Homework 1 Report

Data Analysis and Machine Learning with Python

B09901061 莊政達

Dataset 1: Housing

A1.

1. Check if there is duplicated data and delete the duplicates.
2. Check if there is a missing value in the data. Handle the missing value according to a determined strategy. For example, delete the whole row or column if there is a missing value, replace the missing value with other data's mean, or simply fill the box with a "NAN" word.

A2. I received the following response from ChatGPT:

" 1. Handle Missing Values:

- Identify missing values in the dataset.

- Decide on a strategy to handle missing values, such as imputation (replacing missing values with a calculated or estimated value), deletion of rows or columns with missing values, or considering missing values as a separate category.

2. Check for Duplicates:

- Identify and remove duplicate rows from the dataset to avoid bias in the analysis.

3. Data Formatting:

- Standardize formats for consistency, such as date formats, numeric formats, and text cases.

4. Data Transformation:

- If necessary, transform variables by applying transformations like log transformation, scaling, or encoding categorical variables using techniques such

as one-hot encoding or label encoding.

5. Normalize or Standardize Data:

- Scale numerical features if needed to ensure they have similar ranges, especially when using algorithms sensitive to feature scales like gradient descent-based algorithms.

These steps are fundamental in ensuring that the data is clean, consistent, and ready for further analysis without introducing biases or errors due to missing values, duplicates, or inconsistent formats."

Ref: https://chat.openai.com/share/217c7ea7-8625-4f32-84fb-419b8f40c85d

The answers are reasonable since ChatGPT not only mentions handling missing values and duplicates but also more pre-processing to make the data more readable such as formatting.

However, it's worth mentioning that data transformation isn't suitable for every data. When data's uniqueness is the specialist, we should rather abandon the whole row or column than modify the value.

A3. According to the given data, I will set some conditions to decide which ones to rent. The following are the conditions that must be met:

1. Located in the city center.
2. More than 1 room.
3. The renting price must not exceed 75% of the available choices.
4. As near as possible to schools.

```
## in city center,>=2 rooms,,maximum +25% of renting price,miles ot school as near as possible
# Filter out the house locate in city center
dt_rent = dt.loc[(dt['Location'] == "City Center")]
# Filter out the house with 2 rooms
dt_rent = dt_rent.drop(dt_rent[dt_rent['No. of Rooms'] < 2].index)
# Get details of the house of the rent price
# print(dt_in_city['Rent Price per Month'].describe()) #75% = 17053 50% = 13446 mean = 13387
dt_rent = dt_rent.drop(dt_rent[dt_rent['Rent Price per Month'] > 17053].index)
# Get details of the house of the miles to school
dt_rent.sort_values(by=['Miles (dist. between school and house)'], inplace=True, ascending=True)
# print (dt_rent.head(3))
```

After analyzing the data, I prefer to choose 1 of the following 3 houses to rent, which is index 785,976, or 653 (in preference order).

	Area	No. of Rooms	No. of Bathrooms	Location	Miles (dist. between school and house)	Rent Price per Month	Sell Price
785	2041	2	1	City Center	19	8912	27709264
976	2918	2	1	City Center	22	6816	37785895
653	767	2	1	City Center	31	10195	69574747

A4. If I have enough funds to purchase a house, I would proceed with a purchase for sure. Just like choosing a house to rent, my conditions that must be met are as follows:

1. Located in the city center.
2. More than 1 room.
3. The renting price must not exceed 5.00e+07 (between 50% to 75% of the available choices)
4. As near as possible to schools.

```
##buying a house
dt_buy = dt.loc[(dt['Location'] == "City Center")]
dt_buy = dt_buy.drop(dt_buy[dt_buy["No. of Rooms"] < 2].index)
dt_buy = dt_buy.sort_values(by=["No. of Rooms"], ascending = False)
# print(dt_buy["Sell Price"].describe())
dt_buy = dt_buy.drop(dt_buy[dt_buy["Sell Price"] > 5.00e+07].index)
dt_buy.sort_values(by=['Miles (dist. between school and house)'], inplace=True, ascending=True)
# print (dt_buy.head(2))
```

After analyzing the data, I prefer to choose 1 of the following 2 houses to buy according to my budget. If I'm not tight on budget, I will purchase the house that is in index 275, otherwise, I will buy the house that is in index 785.

	Area	No. of Rooms	No. of Bathrooms	Location	Miles (dist. between school and house)	Rent Price per Month	Sell Price
275	2574	3	1	City Center	10	18602	48812486
785	2041	2	1	City Center	19	8912	27709264

A5.

For Sale: House 286 is unusually high. It is located in rural with 1 room and 1 bathroom. The area is only 516, which is much lower than the average area of houses in the data. However, it cost 77098148 and exceeds 75% of houses in the data.

	Area	No. of Rooms	No. of Bathrooms	Location	Miles (dist. between school and house)	Rent Price per Month	Sell Price
217	762	2	1	City Center	118	9326	79985777
992	2894	1	1	Rural	277	12811	79971622
55	1629	3	1	City Center	93	14386	79741006
172	2089	2	1	Suburb	241	15061	79657758
443	589	2	1	City Center	73	13365	79416381
526	2310	2	1	Suburb	350	17414	79414100
240	1943	1	1	Suburb	99	9318	79380135
528	1705	1	1	City Center	392	9093	79304060
291	2268	3	1	Suburb	485	19234	79199975
441	1253	1	1	City Center	378	17686	79103466
418	1848	1	1	Suburb	165	16744	78792724
815	2129	2	1	Suburb	257	18543	78630316
718	2468	1	1	Rural	364	10138	78598476
919	919	2	1	City Center	464	7673	78564347
270	2585	2	1	Rural	404	8161	78555305
831	1031	2	1	City Center	303	19652	78336392
355	1719	3	1	Suburb	409	14891	78217592
307	2316	2	1	City Center	459	8179	78216606
58	2949	2	1	City Center	433	8204	78079673
613	2724	1	1	City Center	258	18712	77967803
612	1762	3	1	Suburb	201	19779	77926775
839	2073	3	1	City Center	334	8668	77831204
162	2020	1	1	City Center	85	8743	77772618
163	2989	2	1	City Center	183	13866	77626426
215	1938	1	1	Suburb	450	9583	77582311
918	1422	2	1	City Center	389	10729	77284146
743	1025	2	1	Suburb	331	16442	77264168
412	2570	1	1	Suburb	389	14712	77224605
286	516	1	1	Rural	249	19891	77098148
57	1383	2	1	Suburb	290	14680	76931591

[redacted]

For Rent: House 169 is unusually high. It is located in the suburb with 2 rooms and 1 bathroom. The distance to a school is 317 miles, which isn't a short distance. However, the rent per month is 19912, which is the eighth highest rental price in the data. The price exceeds many houses that have better locations or bigger areas.

	Area	No. of Rooms	No. of Bathrooms	Location	Miles (dist. between school and house)	Rent Price per Month	Sell Price
16	2933	1	1	Suburb	152	19993	72607761
155	1183	3	1	Rural	17	19979	67303241
852	2719	2	1	Rural	171	19976	61413150
362	1135	1	1	Rural	428	19926	41980709
197	2294	3	1	Rural	81	19917	8298756
909	2938	3	1	Rural	50	19917	72147880
67	1775	3	1	Suburb	119	19913	64544710
169	501	2	1	Suburb	317	19912	62986402
840	2328	1	1	City Center	437	19907	70513077
286	516	1	1	Rural	249	19891	77098148

[redacted]

Dataset 2: Family

A1. I group the data by family to get the value of each family's total income and spending. Starting from adding columns to count family, adults, and children.

```
dt["Family_num"] = dt["Family"].apply(family_num)
dt["Adult_num"] = dt["Member"].str.contains("Adult")
dt["Child_num"] = dt["Member"].str.contains("Child")
# print(dt["Adult_num"])

# aggregate data at once
# combine memeber to as family
aggregation_function = {"Spend":"sum","Income":"sum","Family_num":"first","Adult_num":"sum","Child_num":"sum","Member":join_member}
dt = dt.groupby(dt["Family"]).aggregate(aggregation_function)
dt["Family_num"] = dt["Family_num"].astype(int)
dt["Family_num"] = dt["Family_num"].astype(int)
```

[illegible]

And then aggregate the data to create a more user-friendly data frame.

The example result is as follows

Family	Spend	Income	Family_num	Adult_num	Child_num	Member
family1	2129097	4761087	1	3	0	Adult1 Adult2 Adult3
family2	890424	2939887	2	3	2	Adult1 Adult2 Adult3 Child1 Child2
family3	807835	2301931	3	1	0	Adult1
family4	1128708	2896133	4	2	0	Adult1 Adult2
family5	501827	1428679	5	3	1	Adult1 Adult2 Adult3 Child1
...	...	...	...	...	...	...
family96	135954	325062	96	1	2	Adult1 Child1 Child2
family97	774694	2663794	97	2	0	Adult1 Adult2
family98	1031955	3018609	98	2	0	Adult1 Adult2
family99	493578	1827150	99	1	0	Adult1
family100	258414	1031646	100	2	0	Adult1 Adult2

[illegible]

Sort the data by total income, we know Family 6 has the highest annual income, which is 7804425. And Family 94 has the lowest annual income, which is 46790.

```
Highest annual income:
Spend                                  2879221
Income                                 7804425
Family_num                                   6
Adult_num                                    3
Child_num                                    2
Member        Adult1 Adult2 Adult3 Child1 Child2
```

[illegible]

```
Lowest annual income:
Spend                   30029
Income                  46790
Family_num                 94
Adult_num                   1
Child_num                   1
Member         Adult1 Child1
```

[illegible]

A2. According to the given data, no family doesn't possess adequate annual income to cover all members' spending. I determine this by searching if there is a family's spending > income and no result is found.

```
Spending exceed income:  Empty DataFrame
```

A3. Yes, there are 23 single-parent families and 27 childless families. To count single-parent families, I calculate the family with at least 1 child but only 1 parent at present.

```
dt_single = dt.loc[(dt['Adult_num'] == 1) & (dt['Child_num'] > 0)]
print (dt_single)
print ("Single parent family: ",dt_single['Family_num'].count())
```

```
Single parent family:  23
```

For childless families, I take the families with no children and the number of adults is less than 3.

```
dt_childless = dt.loc[((dt['Child_num'] == 0) & (dt['Adult_num'] < 3)) ]
print (dt_childless)
print ("Childless family: ",dt_childless['Family_num'].count())
```

```
Childless family:  27
```

A4. I don't see obvious errors like missing or duplicate data. However, I think the data could also include the age of the members, which could be more accurate when calculating information like single-parent families. For example, if a family has only 2 adults can't be determined whether it is a parent and grown children or a couple.

A5. I think ChatGPT or Bing can assist with the four questions above. With AI tools given suitable prompts and files, I believe it can solve the four questions completely. Besides the questions above and the files, I will also prompt it with more specific conditions like a single-parent family means there is only 1 parent taking care of at least 1 child, and a childless family doesn't include if there is a child who grows up and becomes an adult currently.