PS06

May 22, 2023

1 PS 06

1.1 Name: Xinyu Chang

---

[1]:
```
# import the packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.formula.api as smf
from sklearn.metrics import mean_squared_error
```

1.2 1 Who will win the elections?

1.2.1 1.1 Load and check

[2]:
```
election_df = pd.read_csv("us-elections_2000-2020.csv", sep='\t')
```

[3]:
```
election_df.shape
```

[3]:
```
(37390, 24)
```

[4]:
```
election_df.head()
```

[4]:
```
   FIPS  year    state state2   county     office        candidate       party  \
0  1001  2000  Alabama     AL  Autauga  President          Al Gore    democrat
1  1001  2000  Alabama     AL  Autauga  President  George W. Bush  republican
2  1001  2004  Alabama     AL  Autauga  President       John Kerry    democrat
3  1001  2004  Alabama     AL  Autauga  President  George W. Bush  republican
4  1001  2008  Alabama     AL  Autauga  President     Barack Obama    democrat

   candidatevotes  totalvotes  …  EDU600209D  POP010210D  POP220210D  \
0          4942.0       17208  …       31469       54571       42855
1         11993.0       17208  …       31469       54571       42855
2          4758.0       20081  …       31469       54571       42855
3         15196.0       20081  …       31469       54571       42855
4          6093.0       23641  …       31469       54571       42855
```

```
   POP250210D  POP320210D  POP400210D  PST110209D  BIRTHS2020  DEATHS2020  \
0        9643         474        1310        7085       143.0       168.0
1        9643         474        1310        7085       143.0       168.0
2        9643         474        1310        7085       143.0       168.0
3        9643         474        1310        7085       143.0       168.0
4        9643         474        1310        7085       143.0       168.0

   region
0   south
1   south
2   south
3   south
4   south

[5 rows x 24 columns]
```

[5]:
```
election_df.isna().sum()
```

[5]:
```
FIPS                 0
year                 0
state                0
state2               0
county               0
office               0
candidate            0
party                0
candidatevotes       4
totalvotes           0
income            6762
population        6762
LND010200D           0
EDU695209D           0
EDU600209D           0
POP010210D           0
POP220210D           0
POP250210D           0
POP320210D           0
POP400210D           0
PST110209D           0
BIRTHS2020          20
DEATHS2020          20
region               0
dtype: int64
```

There are 37390 rows and 24 columns in the dataset. The first 5 lines of data look reasonable and the data fits the type that the column defines. Also, there are 6762 missing values for the income column and the population column. There are 20 missing values for the BIRTHS2020 and

DEATHS2020 column.

1.2.2 1.2 Fill missings

1.Print the rows of the data frame from index 6264 to 6271 (i.e. these index values of the dataframe). For simplicity, you may only include variables fips, county, year and income.

[6]:
```
election_df.columns.get_loc('FIPS')
```

[6]:
```
0
```

[7]:
```
election_df.columns.get_loc('county')
```

[7]:
```
4
```

[8]:
```
election_df.columns.get_loc('year')
```

[8]:
```
1
```

[9]:
```
election_df.columns.get_loc('income')
```

[9]:
```
10
```

[10]:
```
print(election_df.iloc[6264: 6271, [0, 1, 4, 10]])
```

```
       FIPS  year        county   income
6264  15007  2016         Kauai  44958.0
6265  15007  2016         Kauai  44958.0
6266  15007  2020  Kauai County      NaN
6267  15007  2020  Kauai County      NaN
6268  15009  2000          Maui      NaN
6269  15009  2000          Maui      NaN
6270  15009  2004          Maui      NaN
```

2. You see that some income values are missing in the example from question 1.

(a) Which values do you expect to see instead of NA-s in lines 6266, 6267, 6268 and 6269? We can expect the following values instead of NAs in lines 6266, 6267, 6268, and 6269:

For lines 6266 and 6267 (Kauai County), the most recent available 'income' value is 44,958 from the year 2016 (index 6264 and 6265) due to their affiliation with the same county. For lines 6268 and 6269 (Maui County), we don't have enough data in the provided output to determine the most recent 'income' value.But the income value will be same for the line 6268 and line 6269 due to their affiliation with the same county.

(b)How is it related to the non-missing income, county and fips values? The missing 'income' values are related to the non-missing 'income', 'county', and 'FIPS' values because they help us determine the most recent 'income' data available for the corresponding county. Even though their county names may be different, we can still use the same FIP code to match between counties and fill in the missing income value because the same FIPS correspond the same "income" value.

(c)What method would you use to fill in the missings (what computer code and variables)? To fill in the missing 'income' values, we can first sort the DataFrame by 'FIPS' (county FIPS code) and 'year', and then use the fillna() method with the 'ffill' method (forward fill) within each group of 'county'.

3. Fill the missings in all columns you need (not only in income) with the most recent values that exist in the data. Ensure you do not fill missings with values from other counties.

[11]:
```
sorted_election = election_df.sort_values(by=['FIPS', 'year'])
sorted_election
```

[11]:
```
        FIPS  year    state state2          county     office          candidate  \
0       1001  2000  Alabama     AL         Autauga  President             Al Gore
1       1001  2000  Alabama     AL         Autauga  President     George W. Bush
2       1001  2004  Alabama     AL         Autauga  President         John Kerry
3       1001  2004  Alabama     AL         Autauga  President     George W. Bush
4       1001  2008  Alabama     AL         Autauga  President       Barack Obama
…        …   …       …     …             …          …                  …
37385  56045  2012  Wyoming     WY          Weston  President        Mitt Romney
37386  56045  2016  Wyoming     WY          Weston  President    Hillary Clinton
37387  56045  2016  Wyoming     WY          Weston  President       Donald Trump
37388  56045  2020  Wyoming     WY   Weston County  President       Donald Trump
37389  56045  2020  Wyoming     WY   Weston County  President       Joshep Biden

            party  candidatevotes  totalvotes  …  EDU600209D  POP010210D  \
0        democrat          4942.0       17208  …       31469       54571
1      republican         11993.0       17208  …       31469       54571
2        democrat          4758.0       20081  …       31469       54571
3      republican         15196.0       20081  …       31469       54571
4        democrat          6093.0       23641  …       31469       54571
…             …               …           …  …        …           …
37385  republican          2821.0        3359  …        4681        7208
37386    democrat           299.0        3526  …        4681        7208
37387  republican          3033.0        3526  …        4681        7208
37388  republican          3107.0        3542  …        4681        7208
37389    democrat           360.0        3542  …        4681        7208

       POP220210D  POP250210D  POP320210D  POP400210D  PST110209D  BIRTHS2020  \
0           42855        9643         474        1310        7085       143.0
1           42855        9643         474        1310        7085       143.0
2           42855        9643         474        1310        7085       143.0
3           42855        9643         474        1310        7085       143.0
4           42855        9643         474        1310        7085       143.0
```

```
...             ...        ...         ...        ...       ...        ...
37385          6885         21          20        216       365       16.0
37386          6885         21          20        216       365       16.0
37387          6885         21          20        216       365       16.0
37388          6885         21          20        216       365       16.0
37389          6885         21          20        216       365       16.0

       DEATHS2020  region
0           168.0   south
1           168.0   south
2           168.0   south
3           168.0   south
4           168.0   south
...           ...     ...
37385        14.0    west
37386        14.0    west
37387        14.0    west
37388        14.0    west
37389        14.0    west

[37390 rows x 24 columns]
```

[12]:
```
sorted_election['FIPS2'] = sorted_election['FIPS']
necessary_columns = ['income', 'population', 'LND010200D', 'EDU695209D',␣
 ↪'EDU600209D', 'BIRTHS2020', 'DEATHS2020']
for column in necessary_columns:
    sorted_election[column] = sorted_election.groupby('FIPS2')[column].
 ↪fillna(method='ffill')
print(sorted_election.head())
```

```
   FIPS  year    state state2   county     office       candidate       party  \
0  1001  2000  Alabama     AL  Autauga  President         Al Gore    democrat
1  1001  2000  Alabama     AL  Autauga  President  George W. Bush  republican
2  1001  2004  Alabama     AL  Autauga  President      John Kerry    democrat
3  1001  2004  Alabama     AL  Autauga  President  George W. Bush  republican
4  1001  2008  Alabama     AL  Autauga  President    Barack Obama    democrat

   candidatevotes  totalvotes  ...  POP010210D  POP220210D  POP250210D  \
0          4942.0       17208  ...       54571       42855        9643
1         11993.0       17208  ...       54571       42855        9643
2          4758.0       20081  ...       54571       42855        9643
3         15196.0       20081  ...       54571       42855        9643
4          6093.0       23641  ...       54571       42855        9643

   POP320210D  POP400210D  PST110209D  BIRTHS2020  DEATHS2020  region  FIPS2
0         474        1310        7085       143.0       168.0   south   1001
1         474        1310        7085       143.0       168.0   south   1001
```

```
2           474         1310        7085        143.0        168.0    south    1001
3           474         1310        7085        143.0        168.0    south    1001
4           474         1310        7085        143.0        168.0    south    1001

[5 rows x 25 columns]
```

4. Print out the same lines you did above in 1.1. Does it look what you expected? Pay close attention to the relationship between the FIPS code and the counties.

[13]:
```
print(sorted_election.iloc[6264: 6271, [0, 1, 4, 10]])
```

```
       FIPS  year        county   income
6264  15007  2016         Kauai  44958.0
6265  15007  2016         Kauai  44958.0
6266  15007  2020  Kauai County  44958.0
6267  15007  2020  Kauai County  44958.0
6268  15009  2000          Maui      NaN
6269  15009  2000          Maui      NaN
6270  15009  2004          Maui      NaN
```

In this output, we can see that the missing income values for Kauai County(with the FIPS 15007) have been filled with the most recent available value (44958.0), which is my expected result, but there are still missing values for Maui, as there were no earlier data points to fill those missing values.

1.2.3 1.3 Feature engineering

1.Make a new data frame that only contains 2020 data, and that contains a binary variable: whether or not democrats won in that county in 2020.

[14]:
```
election_2020 = sorted_election[sorted_election['year'] == 2020]
election_2020
```

[14]:
```
        FIPS  year    state state2           county     office      candidate  \
10      1001  2020  Alabama     AL   Autauga County  President  Donald Trump
11      1001  2020  Alabama     AL   Autauga County  President  Joshep Biden
22      1003  2020  Alabama     AL   Baldwin County  President  Donald Trump
23      1003  2020  Alabama     AL   Baldwin County  President  Joshep Biden
34      1005  2020  Alabama     AL   Barbour County  President  Donald Trump
…          …     …        …      …                …          …              …
37365  56041  2020  Wyoming     WY     Uinta County  President  Joshep Biden
37376  56043  2020  Wyoming     WY  Washakie County  President  Donald Trump
37377  56043  2020  Wyoming     WY  Washakie County  President  Joshep Biden
37388  56045  2020  Wyoming     WY    Weston County  President  Donald Trump
37389  56045  2020  Wyoming     WY    Weston County  President  Joshep Biden

            party  candidatevotes  totalvotes  …  POP010210D  POP220210D  \
10     republican         19838.0       27770  …       54571       42855
11       democrat          7503.0       27770  …       54571       42855
22     republican         83544.0      109679  …      182265      156153
```

```
23        democrat          24578.0        109679  …       182265       156153
34      republican           5622.0         10518  …        27457        13180
…            …                  …        …  …          …          …
37365     democrat           1591.0          9402  …        21118        19514
37376   republican           3245.0          4012  …         8533         7795
37377     democrat            651.0          4012  …         8533         7795
37388   republican           3107.0          3542  …         7208         6885
37389     democrat            360.0          3542  …         7208         6885

       POP250210D  POP320210D  POP400210D  PST110209D  BIRTHS2020  DEATHS2020  \
10           9643         474        1310        7085       143.0       168.0
11           9643         474        1310        7085       143.0       168.0
22          17105        1348        7992       39463       527.0       661.0
23          17105        1348        7992       39463       527.0       661.0
34          12875         107        1387         699        64.0       109.0
…             …         …         …         …         …          …
37365          55          61        1855        1185        50.0        49.0
37376          22          48        1162        -380        18.0        32.0
37377          22          48        1162        -380        18.0        32.0
37388          21          20         216         365        16.0        14.0
37389          21          20         216         365        16.0        14.0

       region  FIPS2
10      south   1001
11      south   1001
22      south   1003
23      south   1003
34      south   1005
…       …    …
37365    west  56041
37376    west  56043
37377    west  56043
37388    west  56045
37389    west  56045

[6222 rows x 25 columns]
```

[15]:
```python
democrats_votes = election_2020[election_2020['party'] == 'democrat'].
 ↪candidatevotes.values
democrats_votes
```

[15]:
```
array([ 7503., 24578.,  4816., …,  1591.,   651.,   360.])
```

[16]:
```python
republicans_votes = election_2020[election_2020['party'] == 'republican'].
 ↪candidatevotes.values
republicans_votes
```

[16]: array([19838., 83544., 5622., …, 7496., 3245., 3107.])

[17]:
```
democrats_won = democrats_votes > republicans_votes
```

[18]:
```
election_2020 = election_2020[election_2020.party == 'democrat']
election_2020['democrats_won'] = np.where(democrats_won == True, 1, 0)
election_2020.head()
```

[18]:
```
    FIPS  year    state state2          county     office      candidate  \
11  1001  2020  Alabama     AL  Autauga County  President  Joshep Biden
23  1003  2020  Alabama     AL  Baldwin County  President  Joshep Biden
35  1005  2020  Alabama     AL  Barbour County  President  Joshep Biden
47  1007  2020  Alabama     AL     Bibb County  President  Joshep Biden
59  1009  2020  Alabama     AL   Blount County  President  Joshep Biden

       party  candidatevotes  totalvotes  …  POP220210D  POP250210D  \
11  democrat          7503.0       27770  …       42855        9643
23  democrat         24578.0      109679  …      156153       17105
35  democrat          4816.0       10518  …       13180       12875
47  democrat          1986.0        9595  …       17381        5047
59  democrat          2640.0       27588  …       53068         761

    POP320210D  POP400210D  PST110209D  BIRTHS2020  DEATHS2020  region  FIPS2  \
11         474        1310        7085       143.0       168.0   south   1001
23        1348        7992       39463       527.0       661.0   south   1003
35         107        1387         699        64.0       109.0   south   1005
47          22         406        1698        62.0        90.0   south   1007
59         117        4626        7323       152.0       220.0   south   1009

    democrats_won
11              0
23              0
35              0
47              0
59              0

[5 rows x 26 columns]
```

2. Create auxiliary variables: population density (population divided by land area); and percentage of college graduates.

[19]:
```
election_2020['population_density'] = election_2020['population'] /␣
 ↪election_2020['LND010200D']  / 1000
election_2020['college_grad_percentage'] = (election_2020['EDU695209D'] /␣
 ↪election_2020['EDU600209D']) * 100
election_2020.head()
```

[19]:
```
    FIPS  year    state state2          county     office     candidate  \
11  1001  2020  Alabama     AL  Autauga County  President  Joshep Biden
23  1003  2020  Alabama     AL  Baldwin County  President  Joshep Biden
35  1005  2020  Alabama     AL  Barbour County  President  Joshep Biden
47  1007  2020  Alabama     AL     Bibb County  President  Joshep Biden
59  1009  2020  Alabama     AL   Blount County  President  Joshep Biden

       party  candidatevotes  totalvotes  …  POP320210D  POP400210D  \
11  democrat          7503.0       27770  …         474        1310
23  democrat         24578.0      109679  …        1348        7992
35  democrat          4816.0       10518  …         107        1387
47  democrat          1986.0        9595  …          22         406
59  democrat          2640.0       27588  …         117        4626

    PST110209D  BIRTHS2020  DEATHS2020  region  FIPS2  democrats_won  \
11        7085       143.0       168.0   south   1001              0
23       39463       527.0       661.0   south   1003              0
35         699        64.0       109.0   south   1005              0
47        1698        62.0        90.0   south   1007              0
59        7323       152.0       220.0   south   1009              0

    population_density  college_grad_percentage
11            0.091394                 7.261114
23            0.102421                 9.153772
35            0.028530                 5.295336
47            0.036071                 3.202429
59            0.088371                 4.057496

[5 rows x 28 columns]
```

3. Are countries with younger population more or less democratic? Compute (estimate) yearly birth rate and death rate. This is normally done as as the number of births/deaths per 1000 people per year, please do the same!

[20]:
```
election_2020['birth_rate'] = election_2020['BIRTHS2020'] /␣
 ↪election_2020['population'] * 1000 * 4
election_2020['death_rate'] = election_2020['DEATHS2020'] /␣
 ↪election_2020['population'] * 1000 * 4
election_2020.head()
```

[20]:
```
    FIPS  year    state state2          county     office     candidate  \
11  1001  2020  Alabama     AL  Autauga County  President  Joshep Biden
23  1003  2020  Alabama     AL  Baldwin County  President  Joshep Biden
35  1005  2020  Alabama     AL  Barbour County  President  Joshep Biden
47  1007  2020  Alabama     AL     Bibb County  President  Joshep Biden
59  1009  2020  Alabama     AL   Blount County  President  Joshep Biden
```

```
      party  candidatevotes  totalvotes  …  PST110209D  BIRTHS2020  \
11  democrat          7503.0       27770  …        7085       143.0
23  democrat         24578.0      109679  …       39463       527.0
35  democrat          4816.0       10518  …         699        64.0
47  democrat          1986.0        9595  …        1698        62.0
59  democrat          2640.0       27588  …        7323       152.0

    DEATHS2020  region  FIPS2  democrats_won  population_density  \
11       168.0   south   1001              0            0.091394
23       661.0   south   1003              0            0.102421
35       109.0   south   1005              0            0.028530
47        90.0   south   1007              0            0.036071
59       220.0   south   1009              0            0.088371

    college_grad_percentage  birth_rate  death_rate
11                 7.261114   10.354253   12.164437
23                 9.153772   10.154094   12.735969
35                 5.295336    9.920174   16.895296
47                 3.202429   10.980253   15.939077
59                 4.057496   10.575017   15.305945

[5 rows x 30 columns]
```

4. Ensure that the variables you are going to use are in a reasonable range!

[21]:
```
election_2020 = election_2020[election_2020['college_grad_percentage'] >= 0]
election_2020 = election_2020[election_2020['population_density'] >= 0]
election_2020 = election_2020[election_2020['LND010200D'] > 0]
election_2020 = election_2020.dropna()
election_2020.head()
```

[21]:
```
    FIPS  year    state state2           county     office     candidate  \
11  1001  2020  Alabama     AL   Autauga County  President  Joshep Biden
23  1003  2020  Alabama     AL   Baldwin County  President  Joshep Biden
35  1005  2020  Alabama     AL   Barbour County  President  Joshep Biden
47  1007  2020  Alabama     AL      Bibb County  President  Joshep Biden
59  1009  2020  Alabama     AL    Blount County  President  Joshep Biden

      party  candidatevotes  totalvotes  …  PST110209D  BIRTHS2020  \
11  democrat          7503.0       27770  …        7085       143.0
23  democrat         24578.0      109679  …       39463       527.0
35  democrat          4816.0       10518  …         699        64.0
47  democrat          1986.0        9595  …        1698        62.0
59  democrat          2640.0       27588  …        7323       152.0

    DEATHS2020  region  FIPS2  democrats_won  population_density  \
11       168.0   south   1001              0            0.091394
```

```
23       661.0    south    1003              0            0.102421
35       109.0    south    1005              0            0.028530
47        90.0    south    1007              0            0.036071
59       220.0    south    1009              0            0.088371

    college_grad_percentage  birth_rate  death_rate
11                 7.261114   10.354253   12.164437
23                 9.153772   10.154094   12.735969
35                 5.295336    9.920174   16.895296
47                 3.202429   10.980253   15.939077
59                 4.057496   10.575017   15.305945

[5 rows x 30 columns]
```

[22]:
```python
election_2020['population_density'].min(), election_2020['population_density'].
↪max()
```

[22]:
```
(0.00017285957006722317, 48.42887177968611)
```

[23]:
```python
election_2020['college_grad_percentage'].min(),␣
↪election_2020['college_grad_percentage'].max()
```

[23]:
```
(0.0, 37.00556242274413)
```

1.2.4 1.4 Model

1. Estimate logistic regression model where you explain democrats' winning with population density, education level, income, birth rate, death rate, and census region.

[24]:
```python
election_2020['income'] = election_2020['income'] / 1000
```

[25]:
```python
m = smf.logit('democrats_won ~ population_density + college_grad_percentage +␣
↪income + region + birth_rate + death_rate', \
              data=election_2020).fit()
m.get_margeff().summary()
```

```
Optimization terminated successfully.
         Current function value: 0.304644
         Iterations 8
```

[25]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
        Logit Marginal Effects
=====================================
Dep. Variable:          democrats_won
Method:                          dydx
At:                           overall
==============================================================================
```

```
===========
                            dy/dx    std err          z      P>|z|      [0.025
0.975]
--------------------------------------------------------------------------------
-----------
region[T.northeast]        0.1163      0.020      5.902      0.000       0.078
0.155
region[T.south]            0.0558      0.015      3.788      0.000       0.027
0.085
region[T.west]             0.1316      0.017      7.916      0.000       0.099
0.164
population_density         0.2155      0.023      9.455      0.000       0.171
0.260
college_grad_percentage    0.0240      0.002     12.301      0.000       0.020
0.028
income                    -0.0025      0.001     -3.866      0.000      -0.004
-0.001
birth_rate                -0.0013      0.002     -0.633      0.527      -0.006
0.003
death_rate                -0.0033      0.002     -1.914      0.056      -0.007
7.8e-05
================================================================================
===========
"""
```

2. Why do we use logistic regression here, instead of linear regression? We use logistic regression instead of linear regression because the dependent variable, Democrats winning, is a binary variable (0 or 1) representing a categorical outcome. Logistic regression is specifically designed for modeling binary outcomes and is appropriate when the response variable is not continuous and numerical value but rather belongs to a specific category or class. Linear regression, on the other hand, is suitable for modeling continuous and numerical outcomes.

3. Interpret the results. Which results are statistically significant? The reference category is midwest.

The coefficient for the variable region[T.northeast] is 0.1163. This means that, holding other variables constant, the Northeast region is approximately 11.63% more democratic.

The coefficient for the variable region[T.south] is 0.0558. Holding other variables constant, the South region is approximately 5.58% more democratic.

The coefficient for the variable region[T.west] is 0.1316. Holding other variables constant, the West region is approximately 13.16% more democratic.

The coefficient for the variable population density is 0.02155. Holding other variables constant, an increase in population density by 1000 people per square mile increases the probability of Democrats winning by 21.55%.

For the variable college grad percentage, the coefficient is 0.0240. Holding other variables constant, a one percentage point increase in the percentage of college graduates (so moving, for example, from 30% to 31%) increases the odds of Democrats winning by approximately 2.40%.

For the variable income, the coefficient is -0.0025. Holding other variables constant, a one-unit increase in income (with income measured in thousands of dollars) decreases the odds of Democrats winning by approximately 0.25%.

For the variable birth_rate, a one-unit increase (which means an increase by one birth per 1,000 people) decreases the probability of Democrats winning by 0.13%, all else being constant. However, this effect is not statistically significant (p-value is 0.527 which is greater than the typical significance level of 0.05), meaning we cannot be confident that the birth rate has a real effect on the probability of Democrats winning.

For the variable death_rate, a one-unit increase (which means an increase by one death per 1,000 people) decreases the probability of Democrats winning by 0.33%, all else being constant. However, this effect is not statistically significant (since its p-value is 0.056, which is greater than the typical significance level of 0.05), meaning we cannot be confident that the death rate has a real effect on the probability of Democrats winning.

Except for the birth rate(0.527) and death rate(0.056), all the coefficients have p-values less than 0.05(all with the value of 0.000) indicating that they are statistically significant at the 5% significance level. This suggests that population density, college grad percentage, income, and region are all significant factors in explaining Democrats winning in the 2020 elections. However, the death rate and birth rate are not significant factors in explaining Democrats winning in the 2020 elections.

1.3 2 Model AirBnB Price

1.3.1 2.1 Load an clean

1. Load data. I recommend to select only the variables you need below, bedrooms, price, and accommodates. You may return here again and change the variable selection as you need. Even better, check out the usecols argument for read_csv. Do the basic checks.

[26]:
```
air_df = pd.read_csv("airbnb-bangkok-listings.csv", usecols=["bedrooms",␣
↪"price", "accommodates","room_type"], sep = ",")
air_df.head()
```

[26]:

	room_type	accommodates	bedrooms	price
0	Entire home/apt	3	1.0	$1,845.00
1	Private room	2	1.0	$1,275.00
2	Private room	2	1.0	$800.00
3	Private room	2	1.0	$800.00
4	Private room	2	1.0	$1,845.00

[27]:
```
air_df.shape
```

[27]: (17040, 4)

[28]:
```
air_df.isna().sum()
```

[28]:
```
room_type          0
accommodates       0
bedrooms        1840
price              0
dtype: int64
```

There are 17040 rows and 4 columns in the dataset. The first 5 lines of data look reasonable and the data fits the type that the column defines. However, there are 1840 missing values for the "bedrooms" column.

[29]:
```python
air_df.bedrooms.value_counts()
```

[29]:
```
1.0     11923
2.0      2243
3.0       582
4.0       225
5.0        91
6.0        48
10.0       22
7.0        22
8.0        12
9.0         7
20.0        6
11.0        4
12.0        3
30.0        2
15.0        2
16.0        2
40.0        1
23.0        1
39.0        1
50.0        1
25.0        1
46.0        1
Name: bedrooms, dtype: int64
```

2. Do the basic data cleaning:

(a) convert price to numeric.

[30]:
```python
air_df["price"] = air_df["price"].replace('[\$\,\.]', '', regex=True).
 ↪astype(int)
air_df["price"] = (air_df["price"] / 100).apply(pd.to_numeric).astype(int)
air_df = air_df[air_df["price"] != 0]
air_df.sample(5)
```

[30]:
```
           room_type  accommodates  bedrooms  price
9053    Private room             2       1.0    700
1656 Entire home/apt             2       1.0   1342
12642   Private room             2       NaN   1438
7316     Shared room             2       1.0    800
5267     Shared room             1       1.0    400
```

(b) remove entries with missing or invalid price, bedrooms, and other variables you need below

[31]:
```
air_df.shape
```

[31]:
```
(17039, 4)
```

[32]:
```
air_df.describe()
```

[32]:
```
       accommodates      bedrooms          price
count  17039.000000  15200.000000   17039.000000
mean       3.134574      1.373421    2237.898996
std        2.201793      1.273414    7066.783256
min        1.000000      1.000000       6.000000
25%        2.000000      1.000000     757.000000
50%        2.000000      1.000000    1199.000000
75%        4.000000      1.000000    2000.000000
max       16.000000     50.000000  335482.000000
```

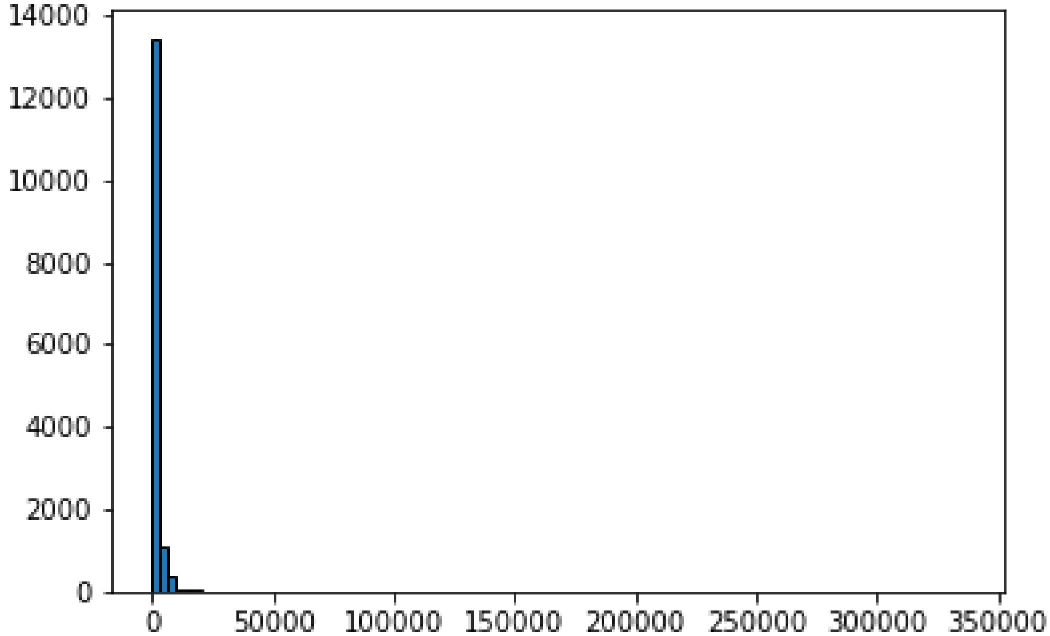
There are some missing values for the bedrooms. Maybe there is studio or the shared room so it doesn't contain the bedrooms.

[33]:
```
air_new = air_df[air_df.price > 0].dropna()
air_new.shape
```

[33]:
```
(15200, 4)
```

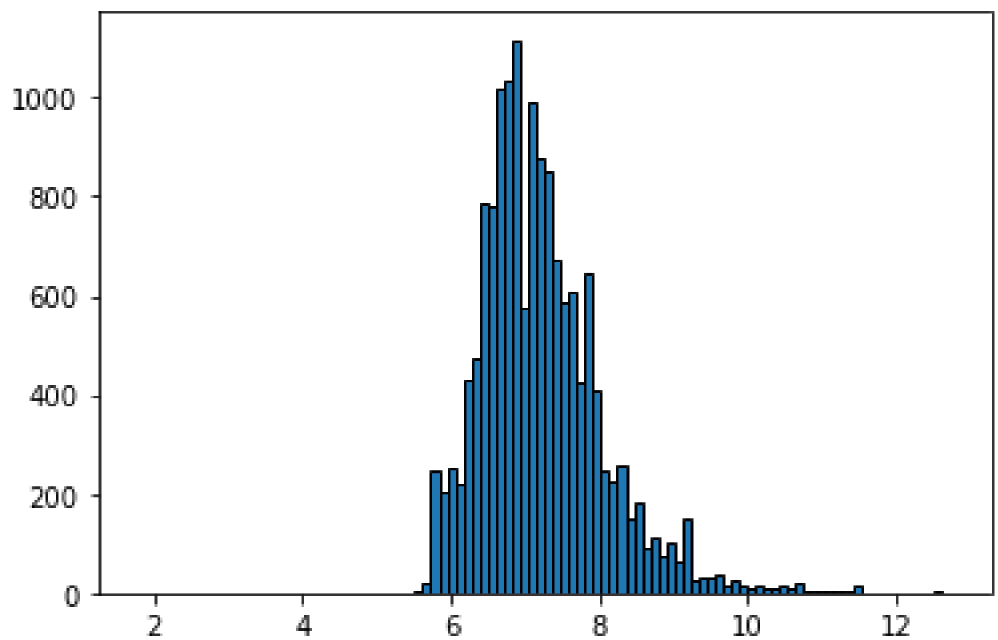
3. Analyze the distribution of price. Does it look like normal? Does it look like something else? Does it suggest you should do a log-transformation?

[34]:
```
price_his = plt.hist(air_new["price"], bins=100, edgecolor="black")
```



Based on the analysis of the distribution of the price variable, we can observe that it is highly right-skewed, meaning that the majority of prices are concentrated towards the lower end, with a long tail towards higher prices. This distribution does not resemble a normal distribution.The highly right-skewed distribution suggests that a log-transformation could be beneficial. By applying a log-transformation to the price variable, we can compress the range of values and reduce the skewness, making the distribution more symmetric and closer to a normal distribution.

[35]:
```
price_log_hist = plt.hist(np.log(air_new['price']), bins=100, edgecolor='black')
```



4. Convert the number of bedrooms into another variable with a limited number of categories only, such as 0, 1, 2, 3, 4+, and use these categories in the models below.

[36]:
```
air_new["new_bedroom_type"] = pd.cut(air_new.bedrooms,
                                     bins= [1, 2, 3, 4, np.inf],
                                     labels = ["1", "2", "3", "4+"],
                                     right = False)
air_new.head()
```

[36]:

	room_type	accommodates	bedrooms	price	new_bedroom_type
0	Entire home/apt	3	1.0	1845	1
1	Private room	2	1.0	1275	1
2	Private room	2	1.0	800	1
3	Private room	2	1.0	800	1
4	Private room	2	1.0	1845	1

[37]:
```
bedroom_counts = air_new["new_bedroom_type"].value_counts()
print(bedroom_counts)
```

```
1     11923
2      2243
3       582
4+      452
Name: new_bedroom_type, dtype: int64
```

1.3.2 2.2 Model

1. Run a linear regression where you explain the listing price with number of bedrooms where bedrooms uses these categories. Interpret the results, including R2.

[38]:
```
m_air = smf.ols('price ~ new_bedroom_type', data=air_new).fit()
m_air.summary()
```

[38]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:                  price   R-squared:                       0.036
Model:                            OLS   Adj. R-squared:                  0.035
Method:                 Least Squares   F-statistic:                     186.5
Date:                Fri, 19 May 2023   Prob (F-statistic):          9.32e-119
Time:                        06:10:39   Log-Likelihood:            -1.5589e+05
No. Observations:               15200   AIC:                         3.118e+05
Df Residuals:                   15196   BIC:                         3.118e+05
Df Model:                           3
Covariance Type:            nonrobust
================================================================================
==========
                             coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----------
Intercept               1742.2689     63.041     27.637      0.000    1618.701
1865.836
new_bedroom_type[T.2]   1481.0031    158.427      9.348      0.000    1170.466
1791.540
new_bedroom_type[T.3]   3317.3668    292.214     11.353      0.000    2744.591
3890.142
new_bedroom_type[T.4+]  6592.1869    329.856     19.985      0.000    5945.629
7238.745
==============================================================================
Omnibus:                    36718.532   Durbin-Watson:                   1.874
Prob(Omnibus):                  0.000   Jarque-Bera (JB):        557010065.623
Skew:                          25.064   Prob(JB):                         0.00
Kurtosis:                     939.470   Cond. No.                         6.02
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

The intercept term of 1742.2689 represents the expected listing price when the number of bedrooms is 1 (the reference category).

The R-squared value is 0.036, which means that approximately 3.6% of the variation in the listing price can be explained by the number of bedrooms. This indicates that the number of bedrooms alone is not a powerful predictor of the listing price.

All the coefficients in this regression model are statistically significant at the 0.05 level. This conclusion is based on the p-values associated with the t-statistics for each coefficient (Intercept, C(new_bedroom_type)[T.2], C(new_bedroom_type)[T.3], and C(new_bedroom_type)[T.4+]), all of which are less than 0.00(less than 0.05). Therefore, we can reject the null hypothesis that these coefficients are zero, and conclude that the number of bedrooms is a significant predictor of the price in the model.

The coefficient for "C(new_bedroom_type)[T.2]" is 1481.0031, which means that, on average, the listing price is expected to be approximately 1,481 more units when the number of bedrooms is 2 compared to 1 bedroom.

The coefficient for "C(new_bedroom_type)[T.3]" is 3317.3668, indicating that the expected listing price is approximately 3,317 more units for 3 bedrooms compared to 1 bedroom.

The coefficient for "C(new_bedroom_type)[T.4+]" is 6592.1869, suggesting that the expected listing price is approximately 6,592 more units for 4 or more bedrooms compared to 1 bedroom.

2. Now repeat the process with the model where you analyze log price instead of price. Interpret the results. Which model behaves better in the sense of R2?

[39]:
```python
air_new['price_log'] = np.log(air_new["price"])
```

[40]:
```python
m_log = smf.ols('price_log ~ new_bedroom_type', data=air_new).fit()
m_log.summary()
```

[40]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:              price_log   R-squared:                       0.203
Model:                            OLS   Adj. R-squared:                  0.203
Method:                 Least Squares   F-statistic:                     1293.
Date:                Fri, 19 May 2023   Prob (F-statistic):               0.00
Time:                        06:11:32   Log-Likelihood:                -16912.
No. Observations:               15200   AIC:                         3.383e+04
Df Residuals:                   15196   BIC:                         3.386e+04
Df Model:                           3
Covariance Type:            nonrobust
================================================================================
==========
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
----------
Intercept               7.0150      0.007   1040.367      0.000       7.002
7.028
new_bedroom_type[T.2]   0.6781      0.017     40.014      0.000       0.645
```

```
0.711
new_bedroom_type[T.3]      1.1249      0.031     35.989      0.000       1.064
1.186
new_bedroom_type[T.4+]     1.3712      0.035     38.865      0.000       1.302
1.440
==============================================================================
Omnibus:                     3643.283   Durbin-Watson:                   1.643
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            16193.066
Skew:                           1.108   Prob(JB):                         0.00
Kurtosis:                       7.545   Cond. No.                         6.02
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

The R-squared value for this model is 0.203, which means that approximately 20.3% of the variation in the logarithm of the price can be explained by the number of bedrooms. Comparing this with the previous model, we can see that the model with log(price) has a higher R-squared value(compared with 0.036 in the m_air model above), indicating a better fit in terms of explaining the variation in the data.

All the coefficients in this regression model are statistically significant at the 0.05 level. This conclusion is based on the p-values associated with the t-statistics for each coefficient (Intercept, C(new_bedroom_type)[T.2], C(new_bedroom_type)[T.3], and C(new_bedroom_type)[T.4+]), all of which are less than 0.00(less than 0.05). Therefore, we can reject the null hypothesis that these coefficients are zero, and conclude that the number of bedrooms is a significant predictor of the log of price in the model.

The intercept term is 7.0150. This represents the estimated logarithm of the price when the number of bedrooms is 1(reference category). Therefore, the expected logarithm of the price for 1 bedroom is 7.0150.

The coefficient for "C(new_bedroom_type)[T.2]" is 0.6781. This means that, on average, the logarithm of the price increases by 0.6781 units when the number of bedrooms changes from 1 to 2.

The coefficient for "C(new_bedroom_type)[T.3]" is 1.1249. This implies that, on average, the logarithm of the price increases by 1.1249 units when the number of bedrooms changes from 1 to 3.

The coefficient for "C(new_bedroom_type)[T.4+]" is 1.3712. This suggests that, on average, the logarithm of the price increases by 1.3712 units when the number of bedrooms changes from 1 to 4 or more.

3. Finally we just add two more variables to the model: room type and accommodates. While room type only contains three values, the other two contain many different categories. Recode these as • accommodates: "1", "2", "3", "4 and more".Run this

model. Interpret and comment the more interesting/important results. Do not forget to mention what are the relevant reference categories and R2.

[41]:
```
air_new["new_accommodate"] = pd.cut(air_new.accommodates,
                            bins= [1, 2, 3, 4, np.inf],
                            labels = ["1", "2", "3", "4 and more"],
                            right=False
                            )
air_new.head()
```

[41]:
```
         room_type  accommodates  bedrooms  price new_bedroom_type  price_log  \
0  Entire home/apt             3       1.0   1845                1   7.520235
1     Private room             2       1.0   1275                1   7.150701
2     Private room             2       1.0    800                1   6.684612
3     Private room             2       1.0    800                1   6.684612
4     Private room             2       1.0   1845                1   7.520235

  new_accommodate
0               3
1               2
2               2
3               2
4               2
```

[42]:
```
m_new = smf.ols("price_log ~ new_bedroom_type + room_type + new_accommodate",␣
 ↪data=air_new).fit()
m_new.summary()
```

[42]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:              price_log   R-squared:                       0.242
Model:                            OLS   Adj. R-squared:                  0.242
Method:             Least Squares   F-statistic:                     539.9
Date:                Fri, 19 May 2023   Prob (F-statistic):               0.00
Time:                        06:12:58   Log-Likelihood:                -16531.
No. Observations:               15200   AIC:                         3.308e+04
Df Residuals:                   15190   BIC:                         3.316e+04
Df Model:                           9
Covariance Type:            nonrobust
================================================================================
=================
                                   coef    std err          t      P>|t|
[0.025      0.975]
--------------------------------------------------------------------------------
-----------------
Intercept                        6.9007      0.032    215.886      0.000
```

```
6.838       6.963
new_bedroom_type[T.2]              0.5087      0.021     24.406      0.000
0.468       0.550
new_bedroom_type[T.3]              0.9379      0.034     27.713      0.000
0.872       1.004
new_bedroom_type[T.4+]             1.2097      0.037     32.816      0.000
1.137       1.282
room_type[T.Hotel room]            0.1332      0.028      4.827      0.000
0.079       0.187
room_type[T.Private room]         -0.0583      0.013     -4.528      0.000
-0.084      -0.033
room_type[T.Shared room]          -0.6869      0.033    -20.786      0.000
-0.752      -0.622
new_accommodate[T.2]               0.1417      0.032      4.466      0.000
0.080       0.204
new_accommodate[T.3]               0.2048      0.035      5.904      0.000
0.137       0.273
new_accommodate[T.4 and more]      0.3136      0.033      9.455      0.000
0.249       0.379
==============================================================================
Omnibus:                     4377.153   Durbin-Watson:                   1.642
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            22672.783
Skew:                           1.295   Prob(JB):                         0.00
Kurtosis:                       8.394   Cond. No.                         13.9
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
"""
```

Most interesting conclusion: The number of bedrooms, room type, and accommodates all demonstrate significant effects on the logarithm of the price. Increasing the number of bedrooms is associated with higher prices, with larger increases observed for properties with more bedrooms. Additionally, certain room types, such as hotel rooms, tend to have higher prices, while private and shared rooms are generally priced lower. Moreover, properties capable of accommodating more guests command higher prices. Overall, these findings align with common expectations and highlight the importance of these variables in determining property prices.

Detailed interpretation: The R-squared value for this model is 0.242, indicating that approximately 24.2% of the variation in the logarithm of the price can be explained by the variables included in the model.

Intercept(reference category): The intercept value of 6.9007 captures the estimated baseline logarithm of the price when all the categorical variables are at their reference categories (1 bedroom, Entire home/apartment, and accommodates 1 person).

Coefficients for new_bedroom_type: The coefficient for "C(new_bedroom_type)[T.2]" is 0.5087. This suggests that, on average, the logarithm of the price increases by approximately 0.5087 units when the number of bedrooms changes from 1 to 2. The coefficient for "C(new_bedroom_type)[T.3]" is 0.9379. It indicates that, on average, the logarithm of the price increases by approximately 0.9379 units when the number of bedrooms changes from 1 to 3. The coefficient for "C(new_bedroom_type)[T.4+]" is 1.2097. This implies that, on average, the logarithm of the price increases by approximately 1.2097 units when the number of bedrooms changes from 1 to 4 or more.

Coefficients for room_type: The coefficient for "room_type[T.Hotel room]" is 0.1332. This indicates that, on average, the logarithm of the price increases by approximately 0.1332 units when the room type is Hotel room compared to the reference category. The coefficient for "room_type[T.Private room]" is -0.0583. It suggests that, on average, the logarithm of the price decreases by approximately 0.0583 units when the room type is Private room compared to the reference category. The coefficient for "room_type[T.Shared room]" is -0.6869. This implies that, on average, the logarithm of the price decreases by approximately 0.6869 units when the room type is Shared room compared to the reference category.

Coefficients for new_accommodate: The coefficient for "C(new_accommodate)[T.2]" is 0.1417. This means that, on average, the logarithm of the price increases by approximately 0.1417 units when the accommodates category changes from 1 to 2. The coefficient for "C(new_accommodate)[T.3]" is 0.2048. It indicates that, on average, the logarithm of the price increases by approximately 0.2048 units when the accommodates category changes from 1 to 3. The coefficient for "C(new_accommodate)[T.4 and more]" is 0.3136. This suggests that, on average, the logarithm of the price increases by approximately 0.3136 units when the accommodates category changes from 1 to 4 or more.

1.3.3 2.3 Predict

1. Now use the model above to predict (log) price for each listing in your data.

[43]:
```
air_new['predicted_log_price'] = m_new.predict(air_new)
air_new.head()
```

[43]:
```
         room_type  accommodates  bedrooms  price new_bedroom_type  price_log  \
0  Entire home/apt             3       1.0   1845                1   7.520235
1     Private room             2       1.0   1275                1   7.150701
2     Private room             2       1.0    800                1   6.684612
3     Private room             2       1.0    800                1   6.684612
4     Private room             2       1.0   1845                1   7.520235

  new_accommodate  predicted_log_price
0               3             7.105497
1               2             6.984033
2               2             6.984033
3               2             6.984033
4               2             6.984033
```

2. Compute root-mean-squared-error (RMSE) of this prediction.

[44]:
```
true_log_prices = air_new['price_log']
predicted_log_prices = air_new['predicted_log_price']
rmse = np.sqrt(mean_squared_error(true_log_prices, predicted_log_prices))
rmse
```

[44]:
```
0.7179466122053302
```

The RMSE of this prediction is 0.7179.

3.Now use your model to predict the price for a 2-bedroom apartment that accommodates 4.You can either leave out the variables that are not specified from your model, or choose reasonable values for those, and explain your reasoning.

[45]:
```
predict_data = pd.DataFrame({'room_type': ['Entire home/apt'],
                            'new_accommodate': ['4 and more'],
                            'new_bedroom_type': ['2']})
predicted_log_price = m_new.predict(predict_data)
predicted_log_price
```

[45]:
```
0    7.722903
dtype: float64
```

[46]:
```
predicted_price = np.exp(predicted_log_price)
print("Predicted Price is:", predicted_price)
```

```
Predicted Price is: 0    2259.509206
dtype: float64
```

To predict the price for a 2-bedroom apartment that accommodates 4, we can use the given regression model and choose reasonable values for the variables that are not specified. Since the variables new_bedroom_type and new_accommodate are not specified, we can make reasonable assumptions. For new_bedroom_type, we can assume it to be '2' representing 2 bedrooms. For new_accommodate, we can assume it to be '4 and more' indicating accommodation for 4 people or more.By plugging these values into the model, we can make a prediction. The model takes into account factors such as the number of bedrooms, the room type, and the accommodation capacity.The predicted log price for the 2-bedroom apartment that accommodates 4 is approximately 7.722903. After applying exponential function to get the actual predict price, the predicted price to be around 2259.51 units.

4.Compute the average log price for all listings in this group (2BR apartment that accommodates 4). Compare the result with your prediction. How close did you get?

[47]:
```
filtered_data = air_new[(air_new['bedrooms'] == 2) & (air_new['accommodates']␣
 ↪== 4)]
average_log_price = filtered_data['price_log'].mean()
average_log_price
```

[47]:
```
7.728450922007445
```

[48]:
```python
average_price = np.exp(average_log_price)
print("Average Price is:", average_price)
```

```
Average Price is: 2272.0798444050124
```

When comparing these values, we can see that the predicted log price and the actual average log price are very close (7.722903 vs 7.728451). After applying exponential function to get the actual average price, the average price to be around 2272.08 units which is close to the actual predicted price 2259.5 units.This indicates that my model appears to be quite accurate in its prediction for this specific scenario.

1.4 I spent 12 hours in this PS.