M2DataWrangling-lab

June 23, 2022

1 Data Wrangling Lab

Estimated time needed: **45 to 60** minutes

In this assignment you will be performing data wrangling.

1.1 Objectives

In this lab you will perform the following:

- Identify duplicate values in the dataset.
- Remove duplicate values from the dataset.
- Identify missing values in the dataset.
- Impute the missing values in the dataset.
- Normalize data in the dataset.

1.2 Hands on Lab

Import pandas module.

[48]:
```
import pandas as pd
```

Load the dataset into a dataframe.

[49]:
```
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.
↪cloud/IBM-DA0321EN-SkillsNetwork/LargeData/m1_survey_data.csv")
```

1.3 Finding duplicates

In this section you will identify duplicate values in the dataset.

Find how many duplicate rows exist in the dataframe.

[50]:
```
df.shape
```

[50]:
```
(11552, 85)
```

[51]:
```
# your code goes here
#use subset=None to check using all columns
```

```
#use keep=first so the first identical row is kept and the subsequent rows␣
 ↪identical to it are considered duplicates
df_dup = df[df.duplicated(subset=None, keep='first')].
 ↪sort_values(by=['Respondent'])
df_dup.shape
```

[51]: (154, 85)

1.4 Removing duplicates

Remove the duplicate rows from the dataframe.

[52]:
```
# your code goes here
df_new = df.drop_duplicates(subset =None, keep = 'first')
df_new.shape
```

[52]: (11398, 85)

Verify if duplicates were actually dropped.

[53]:
```
# your code goes here
df_new.shape[0] == df.shape[0] - df_dup.shape[0]
```

[53]: True

1.5 Finding Missing values

Find the missing values for all columns.

[54]:
```
# your code goes here
df_new.isnull().sum()
```

[54]:
```
Respondent          0
MainBranch          0
Hobbyist            0
OpenSourcer         0
OpenSource         81
                  ...
Sexuality         542
Ethnicity         675
Dependents        140
SurveyLength       19
SurveyEase         14
Length: 85, dtype: int64
```

Find out how many rows are missing in the column 'WorkLoc'

[55]:
```
# your code goes here
df_new['WorkLoc'].isnull().sum()
```

[55]:
```
32
```

1.6 Imputing missing values

Find the value counts for the column WorkLoc.

[56]:
```
# your code goes here
df_new['WorkLoc'].value_counts()
```

[56]:
```
Office                                                 6806
Home                                                   3589
Other place, such as a coworking space or cafe          971
Name: WorkLoc, dtype: int64
```

Identify the value that is most frequent (majority) in the WorkLoc column.

[57]:
```
#make a note of the majority value here, for future reference
mode = df_new['WorkLoc'].mode(dropna=True)
mode
#mode = 'Office'
```

[57]:
```
0    Office
dtype: object
```

Impute (replace) all the empty rows in the column WorkLoc with the value that you have identified as majority.

[58]:
```
# your code goes here
df_new["WorkLoc"].fillna('Office', inplace = True)
```

After imputation there should ideally not be any empty rows in the WorkLoc column.

Verify if imputing was successful.

[59]:
```
# your code goes here
df_new['WorkLoc'].value_counts()
```

[59]:
```
Office                                                 6838
Home                                                   3589
Other place, such as a coworking space or cafe          971
Name: WorkLoc, dtype: int64
```

1.7 Normalizing data

There are two columns in the dataset that talk about compensation.

One is “CompFreq”. This column shows how often a developer is paid (Yearly, Monthly, Weekly).

The other is "CompTotal". This column talks about how much the developer is paid per Year, Month, or Week depending upon his/her "CompFreq".

This makes it difficult to compare the total compensation of the developers.

In this section you will create a new column called 'NormalizedAnnualCompensation' which contains the 'Annual Compensation' irrespective of the 'CompFreq'.

Once this column is ready, it makes comparison of salaries easy.

List out the various categories in the column 'CompFreq'

[60]:
```
# your code goes here
df_new['CompFreq'].unique()
```

[60]:
```
array(['Yearly', 'Monthly', 'Weekly', nan], dtype=object)
```

Create a new column named 'NormalizedAnnualCompensation'. Use the hint given below if needed.

Double click to see the **Hint**.

[65]:
```
# your code goes here
# df.loc[df['column name'] condition, 'new column name'] = 'value if condition␣
 ↪is met'
df_new.loc[df_new['CompFreq'] == 'Yearly', 'NormalizedAnnualCompensation'] =␣
 ↪df_new['CompTotal']
df_new.loc[df_new['CompFreq'] == 'Monthly', 'NormalizedAnnualCompensation'] =␣
 ↪(df_new['CompTotal']) * 12
df_new.loc[df_new['CompFreq'] == 'Weekly', 'NormalizedAnnualCompensation'] =␣
 ↪(df_new['CompTotal']) * 52
```

```
/home/jupyterlab/conda/envs/python/lib/python3.7/site-
packages/pandas/core/indexing.py:1773: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_column(ilocs[0], value, pi)
```

[74]:
```
df_new[['CompTotal', 'CompFreq', 'NormalizedAnnualCompensation']]
```

[74]:

	CompTotal	CompFreq	NormalizedAnnualCompensation
0	61000.0	Yearly	61000.0
1	138000.0	Yearly	138000.0
2	90000.0	Yearly	90000.0
3	29000.0	Monthly	348000.0
4	90000.0	Yearly	90000.0
…	…	…	…
11547	130000.0	Yearly	130000.0

```
11548    74400.0   Yearly                     74400.0
11549   105000.0   Yearly                    105000.0
11550    80000.0   Yearly                     80000.0
11551        NaN      NaN                         NaN

[11398 rows x 3 columns]
```

[75]:
```
#check on some weekly paid rows to see if the math looks OK
# use this rough template to filter: rslt_df =␣
 ↪dataframe[dataframe['Percentage'] > 70]
df_new[df_new['CompFreq'] == 'Weekly'][['CompTotal', 'CompFreq',␣
 ↪'NormalizedAnnualCompensation']].head()
```

[75]:
```
      CompTotal CompFreq NormalizedAnnualCompensation
12       2000.0   Weekly                     104000.0
13      22000.0   Weekly                    1144000.0
46      67800.0   Weekly                    3525600.0
76     137000.0   Weekly                    7124000.0
135         NaN   Weekly                          NaN
```

1.8 Authors

Ramesh Sannareddy

1.8.1 Other Contributors

Rav Ahuja

1.9 Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-10-17	0.1	Ramesh Sannareddy	Created initial version of the lab