  45.0  65000.0      F       Sales    G3                556
```

```
4  Dheeraj Patel  30.0  42000.0      F  Operations    G2                711

   grade_encoded  performance_std_scaler  performance_minmax_scaler
0              2                0.505565                   0.790094
1              0               -0.408053                   0.550708
2              1                0.285037                   0.732311
3              2               -0.246032                   0.593160
4              1                0.451558                   0.775943
```

[37]:
```
# robust scaling

# import RobustScaler
from sklearn.preprocessing import RobustScaler

# initialise the RobustScaler
scaler = RobustScaler()

# to scale data
scaler.fit(data['performance_score'].values.reshape(-1,1))
data['performance_robust_scaler'] = scaler.transform(data['performance_score'].
  ↪values.reshape(-1,1))

# see initial 5 records
data.head()
```

[37]:
```
            name   age   income gender  department grade  performance_score  \
0    Allen Smith  45.0      NaN    NaN  Operations    G3                723
1        S Kumar   NaN  16000.0      F     Finance    G0                520
2    Jack Morgan  32.0  35000.0      M     Finance    G2                674
3      Ying Chin  45.0  65000.0      F       Sales    G3                556
4  Dheeraj Patel  30.0  42000.0      F  Operations    G2                711

   grade_encoded  performance_std_scaler  performance_minmax_scaler  \
0              2                0.505565                   0.790094
1              0               -0.408053                   0.550708
2              1                0.285037                   0.732311
3              2               -0.246032                   0.593160
4              1                0.451558                   0.775943

   performance_robust_scaler
0                   0.316129
1                  -0.993548
2                   0.000000
3                  -0.761290
4                   0.238710
```

[39]:

```
# read the data
data = pd.read_csv('employee.csv')

# create performance grade function
def performance_grade(score):
    if score >= 700:
        return 'A'
    elif score < 700 and score >= 500:
        return 'B'
    else:
        return 'C'

# apply performance grade function on whole DataFrame
# using apply() function
data['performance_grade'] = data.performance_score.apply(performance_grade)

# see initial 5 records
data.head()
```

[39]:

```
            name   age   income gender  department grade  performance_score  \
0    Allen Smith  45.0      NaN    NaN  Operations    G3                723
1        S Kumar   NaN  16000.0      F     Finance    G0                520
2    Jack Morgan  32.0  35000.0      M     Finance    G2                674
3      Ying Chin  45.0  65000.0      F       Sales    G3                556
4  Dheeraj Patel  30.0  42000.0      F  Operations    G2                711

  performance_grade
0                 A
1                 B
2                 B
3                 B
4                 A
```

[40]:

```
## feature splitting

# split the name column in first and last name
data['first_name'] = data.name.str.split(" ").map(lambda var: var[0])
data['last_name'] = data.name.str.split(" ").map(lambda var: var[1])

# check top-5 records
data.head()
```

[40]:

```
            name   age   income gender  department grade  performance_score  \
0    Allen Smith  45.0      NaN    NaN  Operations    G3                723
1        S Kumar   NaN  16000.0      F     Finance    G0                520
2    Jack Morgan  32.0  35000.0      M     Finance    G2                674
3      Ying Chin  45.0  65000.0      F       Sales    G3                556
```

```
4  Dheeraj Patel  30.0  42000.0      F  Operations    G2                711

  performance_grade first_name last_name
0                 A      Allen     Smith
1                 B          S     Kumar
2                 B       Jack    Morgan
3                 B       Ying      Chin
4                 A    Dheeraj     Patel
```

[ ]:

```

```