projectCleaning

December 7, 2022

1 Project Assignment - Cleaning

1. Reading the CSV file and printing information about data and null values
2. replace "?" with null values

1.1 1 Reading the CSV file

Reading the csv file and printing the information about the data

[ ]:
```
import pandas as pd
import numpy as np
from scipy.stats import skew

df_initial = pd.read_csv("project-2018-BRFSS-arthritis.csv")
```

```
Tuples (data pints:  11933
Attributes(variables:  108
```

```
/tmp/ipykernel_5088/4028896668.py:5: DtypeWarning: Columns (8,10,11) have mixed
types. Specify dtype option on import or set low_memory=False.
  df_initial = pd.read_csv("project-2018-BRFSS-arthritis.csv")
```

Normally we should not see any categorical data because all of our values already converted to numeric. As we see above error we have mixed data types because of the "?" symbol

[ ]:
```
print("Tuples (data pints: ", len(df_initial))
print("Attributes(variables: ", len(df_initial.columns))
print("there are ", len(df_initial.select_dtypes(exclude='number').columns), "␣
 ↪categorical attributes")
#df_initial.select_dtypes(exclude='number').columns
```

```
Tuples (data pints:  11933
Attributes(variables:  108
there are  37  categorical attributes
```

1. replace "?" with null values

We want to use pandas special functions to fill the null values. Therefore we will replace them with nan

[ ]:
```
df_modified_null = df_initial
df_modified_null= df_modified_null.replace("?",np.nan)
print(type(df_modified_null))
```

```
<class 'pandas.core.frame.DataFrame'>
```

[ ]:
```
print("any value is null", df_modified_null.isnull().values.any())
print("total tuples that missing ",df_modified_null.isnull().sum().sum())

missing_tuples = df_modified_null.isnull().sum()
missing_tuples= pd.DataFrame({'name':missing_tuples.index,␣
 ↪'total_missing_columns':missing_tuples.values})
notnull = df_modified_null.notna().sum()
notnull  =pd.DataFrame({'name':notnull.index, 'non_missing':notnull.values})
```

```
any value is null True
total tuples that missing  14830
```

[ ]:
```
data_info= missing_tuples.merge(notnull, how='left')
size= len(df_initial)
data_info["percentage_missing"]= round(data_info['total_missing_columns']/size,␣
 ↪2)
print("total columns that has missing values: ",␣
 ↪len(data_info[data_info["total_missing_columns"]!=0]))
#data_info[data_info["total_missing_columns"]!=0]
```

```
total columns that has missing values:  37
```

[ ]:
```
my_list= []
for items in data_info.name:
    my_list.append([df_modified_null[items].value_counts().to_dict()])

data_info["unique_items"]= my_list
data_info
```

[ ]:

	name	total_missing_columns	non_missing	percentage_missing	\
0	x.aidtst3	805	11128	0.07	
1	employ1	34	11899	0.00	
2	income2	114	11819	0.01	
3	weight2	180	11753	0.02	
4	height3	198	11735	0.02	
..	…	…	…	…	
103	x.michd	114	11819	0.01	
104	x.ltasth1	0	11933	0.00	
105	x.casthm1	0	11933	0.00	
106	x.state	0	11933	0.00	
107	havarth3	0	11933	0.00	

```
                                                unique_items
0                      [{'2': 7002, '1': 3657, '9': 469}]
1    [{'1': 4964, '7': 3505, '2': 1091, '8': 869, '…
2    [{'8': 3343, '7': 1592, '6': 1382, '99': 1117,…
3    [{'180': 564, '200': 535, '150': 514, '160': 4…
4    [{'506': 1050, '504': 966, '507': 950, '505': …
..                                                 …
103                          [{'2': 10748, '1': 1071}]
104                        [{1: 10233, 2: 1662, 9: 38}]
105                        [{1: 10708, 2: 1132, 9: 93}]
106  [{36: 936, 24: 487, 27: 464, 12: 417, 31: 382,…
107                                [{2: 7906, 1: 4027}]

[108 rows x 5 columns]
```

[ ]:
```
data_info[data_info["total_missing_columns"]!=0].name.to_list()
```

[ ]:
```
['x.aidtst3',
 'employ1',
 'income2',
 'weight2',
 'height3',
 'children',
 'veteran3',
 'blind',
 'renthom1',
 'marital',
 'educa',
 'deaf',
 'decide',
 'flushot6',
 'seatbelt',
 'hivtst6',
 'hivrisk5',
 'pneuvac4',
 'alcday5',
 'diffwalk',
 'usenow3',
 'diffdres',
 'diffalon',
 'smoke100',
 'persdoc2',
 'medcost',
 'checkup1',
 'x.metstat',
 'htin4',
```

```
 'wtkg3',
 'x.bmi5',
 'x.bmi5cat',
 'htm4',
 'x.race.g1',
 'x.urbstat',
 'x.chispnc',
 'x.michd']
```

[ ]:
```
df_2= df_modified_null
missing_columns= data_info[data_info["total_missing_columns"]!=0].name.to_list()

for item in missing_columns:
    df_2[item] =pd.to_numeric(df_2[item])
```

[ ]:
```
print("there are ", len(df_2.select_dtypes(exclude='number').columns), "␣
 ↪categorical attributes")
df_2.select_dtypes(exclude='number').columns
```

```
there are  0  categorical attributes
```

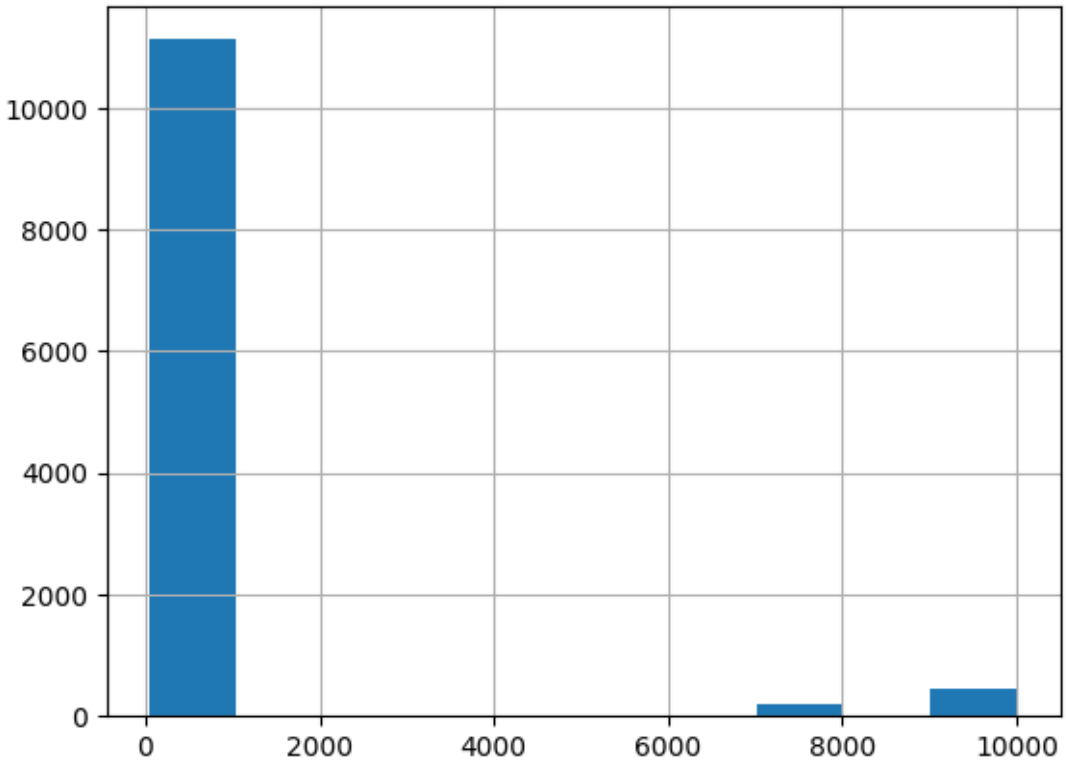
[ ]:
```
Index([], dtype='object')
```

[ ]:
```
total_percentage_missing = round(data_info.total_missing_columns.sum()/ df_2.
 ↪size,3)
print("Over all total percentage that is missing", total_percentage_missing,␣
 ↪"percent")
```

```
Over all total percentage that is missing 0.012 percent
```

[ ]:
```
# there are outliers
df_2.weight2.hist()
```

[ ]:
```
<AxesSubplot: >
```



[ ]:
```
# updating categorical variables with median
# we are using the median because the data includes a lot of outliers
df_3= df_2

df_3[['weight2', 'height3', 'htin4', 'wtkg3',  'x.bmi5',  'htm4']] =␣
 ↪df_3[['weight2', 'height3', 'htin4', 'wtkg3',  'x.bmi5',  'htm4']].
 ↪fillna(df_3[['weight2', 'height3', 'htin4', 'wtkg3',  'x.bmi5',  'htm4']].
 ↪median())
```

[ ]:
```
#df_3.loc[ : , ~df_3.columns.isin(['weight2', 'height3', 'htin4', 'wtkg3',  'x.
 ↪bmi5', 'htm4'])]
y = df_3.loc[ : , ~df_3.columns.isin(['weight2', 'height3', 'htin4', 'wtkg3', ␣
 ↪'x.bmi5', 'htm4'])]
my_list = y.columns.to_list()
for item in my_list:
    df_3[item] = df_3[item].fillna(df_3[item].mode()[0])
```

[ ]:
```
print("any value is null", df_3.isnull().values.any())
print("total tuples that missing ",df_3.isnull().sum().sum())
```

```
any value is null False
total tuples that missing  0
```

[ ]:
```
Q1=df_3.weight2.quantile(q=0.25)
Q2=df_3.weight2.quantile(q=0.50)
Q3=df_3.weight2.quantile(q=0.75)
print("Q1: ",Q1)
print("Q2: ",Q2)
print("Q3: ",Q3)



IQR= Q3-Q1
print("IQR: ",IQR)
lower_threshold = Q1 - (3*IQR)
print("lower threshold:", lower_threshold)
higher_threshold = Q3 + (3*IQR)
print("higher threshold:", higher_threshold)
IQR_outliers_low = df_3.weight2[df_3.weight2 < (lower_threshold) ]
IQR_outliers_high = df_3.weight2[df_3.weight2> (higher_threshold) ]
print("outliers that smaller than lower threshold: ", len(IQR_outliers_low))
print("outliers that smaller than bigger than higher_threshold: ",␣
 ↪len(IQR_outliers_high))
```

```
Q1:  150.0
Q2:  180.0
Q3:  211.0
IQR:  61.0
lower threshold: -33.0
higher threshold: 394.0
outliers that smaller than lower threshold:  0
outliers that smaller than bigger than higher_threshold:  651
```

[ ]:
```
# The 3(IQR) Criterion for Outliers
def outliersTreatment(df, col_name,whisker_width ):
    median = df[col_name].median()
    Q1=df[col_name].quantile(q=0.25)
    Q3=df[col_name].quantile(q=0.75)
    IQR= Q3-Q1
    lower_threshold = Q1 - (whisker_width*IQR)
    upper_threshold = Q3 + (whisker_width*IQR)
    print(lower_threshold)
    print(upper_threshold)
    df[col_name] = np.where(df[col_name] > upper_threshold, median,␣
 ↪df[col_name])
    df[col_name] = np.where(df[col_name] < lower_threshold, median,␣
 ↪df[col_name])
    return df
```

[ ]:
```
df_4 = outliersTreatment(df=df_3, col_name='weight2', whisker_width=3)

df_5= outliersTreatment(df=df_4, col_name='height3', whisker_width=3)

df_6= outliersTreatment(df=df_5, col_name='htin4', whisker_width=3)

df_7= outliersTreatment(df=df_6, col_name='x.bmi5', whisker_width=3)

# bmi seems like a calculated field but there is no time to chec
```

```
0.0
350.0
492.0
520.0
46.0
88.0
508.0
4988.0
```

[ ]:
```
df_8= outliersTreatment(df=df_7, col_name='htm4', whisker_width=3)
print(type(df_7.htm4))
```

```
118.0
223.0
<class 'pandas.core.series.Series'>
```