In [1]:
```python
#libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:
```python
#updating number of output results
pd.set_option('display.max_rows', 500)
```

Business Questions

1. Which regions of the United States have the hottest housing markets?
2. Is there a cutoff for the market hotness index where a significant difference between markets becomes apparent?
3. Does the allure of the certain cities (favorable place to live) play into the cities' assigned market hotness index?
4. Can we predict the market hotness index of cities on a monthly/yearly basis?

Reading Data

Realtor.com Historical Monthly Inventory Data per Metro Area

Monthly data updated on March 31, 2022 with data through March 2022. Next update scheduled for May 10, 2022 with data through April 2022.

Market trends and monthly statistics on active for-sale listings (including median list price, average list price, luxury list price, median days on market, average days on market, total active listings, new listings, price increases, price reductions). Attribution: cite any full or partial use of the data to the 'realtor.com residential listings database.'

In [3]:
```python
#core_metrics csv file
core_history = pd.read_csv("C:/Users/phill/OneDrive/Desktop/Capstone_DSC680/Data/RDC_Inventory_Core_Metrics_Met
```

```
C:\Users\phill\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3444: DtypeWarning: Columns (0,1,3
8,39) have mixed types.Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

In [4]:
```python
core_history.head()
```

Out[4]:

	month_date_yyyymm	cbsa_code	cbsa_title	HouseholdRank	median_listing_price	median_listing_price_mm	median_listing_price_yy	active_li
0	202203	35620	new york-newark-jersey city, ny-nj-pa	1.0	699000.0	0.0029	0.0779	
1	202203	31080	los angeles-long beach-anaheim, ca	2.0	949950.0	0.0231	-0.0496	
2	202203	16980	chicago-naperville-elgin, il-in-wi	3.0	337000.0	0.0307	-0.0369	
3	202203	19100	dallas-fort worth-arlington, tx	4.0	425000.0	0.0391	0.1759	
4	202203	26420	houston-the woodlands-sugar land, tx	5.0	373733.0	0.0310	0.0945	

5 rows × 41 columns

In [5]:
```python
core_history.month_date_yyyymm.unique()
```

Out[5]:
```
array([202203, 202202, 202201, 202112, 202111, 202110, 202109, 202108,
       202107, 202106, 202105, 202104, 202103, 202102, 202101, 202012,
       202011, 202010, 202009, 202008, 202007, 202006, 202005, 202004,
       202003, 202002, 202001, 201912, 201911, 201910, 201909, 201908,
       201907, 201906, 201905, 201904, 201903, 201902, 201901, 201812,
       201811, 201810, 201809, 201808, 201807, 201806, 201805, 201804,
       201803, 201802, 201801, 201712, 201711, 201710, '201710', '201709',
       '201708', '201707', '201706', '201705', '201704', '201703',
       '201702', '201701', '201612', '201611', '201610', '201609',
       '201608', '201607',
       'quality_flag = 1:  year-over-year figures may be impacted'],
      dtype=object)
```

For the variables which are suffixed with "mm" or "yy", they represent the percentage change in the certain data value from the previous month (mm) or previous year (yy). Therefore, they will allow for understanding trends in the values over time.

Realtor.com Market Hotness per Metro Area

Realtor.com Market Hotness Index: scores and rankings based on days on market (supply index) and realtor.com views per property (demand index). Attribution: cite any full or partial use of the data to the 'realtor.com market hotness index'.

In [6]:
```python
#hotness metrics csv file
hot_history = pd.read_csv("C:/Users/phill/OneDrive/Desktop/Capstone_DSC680/Data/RDC_Inventory_Hotness_Metrics_M
```

In [7]:
```python
hot_history.head()
```

Out[7]:

	month_date_yyyymm	cbsa_code	cbsa_title	nielsen_hh_rank	hotness_rank	hotness_rank_mm	hotness_rank_yy	hotness_score	supply_sco
0	202203	29820	las vegas-henderson-paradise, nv	33.0	173.0	-10.0	-4.0	45.652174	87.9598
1	202203	21060	elizabethtown-fort knox, ky	283.0	69.0	-3.0	-17.0	67.892977	77.5919
2	202203	47580	warner robins, ga	229.0	254.0	-27.0	-122.0	25.919732	27.0903
3	202203	25420	harrisburg-carlisle, pa	91.0	117.0	31.0	-4.0	56.020067	45.4849
4	202203	36420	oklahoma city, ok	41.0	246.0	18.0	-11.0	27.257525	41.1371

5 rows × 24 columns

In [8]:
```python
hot_history.month_date_yyyymm.unique()
```

Out[8]:
```
array(['202203', '202202', '202201', '202112', '202111', '202110',
       '202109', '202108', '202107', '202106', '202105', '202104',
       '202103', '202102', '202101', '202012', '202011', '202010',
       '202009', '202008', '202007', '202006', '202005', '202004',
       '202003', '202002', '202001', '201912', '201911', '201910',
       '201909', '201908', '201907', '201906', '201905', '201904',
       '201903', '201902', '201901', '201812', '201811', '201810',
       '201809', '201808', '201807', '201806', '201805', '201804',
       '201803', '201802', '201801', '201712', '201711', '201710',
       '201709', '201708', 'quality_flag = 1'], dtype=object)
```

Simplemaps: World Cities Database

Information on U.S. cities, includes:

- City Name
- State Name
- County Name
- Latitude
- Longitude
- Population
- Density
- Timezone

In [9]:
```python
#us cities information
cities_pd = pd.read_csv("C:/Users/phill/OneDrive/Desktop/Capstone_DSC680/Data/uscities.csv")
```

In [10]:
```python
cities_pd.head()
```

Out[10]:

	city	city_ascii	state_id	state_name	county_fips	county_name	lat	lng	population	density	source	military	incorporated
0	New York	New York	NY	New York	36061	New York	40.6943	-73.9249	18713220	10715	polygon	False	True
1	Los Angeles	Los Angeles	CA	California	6037	Los Angeles	34.1139	-118.4068	12750807	3276	polygon	False	True
2	Chicago	Chicago	IL	Illinois	17031	Cook	41.8373	-87.6862	8604203	4574	polygon	False	True
3	Miami	Miami	FL	Florida	12086	Miami-Dade	25.7839	-80.2102	6445545	5019	polygon	False	True
4	Dallas	Dallas	TX	Texas	48113	Dallas	32.7936	-96.7662	5743938	1526	polygon	False	True

Data Cleansing

Monthly Inventory Data

In [11]:
```python
print("Shape: ", core_history.shape)
```

```
Shape:  (63274, 41)
```

In [12]:
```python
core_history.dtypes
```

Out[12]:
```
month_date_yyyymm                              object
cbsa_code                                      object
cbsa_title                                     object
HouseholdRank                                 float64
median_listing_price                          float64
median_listing_price_mm                       float64
median_listing_price_yy                       float64
active_listing_count                          float64
active_listing_count_mm                       float64
active_listing_count_yy                       float64
median_days_on_market                         float64
median_days_on_market_mm                      float64
median_days_on_market_yy                      float64
new_listing_count                             float64
new_listing_count_mm                          float64
new_listing_count_yy                          float64
price_increased_count                         float64
price_increased_count_mm                      float64
price_increased_count_yy                      float64
price_reduced_count                           float64
price_reduced_count_mm                        float64
price_reduced_count_yy                        float64
pending_listing_count                         float64
pending_listing_count_mm                      float64
pending_listing_count_yy                      float64
median_listing_price_per_square_foot          float64
median_listing_price_per_square_foot_mm       float64
median_listing_price_per_square_foot_yy       float64
median_square_feet                            float64
median_square_feet_mm                         float64
median_square_feet_yy                         float64
average_listing_price                         float64
average_listing_price_mm                      float64
average_listing_price_yy                      float64
total_listing_count                           float64
total_listing_count_mm                        float64
total_listing_count_yy                        float64
pending_ratio                                 float64
pending_ratio_mm                               object
pending_ratio_yy                               object
quality_flag                                  float64
dtype: object
```

I am going to convert the variables 'pending_ratio_mm' and 'pending_ratio_yy' to numeric types, since they are currently defined as objects which does not represent the data.

In [13]:
```python
core_history[core_history['pending_ratio_mm']=='#NAME?']
```

Out[13]:

	month_date_yyyymm	cbsa_code	cbsa_title	HouseholdRank	median_listing_price	median_listing_price_mm	median_listing_price_yy	activ
12811	202102	25100	guymon, ok	891.0	350000.0	2.3333	1.9179	

1 rows × 41 columns

In [14]:
```python
core_history[core_history['pending_ratio_yy']=='#NAME?']
```

Out[14]:

	month_date_yyyymm	cbsa_code	cbsa_title	HouseholdRank	median_listing_price	median_listing_price_mm	median_listing_price_yy	active
2724	202201	25100	guymon, ok	891.0	375000.0	0.3636	2.5714	

1 rows × 41 columns

In [15]:
```python
#replace arbitrary value in pending_ratio_mm column with NaN
core_history['pending_ratio_mm'] = core_history['pending_ratio_mm'].replace({'#NAME?': np.nan})
```

In [16]:
```python
#replace arbitrary value in pending_ratio_yy column with NaN
core_history['pending_ratio_yy'] = core_history['pending_ratio_yy'].replace({'#NAME?': np.nan})
```

In [17]:
```python
#convert pending_ratio_mm to float64
core_history['pending_ratio_mm'] = core_history['pending_ratio_mm'].astype(float, errors = 'raise')
```

In [18]:
```python
#convert pending_ratio_yy to float64
core_history['pending_ratio_yy'] = core_history['pending_ratio_yy'].astype(float, errors = 'raise')
```

In [19]:
```python
core_history.describe()
```

Out[19]:

	HouseholdRank	median_listing_price	median_listing_price_mm	median_listing_price_yy	active_listing_count	active_listing_count_mm	acti
count	63273.000000	6.327300e+04	52269.000000	52269.000000	63273.000000	52268.000000	
mean	459.000000	2.511767e+05	0.007882	0.081275	1047.051381	-0.016822	
std	264.717033	1.715830e+05	0.067474	0.158750	3613.741968	0.237067	
min	1.000000	1.990000e+04	-0.692500	-0.843400	0.000000	-1.000000	
25%	230.000000	1.499000e+05	-0.018200	0.000000	122.000000	-0.075900	
50%	459.000000	2.099000e+05	0.000400	0.064100	271.000000	-0.019800	
75%	688.000000	2.946500e+05	0.028800	0.140600	707.000000	0.032400	
max	917.000000	3.000000e+06	2.985500	5.600000	85761.000000	48.000000	

8 rows × 38 columns

In [20]:
```python
#inventory data
# Rows containing duplicate data -- none!
duplicate_rows_core = core_history[core_history.duplicated()]
print("number of duplicate rows: ", duplicate_rows_core.shape)

# Finding the null values
print("\nNull values")
print(core_history.isnull().sum())
```

```
number of duplicate rows:  (0, 41)

Null values
month_date_yyyymm                              0
cbsa_code                                      0
cbsa_title                                     1
HouseholdRank                                  1
median_listing_price                           1
median_listing_price_mm                    11005
median_listing_price_yy                    11005
active_listing_count                           1
active_listing_count_mm                    11006
active_listing_count_yy                    11006
median_days_on_market                          1
median_days_on_market_mm                   11005
median_days_on_market_yy                   11005
new_listing_count                              1
new_listing_count_mm                       11297
new_listing_count_yy                       11343
price_increased_count                          1
price_increased_count_mm                   37735
price_increased_count_yy                   37435
price_reduced_count                            1
price_reduced_count_mm                     12080
price_reduced_count_yy                     11829
pending_listing_count                       2708
pending_listing_count_mm                   13398
pending_listing_count_yy                   14674
median_listing_price_per_square_foot          18
median_listing_price_per_square_foot_mm    11011
median_listing_price_per_square_foot_yy    11025
median_square_feet                            18
median_square_feet_mm                      11011
median_square_feet_yy                      11025
average_listing_price                          1
average_listing_price_mm                   11005
average_listing_price_yy                   11005
total_listing_count                            1
total_listing_count_mm                     11005
total_listing_count_yy                     11005
pending_ratio                               2708
pending_ratio_mm                           13127
pending_ratio_yy                           14194
quality_flag                               11005
dtype: int64
```

In [21]:
```python
#row 63273 seems to have more informational data than pertaining to the dataset
#i am going to remove it
#last row in dataframe --> tail 1 record
core_history = core_history.drop(core_history.tail(1).index)
```

In [22]:
```python
#looking at rows with missing data
core_history[core_history.isnull().any(axis=1)]
```

Out[22]:

	month_date_yyyymm	cbsa_code	cbsa_title	HouseholdRank	median_listing_price	median_listing_price_mm	median_listing_price_yy	acti
62	202203	35300	new haven-milford, ct	63.0	354500.0	0.0292	0.1936	
129	202203	37900	peoria, il	130.0	110950.0	0.1321	0.0577	
139	202203	11260	anchorage, ak	140.0	425000.0	0.0572	0.1863	
161	202203	46540	utica-rome, ny	162.0	183950.0	0.0856	0.0773	
162	202203	20260	duluth, mn-wi	163.0	249950.0	0.1013	0.0989	
...	...	...	...	...	...	...	...	
63268	201607	46900	vernon, tx	913.0	139000.0	NaN	NaN	
63269	201607	18780	craig, co	914.0	175000.0	NaN	NaN	
63270	201607	29500	lamesa, tx	915.0	101900.0	NaN	NaN	
63271	201607	49820	zapata, tx	916.0	128000.0	NaN	NaN	
63272	201607	37780	pecos, tx	917.0	169900.0	NaN	NaN	

42673 rows × 41 columns

Many of the missing values in this dataset pertain to the variables suffixed with "mm" or "yy", which as explained above indicate the percentage change in the variable from the previous month or year respectively.

If I remove the entire row that has missing values, then we will lose all of the data for the specific metro area which I am not particularly interested in doing.

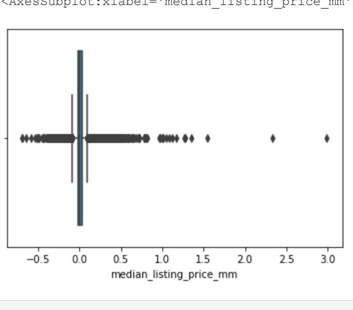
Therefore, I am leaning towards defining a default value for the missing data and then in visualizations they can be subsetted out to look at trends for other cities.

Boxplot and Histogram of median_listing_price_mm to assess how to fill missing values

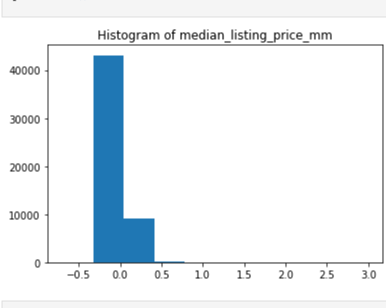
In [23]:
```python
#trying to decide on missing values to fill in
sns.boxplot(core_history.median_listing_price_mm)

#many outliers
#main distribution around 0
```

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[23]:
```
<AxesSubplot:xlabel='median_listing_price_mm'>
```



In [24]:
```python
plt.hist(core_history['median_listing_price_mm'])
plt.title("Histogram of median_listing_price_mm")
plt.show()
```



In [25]:
```python
core_history['median_listing_price_yy'].mode()
```

Out[25]:
```
0    0.0
dtype: float64
```

In [26]:
```python
core_history['median_listing_price_yy'].median()
```

Out[26]:
```
0.0641
```

I am going to come back to handling the missing values, since I am conflicted on whether it will disrupt the solution of my business questions. I also think it will be better to handle missing data after I merge all of the datasets, so we can have a better picture.

I do not want to naively fill in data that will lead to a misunderstanding of what each metro area brings to the issue.

Market Hotness

In [27]:
```python
print("Shape: ", hot_history.shape)
```

```
Shape:  (16801, 24)
```

In [28]:
```python
hot_history.dtypes
```

Out[28]:
```
month_date_yyyymm                         object
cbsa_code                                 object
cbsa_title                                object
nielsen_hh_rank                          float64
hotness_rank                             float64
hotness_rank_mm                          float64
hotness_rank_yy                          float64
hotness_score                            float64
supply_score                             float64
demand_score                             float64
median_days_on_market                    float64
median_days_on_market_mm                 float64
median_dom_mm_day                        float64
median_days_on_market_yy                 float64
median_dom_yy_day                        float64
median_dom_vs_us                         float64
ldp_unique_viewers_per_property_mm       float64
ldp_unique_viewers_per_property_yy       float64
ldp_unique_viewers_per_property_vs_us    float64
median_listing_price                     float64
median_listing_price_mm                  float64
median_listing_price_yy                  float64
median_listing_price_vs_us               float64
quality_flag                             float64
dtype: object
```

In [29]:
```python
#market hotness data
# Rows containing duplicate data -- none!
duplicate_rows_hot = hot_history[hot_history.duplicated()]
print("number of duplicate rows: ", duplicate_rows_hot.shape)

# Finding the null values
print("\nNull values")
print(hot_history.isnull().sum())
```

```
number of duplicate rows:  (0, 24)

Null values
month_date_yyyymm                           0
cbsa_code                                   0
cbsa_title                                  1
nielsen_hh_rank                             1
hotness_rank                                1
hotness_rank_mm                          3601
hotness_rank_yy                          3601
hotness_score                               1
supply_score                                1
demand_score                                1
median_days_on_market                       1
median_days_on_market_mm                 3601
median_dom_mm_day                        3601
median_days_on_market_yy                 3601
median_dom_yy_day                        3601
median_dom_vs_us                            1
ldp_unique_viewers_per_property_mm       3601
ldp_unique_viewers_per_property_yy       3601
ldp_unique_viewers_per_property_vs_us       1
median_listing_price                        1
median_listing_price_mm                  3601
median_listing_price_yy                  3601
median_listing_price_vs_us                  1
quality_flag                                1
dtype: int64
```

In [30]:
```python
#looking at rows with missing data
hot_history[hot_history.isnull().any(axis=1)]
```

Out[30]:

	month_date_yyyymm	cbsa_code	cbsa_title	nielsen_hh_rank	hotness_rank	hotness_rank_mm	hotness_rank_yy	hotness_score	suppl
13200	201807	49660	youngstown-warren-boardman, oh-pa	94.0	166.0	NaN	NaN	45.652174	32
13201	201807	21060	elizabethtown-fort knox, ky	283.0	107.0	NaN	NaN	61.705686	71
13202	201807	43900	spartanburg, sc	159.0	165.0	NaN	NaN	45.986622	39
13203	201807	25060	gulfport-biloxi-pascagoula, ms	133.0	269.0	NaN	NaN	16.889632	10
13204	201807	35300	new haven-milford, ct	63.0	158.0	NaN	NaN	46.822742	45
...	...	...	...	...	...	...	...	...	
16796	201708	20260	duluth, mn-wi	163.0	197.0	NaN	NaN	37.625418	15
16797	201708	41180	st. louis, mo-il	19.0	136.0	NaN	NaN	51.505017	56
16798	201708	17900	columbia, sc	67.0	249.0	NaN	NaN	24.414716	41
16799	201708	45900	traverse city, mi	261.0	223.0	NaN	NaN	32.107023	25
16800	quality_flag = 1	Data falls outside of expected range. Please ...	NaN	NaN	NaN	NaN	NaN	NaN	

3601 rows × 24 columns

In [31]:
```python
#row 16800 seems to have more informational data than pertaining to the dataset
#i am going to remove it
#last row in dataframe --> tail 1 record
hot_history = hot_history.drop(hot_history.tail(1).index)
```

Cities Database

In [32]:
```python
print(cities_pd.shape)
```

```
(28338, 17)
```

In [33]:
```python
cities_pd.dtypes
```

Out[33]:
```
city             object
city_ascii       object
state_id         object
state_name       object
county_fips       int64
county_name      object
lat             float64
lng             float64
population        int64
density           int64
source           object
military           bool
incorporated       bool
timezone         object
ranking           int64
zips             object
id                int64
dtype: object
```

In [34]:
```python
#looking at unique values of some categorical variables
cities_pd.source.unique()
```

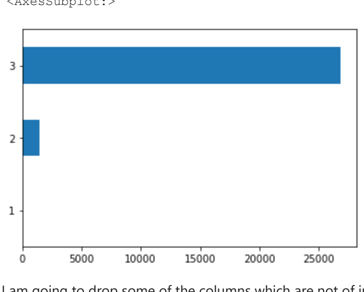
Out[34]:
```
array(['polygon'], dtype=object)
```

In [35]:
```python
cities_pd.incorporated.unique()
```

Out[35]:
```
array([ True, False])
```

In [36]:
```python
#bar chart of city rankings
#captures the importance of a city --> from 1-5
cities_pd.ranking.value_counts().sort_values().plot(kind = 'barh')
```

Out[36]:
```
<AxesSubplot:>
```



I am going to drop some of the columns which are not of interest to me for the business problem/questions:

- city_ascii (duplicate information)
- county fips
- source (only has one value)
- timezone (not relevant)
- zips
- id

In [37]:
```python
#dropping above columns
cities_pd = cities_pd.drop(['city_ascii','county_fips','source','timezone','zips','id'],1)
cities_pd
```

```
C:\Users\phill\AppData\Local\Temp/ipykernel_25828/4130845510.py:2: FutureWarning: In a future version of pandas
all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  cities_pd = cities_pd.drop(['city_ascii','county_fips','source','timezone','zips','id'],1)
```

Out[37]:

	city	state_id	state_name	county_name	lat	lng	population	density	military	incorporated	ranking
0	New York	NY	New York	New York	40.6943	-73.9249	18713220	10715	False	True	1
1	Los Angeles	CA	California	Los Angeles	34.1139	-118.4068	12750807	3276	False	True	1
2	Chicago	IL	Illinois	Cook	41.8373	-87.6862	8604203	4574	False	True	1
3	Miami	FL	Florida	Miami-Dade	25.7839	-80.2102	6445545	5019	False	True	1
4	Dallas	TX	Texas	Dallas	32.7936	-96.7662	5743938	1526	False	True	1
...	...	...	...	...	...	...	...	...	...	...	...
28333	Gross	NE	Nebraska	Boyd	42.9461	-98.5697	2	6	False	True	3
28334	Lotsee	OK	Oklahoma	Tulsa	36.1334	-96.2091	2	39	False	True	3
28335	The Ranch	MN	Minnesota	Mahnomen	47.3198	-95.6952	2	2	False	True	3
28336	Shamrock	OK	Oklahoma	Creek	35.9113	-96.5772	2	2	False	True	3
28337	Monowi	NE	Nebraska	Boyd	42.8307	-98.3296	1	1	False	True	3

28338 rows × 11 columns

In [38]:
```python
print('Shape of Cities Dataset after Column Removal')
print(cities_pd.shape)
```

```
Shape of Cities Dataset after Column Removal
(28338, 11)
```

In [39]:
```python
#city data
# Rows containing duplicate data -- none!
duplicate_rows_city = cities_pd[cities_pd.duplicated()]
print("number of duplicate rows: ", duplicate_rows_city.shape)

# Finding the null values -- none!
print("\nNull values")
print(cities_pd.isnull().sum())
```

```
number of duplicate rows:  (0, 11)

Null values
city            0
state_id        0
state_name      0
county_name     0
lat             0
lng             0
population      0
density         0
military        0
incorporated    0
ranking         0
dtype: int64
```

The cities database dataset is very clean in terms of duplicates and missing data, of which it has zero!

Merging Realtor.com datasets

The inventory dataset and the market hotness dataset are both from Realtor.com. They have a few categorical columns in common, and for the purpose of joining, I am going to use the cbsa_code column since it should be the same between the two datasets.

I was originally going to use cbsa_title but it is more prone to give incorrect results, since strings can contain different spacing and punctuations.

I am going to do an inner join between the DataFrames, so we can capture information for the metro areas that have information from both datasets, instead of having null values with an outer join which will not be useful. This may subset the dataset but it will be more advantageous to the problem at hand.

I ended up joining the datasets in Tableau and then exporting that dataset to CSV to read into Python, since the merge using Python functions was not working as expected.

In [40]:
```python
#merged in external source --> Tableau
#python merge was giving incorrect results
#reading in merged dataset
merged_realtor = pd.read_csv("C:/Users/phill/OneDrive/Desktop/Capstone_DSC680/Data/joined_realtor_data.csv")
```

In [41]:
```python
merged_realtor.shape
```

Out[41]:
```
(16800, 60)
```

In [42]:
```python
merged_realtor.head()
```

Out[42]:

	Month Date Yyyymm	Cbsa Code	Cbsa Title	Median Dom Mm Day	Median Dom Yy Day	Active Listing Count	Active Listing Count Mm	Active Listing Count Yy	Average Listing Price	Average Listing Price Mm	...	Price Increased Count Yy	Price Reduced Count	Price Reduced Count Mm	Price Reduced Count Yy	Qu
0	202203	35620	new york-newark-jersey city, ny-nj-pa	-17.0	-12.5	42816	0.0564	-0.0807	1700187	0.0001	...	0.3333	5968	0.1477	-0.0744	
1	202203	31080	los angeles-long beach-anaheim, ca	-0.5	-5.0	7081	0.0703	-0.2700	2411312	-0.0253	...	0.0580	1176	0.1951	-0.3438	
2	202203	16980	chicago-naperville-elgin, il-in-wi	-6.0	-8.0	11706	0.0225	-0.2399	528104	0.0197	...	-0.1183	2644	0.1338	-0.2205	
3	202203	19100	dallas-fort worth-arlington, tx	-6.0	-12.5	4524	0.0891	-0.2105	660783	0.0312	...	-0.4571	1056	0.3679	-0.1456	
4	202203	26420	houston-the woodlands-sugar land, tx	-3.5	-12.0	10603	-0.0384	-0.1722	542520	0.0218	...	-0.1067	2728	0.0912	-0.0870	

5 rows × 60 columns

In [43]:
```python
merged_realtor['Month Date Yyyymm'].unique()
```

Out[43]:
```
array([202203, 202202, 202201, 202112, 202111, 202110, 202109, 202108,
       202107, 202106, 202105, 202104, 202103, 202102, 202101, 202012,
       202011, 202010, 202009, 202008, 202007, 202006, 202005, 202004,
       202003, 202002, 202001, 201912, 201911, 201910, 201909, 201908,
       201907, 201906, 201905, 201904, 201903, 201902, 201901, 201812,
       201811, 201810, 201809, 201808, 201807, 201806, 201805, 201804,
       201803, 201802, 201801, 201712, 201711, 201710, 201709, 201708],
      dtype=int64)
```

In [44]:
```python
#merged data
# Rows containing duplicate data -- none!
duplicate_rows_merge = merged_realtor[merged_realtor.duplicated()]
print("number of duplicate rows: ", duplicate_rows_merge.shape)

# Finding the null values -- none!
print("\nNull values")
print(merged_realtor.isnull().sum())
```

```
number of duplicate rows:  (0, 60)

Null values
Month Date Yyyymm                                  0
Cbsa Code                                          0
Cbsa Title                                         0
Median Dom Mm Day                               3600
Median Dom Yy Day                               3600
Active Listing Count                               0
Active Listing Count Mm                            0
Active Listing Count Yy                            0
Average Listing Price                              0
Average Listing Price Mm                           0
Average Listing Price Yy                           0
Demand Score                                       0
Hotness Rank                                       0
Hotness Rank Mm                                 3600
Hotness Rank Yy                                 3600
Hotness Score                                      0
Household Rank                                     0
Ldp Unique Viewers Per Property Mm              3600
Ldp Unique Viewers Per Property Vs Us              0
Ldp Unique Viewers Per Property Yy              3600
Median Days On Market                              0
Median Days On Market Mm                           0
median days on market mm_hotness                3600
Median Days On Market Yy                           0
median days on market yy_hotness                3600
Median Dom Vs Us                                   0
Median Listing Price                               0
Median Listing Price Mm                            0
median listing price mm_hotness                 3600
Median Listing Price Per Square Foot               0
Median Listing Price Per Square Foot Mm            0
Median Listing Price Per Square Foot Yy            0
Median Listing Price Vs Us                         0
Median Listing Price Yy                            0
median listing price yy_hotness                 3600
Median Square Feet                                 0
Median Square Feet Mm                              0
Median Square Feet Yy                              0
New Listing Count                                  0
New Listing Count Mm                               0
New Listing Count Yy                               0
Nielsen Hh Rank                                    0
Pending Listing Count                            164
Pending Listing Count Mm                         225
Pending Listing Count Yy                         356
Pending Ratio                                    164
Pending Ratio Mm                                 210
Pending Ratio Yy                                 314
Price Increased Count                              0
Price Increased Count Mm                        1553
Price Increased Count Yy                        1479
Price Reduced Count                                0
Price Reduced Count Mm                             0
Price Reduced Count Yy                             0
Quality Flag                                       0
quality flag_hotness                               0
Supply Score                                       0
Total Listing Count                                0
Total Listing Count Mm                             0
Total Listing Count Yy                             0
dtype: int64
```

In [45]:

```
#looking at rows with missing data
merged_realtor[merged_realtor.isnull().any(axis=1)]
```

Out[45]:

	Month Date Yyyymm	Cbsa Code	Cbsa Title	Median Dom Mm Day	Median Dom Yy Day	Active Listing Count	Active Listing Count Mm	Active Listing Count Yy	Average Listing Price	Average Listing Price Mm	...	Price Increased Count Yy	Price Reduced Count	Price Reduced Count Mm	Price Reduced Count Yy
62	202203	35300	new haven-milford, ct	-20.5	-9.5	616	-0.0191	-0.6658	514659	0.0036	...	-0.50	132	0.1786	-0.0294
129	202203	37900	peoria, il	-7.5	-9.0	422	-0.0766	-0.4562	208356	0.1836	...	-1.00	144	0.0000	-0.4286
139	202203	11260	anchorage, ak	-24.5	-20.5	346	0.0176	-0.4627	478697	0.0645	...	0.00	64	0.2308	-0.4839
161	202203	46540	utica-rome, ny	-4.5	-28.0	270	-0.0493	-0.0816	267986	0.0762	...	NaN	48	0.0000	-0.1429
162	202203	20260	duluth, mn-wi	-25.5	-26.0	287	-0.0304	-0.3051	385764	0.0570	...	NaN	60	0.8750	0.2500
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
16795	201708	13900	bismarck, nd	NaN	NaN	687	0.0346	0.0802	318638	0.0055	...	NaN	224	0.0980	0.0769
16796	201708	44220	springfield, oh	NaN	NaN	642	0.2840	-0.0287	152726	-0.0080	...	0.40	248	0.1698	0.0508
16797	201708	12980	battle creek, mi	NaN	NaN	555	0.0221	0.5417	173958	0.0137	...	NaN	200	0.1364	1.3810
16798	201708	30140	lebanon, pa	NaN	NaN	856	-0.0081	-0.2337	236837	0.0028	...	0.25	192	0.3714	-0.0400
16799	201708	46180	tupelo, ms	NaN	NaN	532	0.0019	0.0114	213864	-0.0260	...	NaN	188	0.0930	1.3500

5679 rows × 60 columns

Metropolitan Wikipedia

In [46]:

```
#libraries
import requests # library to handle requests
from bs4 import BeautifulSoup # library to parse HTML documents
```

In [47]:

```
# get the response in the form of html
wikiurl="https://en.wikipedia.org/wiki/Metropolitan_statistical_area"
table_class="wikitable sortable jquery-tablesorter"
response=requests.get(wikiurl)
print(response.status_code)
```

```
200
```

In [48]:

```
# parse data from the html into a beautifulsoup object
soup = BeautifulSoup(response.text, 'html.parser')
metrotable=soup.find('table',{'class':"wikitable"})
```

In [49]:

```
df=pd.read_html(str(metrotable))
# convert list to dataframe
df=pd.DataFrame(df[0])
print(df.head())
print(df.tail())
```

```
   Rank              Metropolitan statistical area  2021 estimate  \
0     1  New York-Newark-Jersey City, NY-NJ-PA MSA       19768458
1     2     Los Angeles-Long Beach-Anaheim, CA MSA       12997353
2     3      Chicago-Naperville-Elgin, IL-IN-WI MSA        9509934
3     4         Dallas-Fort Worth-Arlington, TX MSA        7759615
4     5    Houston-The Woodlands-Sugar Land, TX MSA        7206841

   2020 Census % change Encompassing combined statistical area
0     20140470   -1.85%       New York-Newark, NY-NJ-CT-PA CSA
1     13200998   -1.54%         Los Angeles-Long Beach, CA CSA
2      9618502   -1.13%       Chicago-Naperville, IL-IN-WI CSA
3      7637387   +1.60%           Dallas-Fort Worth, TX-OK CSA
4      7122240   +1.19%          Houston-The Woodlands, TX CSA
     Rank Metropolitan statistical area  2021 estimate  2020 Census % change  \
379   380              Danville, IL MSA          73095        74188   -1.47%
380   381           Lewiston, ID-WA MSA          64851        64375   +0.74%
381   382           Walla Walla, WA MSA          62682        62584   +0.16%
382   383                  Enid, OK MSA          61926        62846   -1.46%
383   384           Carson City, NV MSA          58993        58639   +0.60%

     Encompassing combined statistical area
379                                     NaN
380                                     NaN
381  Kennewick-Richland-Walla Walla, WA CSA
382                                     NaN
383        Reno-Carson City-Fernley, NV CSA
```

There seem to be control characters in the columns with spaces, so I am going to work to remove them so they are easier to work with and access.

In [50]:

```
df.columns
```

Out[50]:

```
Index(['Rank', 'Metropolitan statistical area', '2021 estimate', '2020 Census',
       '% change', 'Encompassing combined statistical area'],
      dtype='object')
```

In [51]:

```
#updating column names with underscores
df.columns = [c.replace(' ', '_') for c in df.columns]
```

In [52]:

```
df.columns = df.columns.str.strip()
```

In [53]:

```
df.columns[5]
```

Out[53]:

```
'Encompassing_combined_statistical\xa0area'
```

In [54]:

```
#lower case text in column metropolitan area
df['Metropolitan\xa0statistical\xa0area'] = df['Metropolitan\xa0statistical\xa0area'].str.lower()
```

In [55]:

```
#replacing metropolitan area name 'MSA' with ''
df['Metropolitan\xa0statistical\xa0area'] = df['Metropolitan\xa0statistical\xa0area'].str.replace(r'msa', '')
```

In [56]:

```
#lower case text in column statistical area
df['Encompassing_combined_statistical\xa0area'] = df['Encompassing_combined_statistical\xa0area'].str.lower()
```

In [57]:

```
#replacing statistical area name 'CSA' with ''
df['Encompassing_combined_statistical\xa0area'] = df['Encompassing_combined_statistical\xa0area'].str.replace(r
```

In [58]:

```
#renaming column names with control characters
df = df.rename(columns={'Metropolitan\xa0statistical\xa0area': 'metro_area', 'Encompassing_combined_statistical
```

In [59]:

```
df.head()
```

Out[59]:

	Rank	metro_area	2021_estimate	2020_Census	%_change	combined_statistical_area
0	1	new york-newark-jersey city, ny-nj-pa	19768458	20140470	−1.85%	new york-newark, ny-nj-ct-pa
1	2	los angeles-long beach-anaheim, ca	12997353	13200998	−1.54%	los angeles-long beach, ca
2	3	chicago-naperville-elgin, il-in-wi	9509934	9618502	−1.13%	chicago-naperville, il-in-wi
3	4	dallas-fort worth-arlington, tx	7759615	7637387	+1.60%	dallas-fort worth, tx-ok
4	5	houston-the woodlands-sugar land, tx	7206841	7122240	+1.19%	houston-the woodlands, tx

In [60]:

```
df.dtypes
```

Out[60]:

```
Rank                          int64
metro_area                   object
2021_estimate                 int64
2020_Census                   int64
%_change                     object
combined_statistical_area    object
dtype: object
```

In [61]:

```
df.describe()
```

Out[61]:

	Rank	2021_estimate	2020_Census
count	384.000000	3.840000e+02	3.840000e+02
mean	192.500000	7.460229e+05	7.450639e+05
std	110.995495	1.655492e+06	1.670046e+06
min	1.000000	5.899300e+04	5.863900e+04
25%	96.750000	1.476232e+05	1.474762e+05
50%	192.500000	2.487975e+05	2.484905e+05
75%	288.250000	6.012120e+05	5.956365e+05
max	384.000000	1.976846e+07	2.014047e+07

The variable '%_change' is currently of object type, but I'd like it to be converted to a float since it is a numerical measure.

Therefore, I am going to work on correcting that.

In [62]:

```
print(df['%_change'][0][1:-1])
```

```
1.85
```

In [63]:

```
#remove first and last characters from the value
#first char is + or - currently
#last char is %
df['%_change'] = df['%_change'].str[1:-1]
```

In [64]:

```
df['%_change'].head(20)
```

Out[64]:

```
0     1.85
1     1.54
2     1.13
3     1.60
4     1.19
5     0.45
6     0.26
7     0.89
8     0.76
9     2.07
10    0.84
11    1.16
12    2.65
13    0.61
14    0.18
15    0.01
16    0.38
17    1.39
18    0.30
19    0.22
Name: %_change, dtype: object
```

In [65]:

```
#cast object type to float type
#convert df['%_change'] to float64
df['%_change'] = df['%_change'].astype(float, errors = 'raise')
```

In [66]:

```
#checking column types after update
df.dtypes
```

Out[66]:

```
Rank                          int64
metro_area                   object
2021_estimate                 int64
2020_Census                   int64
%_change                    float64
combined_statistical_area    object
dtype: object
```

Merging Joined Realor Dataset and Metro Population Dataset

The Metro Population dataset contains population information per metro area for the year of 2020 and then also provides an estimate for 2021.

The merged realtor dataset has information for dates ranging from 2017-2022, so in order to join the two datasets, I am going to subset the realtor one to only focus on the dates in the years of 2020 and 2021 in order to make the metro dataset relevant.

In [67]:

```
#convert int month date column to date
merged_realtor['Month Date Yyyymm'] = pd.to_datetime(merged_realtor['Month Date Yyyymm'].astype(str), format='%
```

In [68]:

```
#creating new columns for month and year in merged dataset
merged_realtor['year'] = pd.DatetimeIndex(merged_realtor['Month Date Yyyymm']).year
```

In [69]:

```
merged_realtor['year'].unique()
```

Out[69]:

```
array([2022, 2021, 2020, 2019, 2018, 2017], dtype=int64)
```

In [70]:

```
#month column
merged_realtor['month'] = pd.DatetimeIndex(merged_realtor['Month Date Yyyymm']).month
```

In [71]:

```
merged_realtor['month'].unique()
```

Out[71]:

```
array([ 3,  2,  1, 12, 11, 10,  9,  8,  7,  6,  5,  4], dtype=int64)
```

In [72]:

```
merged_realtor.head()
```

Out[72]:

	Month Date Yyyymm	Cbsa Code	Cbsa Title	Median Dom Mm Day	Median Dom Yy Day	Active Listing Count	Active Listing Count Mm	Active Listing Count Yy	Average Listing Price	Average Listing Price Mm	...	Price Reduced Count Mm	Price Reduced Count Yy	Quality Flag	quality flag_hotness	
0	2022-03-01	35620	new york-newark-jersey city, ny-nj-pa	-17.0	-12.5	42816	0.0564	-0.0807	1700187	0.0001	...	0.1477	-0.0744	0	0	2
1	2022-03-01	31080	los angeles-long beach-anaheim, ca	-0.5	-5.0	7081	0.0703	-0.2700	2411312	-0.0253	...	0.1951	-0.3438	0	0	7
2	2022-03-01	16980	chicago-naperville-elgin, il-in-wi	-6.0	-8.0	11706	0.0225	-0.2399	528104	0.0197	...	0.1338	-0.2205	0	0	5
3	2022-03-01	19100	dallas-fort worth-arlington, tx	-6.0	-12.5	4524	0.0891	-0.2105	660783	0.0312	...	0.3679	-0.1456	0	0	7
4	2022-03-01	26420	houston-the woodlands-sugar land, tx	-3.5	-12.0	10603	-0.0384	-0.1722	542520	0.0218	...	0.0912	-0.0870	0	0	3

5 rows × 62 columns

In [73]:

```
#subsetting merged dataset for years of 2021 and 2020
realtor_years = merged_realtor[(merged_realtor.year == 2020) | (merged_realtor.year == 2021)]
```

In [74]:

```
realtor_years.shape

#less than half of the original, not too bad!
```

Out[74]:

```
(7200, 62)
```

In [75]:

```
realtor_years['year'].unique()
```

Out[75]:

```
array([2021, 2020], dtype=int64)
```

In [76]:

```
#I am going to drop the month date column now since we have individual 'year' and 'month
realtor_years = realtor_years.drop(['Month Date Yyyymm'],1)
```

```
C:\Users\phill\AppData\Local\Temp/ipykernel_25828/4283728241.py:2: FutureWarning: In a future version of pandas
all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  realtor_years = realtor_years.drop(['Month Date Yyyymm'],1)
```

In [77]:

```
#cleansing strings for first dataset
realtor_years['Cbsa Title'] = realtor_years['Cbsa Title'].apply(lambda x: x.lower().strip())
```

In [78]:

```
#cleansing strings for second dataset
df['metro_area'] = df['metro_area'].apply(lambda x: x.lower().strip())
```

In [79]:

```
#merge datasets on specified key columns
merged_metro = pd.merge(realtor_years,df,left_on='Cbsa Title',right_on='metro_area')
```

In [80]:

```
merged_metro.head()
```

Out[80]:

	Cbsa Code	Cbsa Title	Median Dom Mm Day	Median Dom Yy Day	Active Listing Count	Active Listing Count Mm	Active Listing Count Yy	Average Listing Price	Average Listing Price Mm	Average Listing Price Yy	...	Total Listing Count Mm	Total Listing Count Yy	year	month	Rank	metro_ar
0	35620	new york-newark-jersey city, ny-nj-pa	11.0	-7.0	46763	-0.1382	-0.1824	1631077	0.0473	0.2380	...	-0.0940	-0.1408	2021	12	1	new yor newar jersey cit ny-nj-p
1	35620	new york-newark-jersey city, ny-nj-pa	4.0	-6.0	54263	-0.0524	-0.1738	1557436	0.0819	0.2129	...	-0.0390	-0.1636	2021	11	1	new yor newar jersey cit ny-nj-p
2	35620	new york-newark-jersey city, ny-nj-pa	4.0	4.0	57262	0.0060	-0.1657	1439479	0.0207	0.1067	...	-0.0105	-0.1574	2021	10	1	new yor newar jersey cit ny-nj-p
3	35620	new york-newark-jersey city, ny-nj-pa	0.5	-1.0	56921	-0.0022	-0.1521	1410300	0.0249	0.1403	...	-0.0026	-0.1175	2021	9	1	new yor newar jersey cit ny-nj-p
4	35620	new york-newark-jersey city, ny-nj-pa	-1.0	3.5	57049	0.0366	-0.1444	1376032	0.0776	0.1214	...	0.0406	-0.0924	2021	8	1	new yor newar jersey cit ny-nj-p

5 rows × 67 columns

In [81]:

```
merged_metro.shape
```

Out[81]:

```
(6192, 67)
```

In [82]:

```
merged_metro.columns
```

Out[82]:

```
Index(['Cbsa Code', 'Cbsa Title', 'Median Dom Mm Day', 'Median Dom Yy Day',
       'Active Listing Count', 'Active Listing Count Mm',
       'Active Listing Count Yy', 'Average Listing Price',
       'Average Listing Price Mm', 'Average Listing Price Yy', 'Demand Score',
       'Hotness Rank', 'Hotness Rank Mm', 'Hotness Rank Yy', 'Hotness Score',
       'Household Rank', 'Ldp Unique Viewers Per Property Mm',
       'Ldp Unique Viewers Per Property Vs Us',
       'Ldp Unique Viewers Per Property Yy', 'Median Days On Market',
       'Median Days On Market Mm', 'median days on market mm_hotness',
       'Median Days On Market Yy', 'median days on market yy_hotness',
       'Median Dom Vs Us', 'Median Listing Price', 'Median Listing Price Mm',
       'median listing price mm_hotness',
       'Median Listing Price Per Square Foot',
       'Median Listing Price Per Square Foot Mm',
       'Median Listing Price Per Square Foot Yy', 'Median Listing Price Vs Us',
       'Median Listing Price Yy', 'median listing price yy_hotness',
       'Median Square Feet', 'Median Square Feet Mm', 'Median Square Feet Yy',
       'New Listing Count', 'New Listing Count Mm', 'New Listing Count Yy',
       'Nielsen Hh Rank', 'Pending Listing Count', 'Pending Listing Count Mm',
       'Pending Listing Count Yy', 'Pending Ratio', 'Pending Ratio Mm',
       'Pending Ratio Yy', 'Price Increased Count', 'Price Increased Count Mm',
       'Price Increased Count Yy', 'Price Reduced Count',
       'Price Reduced Count Mm', 'Price Reduced Count Yy', 'Quality Flag',
       'quality flag_hotness', 'Supply Score', 'Total Listing Count',
       'Total Listing Count Mm', 'Total Listing Count Yy', 'year', 'month',
       'Rank', 'metro_area', '2021_estimate', '2020_Census', '%_change',
       'combined_statistical_area'],
      dtype='object')
```

In [83]:

```
#creating correlation plot for all columns
corr = merged_metro.corr()
corr.style.background_gradient(cmap='coolwarm')
```

Out[83]:

	Cbsa Code	Median Dom Mm Day	Median Dom Yy Day	Active Listing Count	Active Listing Count Mm	Active Listing Count Yy	Average Listing Price	Average Listing Price Mm	Average Listing Price Yy	Demand Score	Hotness Rank	Hotness Rank Mm
Cbsa Code	1.000000	0.007877	0.054846	0.021216	0.005642	0.046950	0.081449	-0.012544	-0.034003	0.095928	-0.107527	-0.010092
Median Dom Mm Day	0.007877	1.000000	0.231246	0.006812	-0.136367	0.150442	-0.001151	-0.065753	-0.065573	-0.035854	0.088315	-0.508700
Median Dom Yy Day	0.054846	0.231246	1.000000	0.097786	-0.208965	0.426696	-0.063438	0.050753	-0.279756	-0.088590	0.100125	-0.155278
Active Listing Count	0.021216	0.006812	0.097786	1.000000	0.012836	0.100460	0.182266	-0.003431	-0.043520	-0.268661	0.170073	-0.018193
Active Listing Count Mm	0.005642	-0.136367	-0.208965	0.012836	1.000000	0.199868	-0.004483	-0.265055	-0.086072	-0.045502	-0.064947	-0.020810
Active Listing Count Yy	0.046950	0.150442	0.426696	0.100460	0.199868	1.000000	-0.057394	-0.087863	-0.447008	-0.232489	0.141616	-0.091095
Average Listing Price	0.081449	-0.001151	-0.063438	0.182266	-0.004483	-0.057394	1.000000	0.066678	0.268598	0.017333	-0.069405	0.020239
Average Listing Price Mm	-0.012544	-0.065753	0.050753	-0.003431	-0.265055	-0.087863	0.066678	1.000000	0.303435	0.053011	-0.013252	0.033772
Average Listing Price Yy	-0.034003	-0.065573	-0.279756	-0.043520	-0.086072	-0.447008	0.268598	0.303435	1.000000	0.149194	-0.130189	0.043415
Demand Score	0.095928	-0.035854	-0.088590	-0.268661	-0.045502	-0.232489	0.017333	0.053011	0.149194	1.000000	-0.798863	0.097320
Hotness Rank	-0.107527	0.088315	0.100125	0.170073	-0.064947	0.141616	-0.069405	-0.013252	-0.130189	-0.798863	1.000000	-0.153072
Hotness Rank Mm	-0.010092	-0.508700	-0.155278	-0.018193	-0.020810	-0.091095	0.020239	0.033772	0.043415	0.097320	-0.153072	1.000000
Hotness Rank Yy	-0.064536	-0.151377	-0.529468	-0.074959	-0.006487	-0.408545	0.073430	0.053134	0.260651	0.278672	-0.342328	0.301847
Hotness Score	0.106691	-0.090573	-0.104870	-0.178096	0.066947	-0.145804	0.060382	0.011021	0.128789	0.804682	-0.991518	0.143849
Household Rank	-0.035628	-0.008775	-0.075846	-0.409578	0.012034	0.046412	-0.175466	0.025075	0.046874	0.185555	0.025241	0.011304
Ldp Unique Viewers Per Property Mm	0.007273	0.130309	0.214428	0.004356	-0.542419	-0.089725	-0.028821	0.162345	0.011132	0.050939	0.010274	0.165906
Ldp Unique Viewers Per Property Vs Us	0.084973	-0.046203	-0.069325	-0.246850	-0.078881	-0.217639	0.013317	0.064115	0.138881	0.938262	-0.762150	0.083548
Ldp Unique Viewers Per Property Yy	0.040640	-0.224590	-0.407494	-0.113617	-0.221999	-0.787839	0.053838	0.115644	0.442456	0.199903	-0.134179	0.116897
Median Days On Market	-0.055681	0.153948	0.375128	0.057866	-0.394274	0.158089	-0.097791	0.092952	-0.133610	-0.208744	0.547891	-0.075138
Median Days On Market Mm	0.006419	0.915465	0.227440	0.000950	-0.138538	0.192826	0.001720	-0.059357	-0.084586	-0.026950	0.074087	-0.523592
median days on market mm_hotness	0.006425	0.915465	0.227440	0.000951	-0.138538	0.192827	0.001720	-0.059358	-0.084588	-0.026952	0.074088	-0.523588
Median Days On Market Yy	0.039472	0.211491	0.914344	0.085702	-0.274375	0.399349	-0.071056	0.081933	-0.254414	-0.105659	0.195082	-0.136499
median days on market yy_hotness	0.039471	0.211490	0.914343	0.085701	-0.274374	0.399349	-0.071055	0.081933	-0.254413	-0.105658	0.195081	-0.136497
Median Dom Vs Us	-0.073144	0.134700	0.125705	0.015453	-0.126537	0.042649	-0.057349	0.030675	-0.068610	-0.269711	0.721532	-0.096764
Median Listing Price	0.063430	-0.000953	-0.030097	0.156813	0.015526	-0.031635	0.919130	0.042457	0.228999	0.001898	-0.110128	0.014031
Median Listing Price Mm	-0.007380	-0.118579	0.035158	-0.008548	-0.143835	-0.053857	0.023106	0.473054	0.135993	0.054248	-0.028000	0.033390
median listing price mm_hotness	-0.007371	-0.118584	0.035162	-0.008547	-0.143847	-0.053851	0.023120	0.473054	0.135995	0.054246	-0.028005	0.033389
Median Listing Price Per Square Foot	0.098381	0.012273	-0.028381	0.204490	0.021313	0.007259	0.856139	0.019909	0.152906	-0.043542	-0.044723	0.012228
Median Listing Price Per Square Foot Mm	-0.004120	-0.087449	-0.058250	0.026056	-0.144580	-0.127338	0.036206	0.344013	0.179903	0.081734	-0.077201	0.033202
Median Listing Price Per Square Foot Yy	0.004228	0.029458	-0.255161	0.060341	0.078484	-0.231909	0.185810	0.055812	0.495553	0.128551	-0.201243	-0.005849
Median Listing Price Vs Us	0.064251	-0.013257	0.015708	0.168915	-0.016053	-0.007117	0.905963	0.051258	0.209440	0.003536	-0.111637	0.014990
Median Listing Price Yy	-0.026089	-0.038038	-0.138386	-0.044651	-0.102801	-0.279214	0.157659	0.161970	0.635176	0.135945	-0.130773	0.027181
median listing price yy_hotness	-0.026089	-0.038032	-0.138388	-0.044647	-0.102801	-0.279216	0.157657	0.161972	0.635177	0.135945	-0.130777	0.027183
Median Square Feet	-0.113210	-0.047661	0.034138	-0.074154	-0.021086	-0.085054	0.127968	0.073567	0.211264	-0.032637	-0.079403	0.010214
Median Square Feet Mm	-0.011318	-0.080911	0.080324	-0.026324	-0.042983	0.016458	-0.001814	0.332617	0.047042	0.011052	0.018298	-0.001245
Median Square Feet Yy	-0.028831	-0.075087	0.085977	-0.095335	-0.187572	-0.090681	-0.034756	0.135332	0.278232	0.041044	0.043225	0.035975
New Listing Count	0.021365	-0.024770	0.056507	0.887919	0.060034	0.033825	0.209199	-0.014202	-0.023119	-0.266800	0.099775	-0.012835
New Listing Count Mm	-0.002293	-0.228351	0.104086	0.011034	0.080912	-0.017257	-0.022866	0.067290	-0.010998	0.044812	-0.022198	0.162981
New Listing Count Yy	0.032415	-0.137752	-0.322347	-0.016672	0.189905	0.031885	0.014938	-0.041430	0.041410	-0.021557	-0.001761	0.054800
Nielsen Hh Rank	-0.035628	-0.008775	-0.075846	-0.409578	0.012034	0.046412	-0.175466	0.025075	0.046874	0.185555	0.025241	0.011304
Pending Listing Count	0.030454	0.004717	0.047291	0.835893	0.014299	-0.008005	0.196401	-0.009179	-0.003261	-0.272360	0.144815	-0.018939
Pending Listing Count Mm	0.015275	0.007307	-0.008041	-0.002620	-0.067644	-0.006661	0.019207	0.083187	0.023728	-0.014792	0.007546	0.033243
Pending Listing Count Yy	0.008129	0.010404	0.064077	-0.003654	-0.024914	-0.000967	0.019314	0.004142	0.006034	-0.076720	0.080178	-0.010482
Pending Ratio	0.072273	-0.039802	-0.191032	-0.096211	0.036089	-0.383696	0.063879	0.006304	0.218745	0.173319	-0.180391	0.015799
Pending Ratio Mm	0.014116	-0.138065	0.140498	-0.006409	-0.594986	-0.189110	0.004332	0.269536	0.088531	0.031963	-0.003780	0.015592
Pending Ratio Yy	0.040511	-0.124519	-0.252906	-0.084529	-0.087454	-0.637727	0.028975	0.062802	0.341528	0.134929	-0.090424	0.021808
Price Increased Count	0.046081	-0.025747	0.015770	0.574588	-0.018363	-0.044697	0.145934	0.008193	0.011738	-0.292049	0.131598	-0.014912
Price Increased Count Mm	0.007511	-0.037535	0.028635	-0.003667	-0.075136	0.000942	0.003891	0.027524	0.000715	-0.031747	0.022424	0.004554
Price Increased Count Yy	0.007952	-0.078931	-0.121897	-0.017513	-0.054976	-0.158229	-0.009967	0.040652	0.057165	-0.075189	0.077457	-0.005149
Price Reduced Count	0.007293	0.010009	0.111422	0.928124	0.036157	0.115221	0.152903	-0.016728	-0.067616	-0.276484	0.140865	-0.024554
Price Reduced Count Mm	-0.004978	-0.105423	-0.015387	-0.009680	0.437635	0.044552	-0.023374	-0.055925	-0.040304	0.024560	-0.054361	0.081486
Price Reduced Count Yy	0.019222	0.130472	0.113264	-0.008436	0.178245	0.661799	-0.063335	-0.089289	-0.238515	-0.111669	0.070023	-0.052529
Quality Flag	0.036514	0.045840	0.113044	-0.049454	0.144675	0.147560	0.098223	0.032747	-0.013107	0.057001	-0.060395	-0.018551
quality flag_hotness	0.027651	0.055840	0.154768	-0.041734	0.107987	0.068067	0.094711	0.055271	-0.008980	0.067298	-0.041791	-0.005640
Supply Score	0.074227	-0.109818	-0.078814	-0.012465	0.155106	0.002672	0.079957	-0.036590	0.055309	0.275931	-0.784937	0.132956
Total Listing Count	0.027011	0.006334	0.081080	0.973679	0.014002	0.059673	0.195915	-0.005865	-0.028395	-0.279728	0.165890	-0.019246
Total Listing Count Mm	0.012469	-0.245090	-0.064242	0.014301	0.489837	0.056189	-0.006676	-0.008083	-0.010670	-0.017388	-0.060009	-0.019279
Total Listing Count Yy	0.080416	0.081261	0.328752	0.059121	0.109395	0.530479	-0.054209	-0.040891	-0.224187	-0.164512	0.137017	-0.085880
year	0.000000	0.040329	-0.366282	-0.091525	0.157289	-0.150586	0.080105	-0.040107	0.119947	0.001844	0.000068	0.002249
month	-0.000000	0.291721	-0.026378	-0.022161	0.034892	0.055737	0.037096	-0.108106	0.014983	-0.001172	0.001208	-0.001076
Rank	-0.035442	-0.008937	-0.075443	-0.407659	0.010169	0.041034	-0.198316	0.020448	0.029500	0.192374	0.040731	0.011082
2021_estimate	0.032219	0.007919	0.075072	0.871316	0.008558	0.038522	0.253533	-0.010317	-0.018532	-0.230772	0.070427	-0.018271
2020_Census	0.032682	0.007894	0.075059	0.872357	0.008757	0.039298	0.254408	-0.010327	-0.018699	-0.229796	0.071103	-0.018364
%_change	0.018342	-0.003237	-0.052047	0.081583	-0.004444	0.003857	0.101061	0.042844	0.125086	-0.155897	0.101804	0.021701

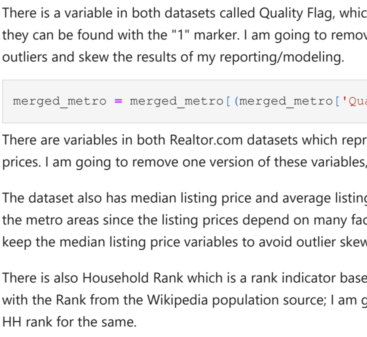
In [84]:

```
sns.boxplot(merged_metro['Average Listing Price'])
```

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[84]:

```
<AxesSubplot:xlabel='Average Listing Price'>
```



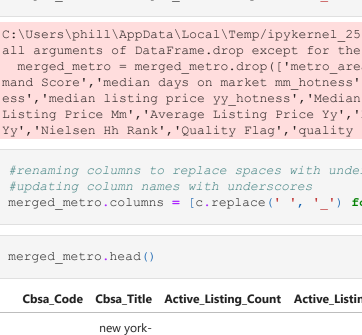
In [85]:

```
sns.boxplot(merged_metro['Median Listing Price'])
```

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[85]:

```
<AxesSubplot:xlabel='Median Listing Price'>
```



There is a variable in both datasets called Quality Flag, which was created to indicate data values that are outside of the typical range ... they can be found with the "1" marker. I am going to remove any of these values from the dataset, since they can potentially represent outliers and skew the results of my reporting/modeling.

In [86]:

```
merged_metro = merged_metro[(merged_metro['Quality Flag']==0) | (merged_metro['quality flag_hotness']==0)]
```

There are variables in both Realtor.com datasets which represent the same measures such as median days on market & median listing prices. I am going to remove one version of these variables, since they are highly correlated.

The dataset also has median listing price and average listing price. In the case of different housing markets, there can be outliers among the metro areas since the listing prices depend on many factors such as location, square feet, year built, etc. Therefore, I am just going to keep the median listing price variables to avoid outlier skew.

There is also Household Rank which is a rank indicator based on the household count for a metro area. This variable is highly correlated with the Rank from the Wikipedia population source; I am going to only keep the population one. I am also going to take out the Nielsen HH rank for the same.

I am not very interested in the Pending Ratio variables, so I am going to remove them.

In [87]:

```
#dropping columns for reduction & simplicity
merged_metro = merged_metro.drop(['metro_area','combined_statistical_area','Hotness Score','Supply Score','Dema
```

```
C:\Users\phill\AppData\Local\Temp/ipykernel_25828/1686688935.py:2: FutureWarning: In a future version of pandas
all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  merged_metro = merged_metro.drop(['metro_area','combined_statistical_area','Hotness Score','Supply Score','De
mand Score','median days on market mm_hotness','median days on market yy_hotness','median listing price mm_hotn
ess','median listing price yy_hotness','Median Dom Mm Day','Median Dom Yy Day','Average Listing Price','Average
Listing Price Mm','Average Listing Price Yy','Household Rank','Pending Ratio','Pending Ratio Mm','Pending Ratio
Yy','Nielsen Hh Rank','Quality Flag','quality flag_hotness'],1)
```

In [88]:

```
#renaming columns to replace spaces with underscores
#updating column names with underscores
merged_metro.columns = [c.replace(' ', '_') for c in merged_metro.columns]
```

In [89]:

```
merged_metro.head()
```

Out[89]:

	Cbsa_Code	Cbsa_Title	Active_Listing_Count	Active_Listing_Count_Mm	Active_Listing_Count_Yy	Hotness_Rank	Hotness_Rank_Mm	Hotness_Ra
0	35620	new york-newark-jersey city, ny-nj-pa	46763	-0.1382	-0.1824	298	-1.0	
1	35620	new york-newark-jersey city, ny-nj-pa	54263	-0.0524	-0.1738	297	-6.0	
2	35620	new york-newark-jersey city, ny-nj-pa	57262	0.0060	-0.1657	291	-2.0	
3	35620	new york-newark-jersey city, ny-nj-pa	56921	-0.0022	-0.1521	289	6.0	
4	35620	new york-newark-jersey city, ny-nj-pa	57049	0.0366	-0.1444	295	3.0	

5 rows × 46 columns

In [90]:

```
merged_metro.dtypes

#may need to convert month and year to date objects
```

Out[90]:

```
Cbsa_Code                                    int64
Cbsa_Title                                  object
Active_Listing_Count                         int64
Active_Listing_Count_Mm                    float64
Active_Listing_Count_Yy                    float64
Hotness_Rank                                 int64
Hotness_Rank_Mm                            float64
Hotness_Rank_Yy                            float64
Ldp_Unique_Viewers_Per_Property_Mm         float64
Ldp_Unique_Viewers_Per_Property_Vs_Us      float64
Ldp_Unique_Viewers_Per_Property_Yy         float64
Median_Days_On_Market                        int64
Median_Days_On_Market_Mm                   float64
Median_Days_On_Market_Yy                   float64
Median_Dom_Vs_Us                           float64
Median_Listing_Price                         int64
Median_Listing_Price_Mm                    float64
Median_Listing_Price_Per_Square_Foot         int64
Median_Listing_Price_Per_Square_Foot_Mm    float64
Median_Listing_Price_Per_Square_Foot_Yy    float64
Median_Listing_Price_Vs_Us                 float64
Median_Listing_Price_Yy                    float64
Median_Square_Feet                           int64
Median_Square_Feet_Mm                      float64
Median_Square_Feet_Yy                      float64
New_Listing_Count                            int64
New_Listing_Count_Mm                       float64
New_Listing_Count_Yy                       float64
Pending_Listing_Count                      float64
Pending_Listing_Count_Mm                   float64
Pending_Listing_Count_Yy                   float64
Price_Increased_Count                        int64
Price_Increased_Count_Mm                   float64
Price_Increased_Count_Yy                   float64
Price_Reduced_Count                          int64
Price_Reduced_Count_Mm                     float64
Price_Reduced_Count_Yy                     float64
Total_Listing_Count                          int64
Total_Listing_Count_Mm                     float64
Total_Listing_Count_Yy                     float64
year                                         int64
month                                        int64
Rank                                         int64
2021_estimate                                int64
2020_Census                                  int64
%_change                                   float64
dtype: object
```

In [91]:

```
merged_metro['Hotness_Rank'].describe()
```

Out[91]:

```
count    5317.000000
mean      154.785029
std        86.654318
min         1.000000
25%        79.000000
50%       158.000000
75%       230.000000
max       300.000000
Name: Hotness_Rank, dtype: float64
```

In [92]:

```
merged_metro.describe()
```

Out[92]:

	Cbsa_Code	Active_Listing_Count	Active_Listing_Count_Mm	Active_Listing_Count_Yy	Hotness_Rank	Hotness_Rank_Mm	Hotness_Rank_Yy
count	5317.000000	5317.000000	5317.000000	5317.000000	5317.000000	5317.000000	5317.00000
mean	29403.737070	1927.933045	-0.035330	-0.339228	154.785029	0.035923	1.05529
std	11594.896853	4917.257988	0.105403	0.215077	86.654318	25.887266	53.1980
min	10180.000000	77.000000	-0.564300	-0.790000	1.000000	-218.000000	-204.00000
25%	19300.000000	400.000000	-0.096600	-0.498200	79.000000	-13.000000	-27.00000
50%	29180.000000	729.000000	-0.043300	-0.364200	158.000000	0.000000	0.00000
75%	39820.000000	1525.000000	0.016600	-0.197100	230.000000	12.000000	29.00000
max	49740.000000	68637.000000	1.578700	1.534600	300.000000	157.000000	230.00000

8 rows × 45 columns

There seem like there might be outliers in the listing count variables ... I am going to investigate and decide if they should be removed.

In [93]:
```
sns.boxplot(merged_metro['Active_Listing_Count'])
```

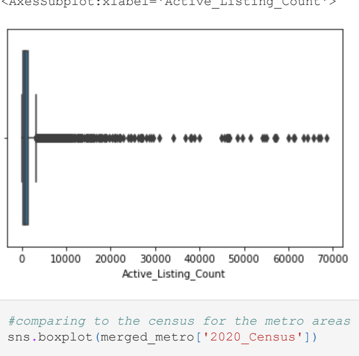
```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[93]: <AxesSubplot:xlabel='Active_Listing_Count'>

In [94]:
```
#comparing to the census for the metro areas
sns.boxplot(merged_metro['2020_Census'])
```

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[94]: <AxesSubplot:xlabel='2020_Census'>



In [95]:
```
# Position of the Outlier
print(np.where(merged_metro['Active_Listing_Count']>1450))
```

```
(array([   0,    1,    2, ..., 4637, 4638, 4639], dtype=int64),)
```

In [96]:
```
#looking at metro areas with above distribution listing counts
merged_metro[merged_metro['Active_Listing_Count']>1450]
```

Out[96]:

	Cbsa_Code	Cbsa_Title	Active_Listing_Count	Active_Listing_Count_Mm	Active_Listing_Count_Yy	Hotness_Rank	Hotness_Rank_Mm	Hotness
0	35620	new york-newark-jersey city, ny-nj-pa	46763	-0.1382	-0.1824	298	-1.0	
1	35620	new york-newark-jersey city, ny-nj-pa	54263	-0.0524	-0.1738	297	-6.0	
2	35620	new york-newark-jersey city, ny-nj-pa	57262	0.0060	-0.1657	291	-2.0	
3	35620	new york-newark-jersey city, ny-nj-pa	56921	-0.0022	-0.1521	289	6.0	
4	35620	new york-newark-jersey city, ny-nj-pa	57049	0.0366	-0.1444	295	3.0	
...	...	...	...	...	...	...	...	
5275	42680	sebastian-vero beach, fl	1927	-0.0365	-0.1058	283	5.0	
5276	42680	sebastian-vero beach, fl	2000	0.0025	-0.1047	288	-22.0	
5277	42680	sebastian-vero beach, fl	1995	0.0096	-0.1277	266	-33.0	
5278	42680	sebastian-vero beach, fl	1976	0.0123	-0.1401	233	-34.0	
5279	42680	sebastian-vero beach, fl	1952	0.0199	-0.0942	199	-9.0	

1404 rows × 46 columns

In [97]:
```
#investigating relationship between metro population and listing count
# Scatter plot
fig, ax = plt.subplots(figsize = (18,10))
ax.scatter(merged_metro['2020_Census'], merged_metro['Active_Listing_Count'])

# x-axis label
ax.set_xlabel('2020 Metro Area Population')

# y-axis label
ax.set_ylabel('Active Listing Count')
plt.show()

#slight positive relationship -- increase together!
```

Given that there seems to be a positive relationship between metro area population and the area's listing counts, I think the outliers depicted are valid since there are areas represented in the dataset with large populations such as New York!

In [98]:
```
#look at duplicates and missing values with final dataset

#duplicates? --> none!
duplicate_rows_metro = merged_metro[merged_metro.duplicated()]
print("number of duplicate rows: ", duplicate_rows_metro.shape)

# Finding the null values
print("\nNull values")
print(merged_metro.isnull().sum())
```

```
number of duplicate rows:  (0, 46)

Null values
Cbsa_Code                                     0
Cbsa_Title                                    0
Active_Listing_Count                          0
Active_Listing_Count_Mm                       0
Active_Listing_Count_Yy                       0
Hotness_Rank                                  0
Hotness_Rank_Mm                               0
Hotness_Rank_Yy                               0
Ldp_Unique_Viewers_Per_Property_Mm            0
Ldp_Unique_Viewers_Per_Property_Vs_Us         0
Ldp_Unique_Viewers_Per_Property_Yy            0
Median_Days_On_Market                         0
Median_Days_On_Market_Mm                      0
Median_Days_On_Market_Yy                      0
Median_Dom_Vs_Us                              0
Median_Listing_Price                          0
Median_Listing_Price_Mm                       0
Median_Listing_Price_Per_Square_Foot          0
Median_Listing_Price_Per_Square_Foot_Mm       0
Median_Listing_Price_Per_Square_Foot_Yy       0
Median_Listing_Price_Vs_Us                    0
Median_Listing_Price_Yy                       0
Median_Square_Feet                            0
Median_Square_Feet_Mm                         0
Median_Square_Feet_Yy                         0
New_Listing_Count                             0
New_Listing_Count_Mm                          0
New_Listing_Count_Yy                          0
Pending_Listing_Count                        34
Pending_Listing_Count_Mm                     39
Pending_Listing_Count_Yy                     70
Price_Increased_Count                         0
Price_Increased_Count_Mm                    532
Price_Increased_Count_Yy                    515
Price_Reduced_Count                           0
Price_Reduced_Count_Mm                        0
Price_Reduced_Count_Yy                        0
Total_Listing_Count                           0
Total_Listing_Count_Mm                        0
Total_Listing_Count_Yy                        0
year                                          0
month                                         0
Rank                                          0
2021_estimate                                 0
2020_Census                                   0
%_change                                     22
dtype: int64
```

In [99]:
```
#looking at rows with missing data
nulls = merged_metro[merged_metro.isnull().any(axis=1)]
```

In [100...
```
nulls[nulls['%_change'].isnull()]
```

Out[100...

	Cbsa_Code	Cbsa_Title	Active_Listing_Count	Active_Listing_Count_Mm	Active_Listing_Count_Yy	Hotness_Rank	Hotness_Rank_Mm	Hotn
1008	10580	albany-schenectady-troy, ny	1381	-0.1204	-0.3312	281	4.0	
1009	10580	albany-schenectady-troy, ny	1570	-0.0649	-0.3236	285	-6.0	
1010	10580	albany-schenectady-troy, ny	1679	-0.0509	-0.2966	279	3.0	
1011	10580	albany-schenectady-troy, ny	1769	-0.0210	-0.2623	282	-1.0	
1012	10580	albany-schenectady-troy, ny	1807	0.0782	-0.2446	281	3.0	
1013	10580	albany-schenectady-troy, ny	1676	0.0799	-0.3176	284	-5.0	
1014	10580	albany-schenectady-troy, ny	1552	0.1182	-0.3926	279	6.0	
1015	10580	albany-schenectady-troy, ny	1388	0.0297	-0.4686	285	-1.0	
1016	10580	albany-schenectady-troy, ny	1348	-0.0189	-0.4944	284	7.0	
1017	10580	albany-schenectady-troy, ny	1374	-0.0672	-0.5129	291	0.0	
1018	10580	albany-schenectady-troy, ny	1473	-0.1734	-0.4866	291	-1.0	
1019	10580	albany-schenectady-troy, ny	1782	-0.1370	-0.4201	290	-15.0	
1020	10580	albany-schenectady-troy, ny	2065	-0.1103	-0.3889	275	-6.0	
1021	10580	albany-schenectady-troy, ny	2321	-0.0276	-0.3852	269	-44.0	
1022	10580	albany-schenectady-troy, ny	2387	-0.0046	-0.4307	225	-12.0	
1023	10580	albany-schenectady-troy, ny	2398	0.0025	-0.4274	213	26.0	
1024	10580	albany-schenectady-troy, ny	2392	-0.0261	-0.4147	239	12.0	
1025	10580	albany-schenectady-troy, ny	2456	-0.0387	-0.3785	251	4.0	
1028	10580	albany-schenectady-troy, ny	2666	-0.0549	-0.2738	271	-15.0	
1029	10580	albany-schenectady-troy, ny	2821	-0.0167	-0.1641	256	7.0	
1030	10580	albany-schenectady-troy, ny	2869	-0.0664	-0.1280	263	7.0	
1031	10580	albany-schenectady-troy, ny	3073	-0.0906	-0.1368	270	-3.0	

22 rows × 46 columns

For the metro area, Albany, when it was read in from Wikipedia, Python converted the 0% rate of change as NaN. I am going to update this value so the data is correctly reflected.

In [101...
```
#if albany is the metro area, apply %_change value as zero (0)
#else, leave %_change value as-is for other areas
merged_metro['%_change'] = np.where(merged_metro['Cbsa_Title'].eq('albany-schenectady-troy, ny'), 0, merged_met
```

In [102...
```
#looking at rows with missing data again
nulls = merged_metro[merged_metro.isnull().any(axis=1)]
```

In [103...
```
#albany's value have been properly updated
nulls[nulls['%_change'].isnull()]
```

Out[103...

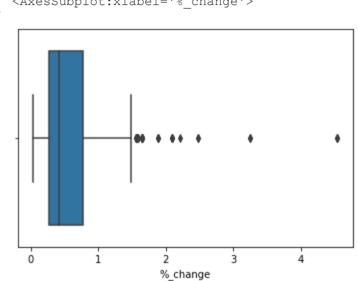
Cbsa_Code	Cbsa_Title	Active_Listing_Count	Active_Listing_Count_Mm	Active_Listing_Count_Yy	Hotness_Rank	Hotness_Rank_Mm	Hotness_Ran

0 rows × 46 columns

In [104...
```
sns.boxplot(nulls['%_change'])
```

```
C:\Users\phill\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variabl
e as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[104...: <AxesSubplot:xlabel='%_change'>



In [105...
```
print(nulls.isnull().sum())
```

```
Cbsa_Code                                     0
Cbsa_Title                                    0
Active_Listing_Count                          0
Active_Listing_Count_Mm                       0
Active_Listing_Count_Yy                       0
Hotness_Rank                                  0
Hotness_Rank_Mm                               0
Hotness_Rank_Yy                               0
Ldp_Unique_Viewers_Per_Property_Mm            0
Ldp_Unique_Viewers_Per_Property_Vs_Us         0
Ldp_Unique_Viewers_Per_Property_Yy            0
Median_Days_On_Market                         0
Median_Days_On_Market_Mm                      0
Median_Days_On_Market_Yy                      0
Median_Dom_Vs_Us                              0
Median_Listing_Price                          0
Median_Listing_Price_Mm                       0
Median_Listing_Price_Per_Square_Foot          0
Median_Listing_Price_Per_Square_Foot_Mm       0
Median_Listing_Price_Per_Square_Foot_Yy       0
Median_Listing_Price_Vs_Us                    0
Median_Listing_Price_Yy                       0
Median_Square_Feet                            0
Median_Square_Feet_Mm                         0
Median_Square_Feet_Yy                         0
New_Listing_Count                             0
New_Listing_Count_Mm                          0
New_Listing_Count_Yy                          0
Pending_Listing_Count                        34
Pending_Listing_Count_Mm                     39
Pending_Listing_Count_Yy                     70
Price_Increased_Count                         0
Price_Increased_Count_Mm                    532
Price_Increased_Count_Yy                    515
Price_Reduced_Count                           0
Price_Reduced_Count_Mm                        0
Price_Reduced_Count_Yy                        0
Total_Listing_Count                           0
Total_Listing_Count_Mm                        0
Total_Listing_Count_Yy                        0
year                                          0
month                                         0
Rank                                          0
2021_estimate                                 0
2020_Census                                   0
%_change                                      0
dtype: int64
```

For the sake of data independence and since the amount of missing values are not too significant in relation to the overall dataset size, I am just going to remove the rows with missing data and will come back if it has a major effect on modeling.

In [106...
```
merged_metro = merged_metro.dropna()
```

In [107...
```
merged_metro.shape
```

Out[107...
```
(4472, 46)
```

In [108...
```
#unique areas represented in the final dataset
merged_metro['Cbsa_Title'].unique()
```

Out[108...
```
array(['new york-newark-jersey city, ny-nj-pa',
       'los angeles-long beach-anaheim, ca',
       'chicago-naperville-elgin, il-in-wi',
       'dallas-fort worth-arlington, tx',
       'houston-the woodlands-sugar land, tx',
       'philadelphia-camden-wilmington, pa-nj-de-md',
       'washington-arlington-alexandria, dc-va-md-wv',
       'miami-fort lauderdale-west palm beach, fl',
       'boston-cambridge-newton, ma-nh', 'seattle-tacoma-bellevue, wa',
       'minneapolis-st. paul-bloomington, mn-wi',
       'riverside-san bernardino-ontario, ca',
       'tampa-st. petersburg-clearwater, fl', 'st. louis, mo-il',
       'denver-aurora-lakewood, co', 'baltimore-columbia-towson, md',
       'pittsburgh, pa', 'portland-vancouver-hillsboro, or-wa',
       'charlotte-concord-gastonia, nc-sc',
       'orlando-kissimmee-sanford, fl', 'cleveland-elyria, oh',
       'san antonio-new braunfels, tx', 'cincinnati, oh-ky-in',
       'kansas city, mo-ks', 'columbus, oh',
       'indianapolis-carmel-anderson, in',
       'las vegas-henderson-paradise, nv', 'austin-round rock, tx',
       'san jose-sunnyvale-santa clara, ca',
       'virginia beach-norfolk-newport news, va-nc',
       'providence-warwick, ri-ma', 'jacksonville, fl',
       'oklahoma city, ok', 'louisville/jefferson county, ky-in',
       'memphis, tn-ms-ar', 'new orleans-metairie, la', 'richmond, va',
       'birmingham-hoover, al', 'rochester, ny', 'tucson, az',
       'salt lake city, ut', 'tulsa, ok', 'albany-schenectady-troy, ny',
       'albuquerque, nm', 'omaha-council bluffs, ne-ia',
       'worcester, ma-ct', 'knoxville, tn',
       'bridgeport-stamford-norwalk, ct',
       'north port-sarasota-bradenton, fl', 'new haven-milford, ct',
       'dayton, oh', 'urban honolulu, hi',
       'allentown-bethlehem-easton, pa-nj', 'columbia, sc',
       'baton rouge, la', 'greensboro-high point, nc', 'fresno, ca',
       'charleston-north charleston, sc',
       'little rock-north little rock-conway, ar',
       'cape coral-fort myers, fl', 'akron, oh', 'el paso, tx',
       'oxnard-thousand oaks-ventura, ca', 'colorado springs, co',
       'madison, wi', 'bakersfield, ca', 'winston-salem, nc',
       'syracuse, ny', 'deltona-daytona beach-ormond beach, fl',
       'boise city, id', 'wichita, ks', 'toledo, oh',
       'des moines-west des moines, ia', 'springfield, ma',
       'lakeland-winter haven, fl', 'palm bay-melbourne-titusville, fl',
       'mcallen-edinburg-mission, tx', 'harrisburg-carlisle, pa',
       'augusta-richmond county, ga-sc',
       'youngstown-warren-boardman, oh-pa', 'durham-chapel hill, nc',
       'portland-south portland, me', 'spokane-spokane valley, wa',
       'chattanooga, tn-ga', 'jackson, ms', 'ogden-clearfield, ut',
       'lexington-fayette, ky', 'lancaster, pa', 'asheville, nc',
       'lafayette, la', 'lansing-east lansing, mi',
       'pensacola-ferry pass-brent, fl', 'springfield, mo',
       'myrtle beach-conway-north myrtle beach, sc-nc',
       'port st. lucie, fl', 'huntsville, al',
       'shreveport-bossier city, la', 'reno, nv', 'modesto, ca',
       'york-hanover, pa', 'corpus christi, tx', 'fort wayne, in',
       'canton-massillon, oh', 'flint, mi', 'provo-orem, ut',
       'mobile, al', 'manchester-nashua, nh',
       'davenport-moline-rock island, ia-il', 'salisbury, md-de',
       'killeen-temple, tx', 'reading, pa', 'beaumont-port arthur, tx',
       'peoria, il', 'fayetteville, nc', 'tallahassee, fl',
       'santa maria-santa barbara, ca', 'huntington-ashland, wv-ky-oh',
       'salem, or', 'savannah, ga', 'ann arbor, mi',
       'hickory-lenoir-morganton, nc', 'ocala, fl', 'montgomery, al',
       'fort collins, co', 'kalamazoo-portage, mi', 'salinas, ca',
       'roanoke, va', 'boulder, co', 'rockford, il', 'lincoln, ne',
       'evansville, in-ky', 'green bay, wi', 'brownsville-harlingen, tx',
       'spartanburg, sc', 'south bend-mishawaka, in-mi',
       'columbus, ga-al', 'utica-rome, ny', 'duluth, mn-wi',
       'lubbock, tx', 'wilmington, nc', 'gainesville, fl', 'erie, pa',
       'cedar rapids, ia', 'fort smith, ar-ok',
       'crestview-fort walton beach-destin, fl', 'norwich-new london, ct',
       'clarksville, tn-ky', 'lynchburg, va',
       'atlantic city-hammonton, nj', 'greeley, co', 'sioux falls, sd',
       'binghamton, ny', 'amarillo, tx', 'santa cruz-watsonville, ca',
       'hagerstown-martinsburg, md-wv', 'fargo, nd-mn',
       'kennewick-richland, wa', 'waco, tx', 'champaign-urbana, il',
       'barnstable town, ma', 'charleston, wv', 'tuscaloosa, al',
       'topeka, ks', 'college station-bryan, tx', 'appleton, wi',
       'chico, ca', 'charlottesville, va', 'medford, or',
       'burlington-south burlington, vt', 'rochester, mn',
       'bellingham, wa', 'johnson city, tn', 'tyler, tx',
       'hilton head island-bluffton-beaufort, sc',
       'lake havasu city-kingman, az', 'daphne-fairhope-foley, al',
       'yakima, wa', 'lafayette-west lafayette, in', 'longview, tx',
       'florence, sc', 'lake charles, la', 'punta gorda, fl',
       'panama city, fl', 'merced, ca', 'las cruces, nm',
       'houma-thibodaux, la', 'saginaw, mi', 'racine, wi',
       'athens-clarke county, ga', 'st. cloud, mn', 'bloomington, il',
       'laredo, tx', 'redding, ca', 'elkhart-goshen, in',
       'warner robins, ga', 'greenville, nc', 'billings, mt',
       'kingston, ny', 'oshkosh-neenah, wi', 'monroe, la', 'columbia, mo',
       'waterloo-cedar falls, ia', 'jacksonville, nc', 'iowa city, ia',
       'joplin, mo', 'eau claire, wi', 'muskegon, mi', 'bloomington, in',
       'yuma, az', 'gainesville, ga', 'terre haute, in', 'santa fe, nm',
       'bowling green, ky', 'pueblo, co', 'dover, de', 'abilene, tx',
       'sebastian-vero beach, fl', 'janesville-beloit, wi',
       'homosassa springs, fl', 'bangor, me', 'auburn-opelika, al',
       'midland, tx', 'burlington, nc', 'florence-muscle shoals, al',
       'jackson, mi', 'rapid city, sd', 'decatur, al', 'dothan, al',
       'state college, pa', 'chambersburg-waynesboro, pa',
       "coeur d'alene, id", 'east stroudsburg, pa', 'rocky mount, nc',
       'grand junction, co', 'kahului-wailuku-lahaina, hi',
       'alexandria, la', 'albany, ga', 'hattiesburg, ms',
       'jefferson city, mo', 'monroe, mi', 'elizabethtown-fort knox, ky',
       'texarkana, tx-ar', 'wichita falls, tx', 'morgantown, wv',
       'the villages, fl', 'johnstown, pa', 'yuba city, ca',
       'pittsfield, ma', 'bismarck, nd', 'springfield, oh',
       'battle creek, mi', 'lebanon, pa'], dtype=object)
```

CBSA Codes with Hotness Rank = 1

In [109...
```
merged_metro[merged_metro['Hotness_Rank']==1]
```

Out[109...

	Cbsa_Code	Cbsa_Title	Active_Listing_Count	Active_Listing_Count_Mm	Active_Listing_Count_Yy	Hotness_Rank	Hotness_Rank_Mm	Hotne
1533	17820	colorado springs, co	1290	0.0705	-0.2728	1	1.0	
2444	23060	fort wayne, in	554	0.0184	0.0473	1	4.0	
2544	31700	manchester-nashua, nh	208	-0.3090	-0.3791	1	1.0	
2546	31700	manchester-nashua, nh	401	-0.1225	-0.2391	1	0.0	
2547	31700	manchester-nashua, nh	457	0.0293	-0.1825	1	1.0	
2559	31700	manchester-nashua, nh	559	0.0164	-0.4625	1	0.0	
2560	31700	manchester-nashua, nh	550	0.0204	-0.4860	1	10.0	
5401	15500	burlington, nc	304	0.1095	-0.0033	1	2.0	
5404	15500	burlington, nc	267	0.2304	-0.2238	1	1.0	
5413	15500	burlington, nc	305	-0.0758	-0.5104	1	2.0	
6134	44220	springfield, oh	174	0.0482	-0.6193	1	3.0	
6137	44220	springfield, oh	183	0.0055	-0.5222	1	29.0	

12 rows × 46 columns

Now that we have our final dataset, I want to export the DataFrame to an Excel file for usage of EDA in Tableau. There are multiple business questions I want to answer, and I think this tool will work great for yielding data insights.

In [110...
```
# determining the name of the file
file_name = 'metro_housing_market.xlsx'

# saving the excel
merged_metro.to_excel(file_name)
print('DataFrame is written to Excel File successfully.')
```

```
DataFrame is written to Excel File successfully.
```

Modeling

Since there are 45 columns (46 without the target variable) in the dataset, I am going to use a feature reduction technique to reduce the number of features that are included in the classification models.

In [111...
```
merged_metro.dtypes
```

Out[111...
```
Cbsa_Code                                    int64
Cbsa_Title                                  object
Active_Listing_Count                         int64
Active_Listing_Count_Mm                    float64
Active_Listing_Count_Yy                    float64
Hotness_Rank                                 int64
Hotness_Rank_Mm                            float64
Hotness_Rank_Yy                            float64
Ldp_Unique_Viewers_Per_Property_Mm         float64
Ldp_Unique_Viewers_Per_Property_Vs_Us      float64
Ldp_Unique_Viewers_Per_Property_Yy         float64
Median_Days_On_Market                        int64
Median_Days_On_Market_Mm                   float64
Median_Days_On_Market_Yy                   float64
Median_Dom_Vs_Us                           float64
Median_Listing_Price                         int64
Median_Listing_Price_Mm                    float64
Median_Listing_Price_Per_Square_Foot         int64
Median_Listing_Price_Per_Square_Foot_Mm    float64
Median_Listing_Price_Per_Square_Foot_Yy    float64
Median_Listing_Price_Vs_Us                 float64
Median_Listing_Price_Yy                    float64
Median_Square_Feet                           int64
Median_Square_Feet_Mm                      float64
Median_Square_Feet_Yy                      float64
New_Listing_Count                            int64
New_Listing_Count_Mm                       float64
New_Listing_Count_Yy                       float64
Pending_Listing_Count                      float64
Pending_Listing_Count_Mm                   float64
Pending_Listing_Count_Yy                   float64
Price_Increased_Count                        int64
Price_Increased_Count_Mm                   float64
Price_Increased_Count_Yy                   float64
Price_Reduced_Count                          int64
Price_Reduced_Count_Mm                     float64
Price_Reduced_Count_Yy                     float64
Total_Listing_Count                          int64
Total_Listing_Count_Mm                     float64
Total_Listing_Count_Yy                     float64
year                                         int64
month                                        int64
Rank                                         int64
2021_estimate                                int64
2020_Census                                  int64
%_change                                   float64
dtype: object
```

For modeling purposes, all variables have to be numeric, so I am going to create a separate dataset for modeling purposes that has 'Cbsa_Title' removed since it is the location descriptor for the 'Cbsa_Code' variable.

Our target variable for modeling is Hotness Rank ... to try and predict the hotness of a given metro area based on the other factors in the dataset. Given that rank is actually a categorical variable (hierarchical), we will be using categorical modeling approaches (classification) to solve the problem.

In [112...
```
model_metro = merged_metro.drop(['Cbsa_Title'],axis=1)
```

In [113...
```
print(model_metro.shape)
```

```
(4472, 45)
```

The multi-class distribution of the target variable seems relatively even! There is not a startling contrast that I am concerned about and need to handle before modeling.

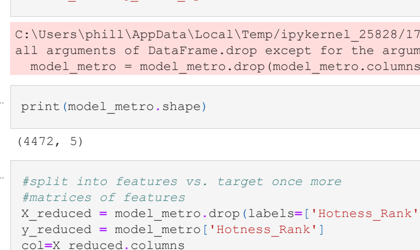
In [114...
```
#matrices of features
X = model_metro.drop(labels=['Hotness_Rank'],axis=1)
y = model_metro['Hotness_Rank']
col=X.columns
```

In [115...
```
print(col)
```

```
Index(['Cbsa_Code', 'Active_Listing_Count', 'Active_Listing_Count_Mm',
       'Active_Listing_Count_Yy', 'Hotness_Rank_Mm', 'Hotness_Rank_Yy',
       'Ldp_Unique_Viewers_Per_Property_Mm',
       'Ldp_Unique_Viewers_Per_Property_Vs_Us',
       'Ldp_Unique_Viewers_Per_Property_Yy', 'Median_Days_On_Market',
       'Median_Days_On_Market_Mm', 'Median_Days_On_Market_Yy',
       'Median_Dom_Vs_Us', 'Median_Listing_Price', 'Median_Listing_Price_Mm',
       'Median_Listing_Price_Per_Square_Foot',
       'Median_Listing_Price_Per_Square_Foot_Mm',
       'Median_Listing_Price_Per_Square_Foot_Yy', 'Median_Listing_Price_Vs_Us',
       'Median_Listing_Price_Yy', 'Median_Square_Feet',
       'Median_Square_Feet_Mm', 'Median_Square_Feet_Yy', 'New_Listing_Count',
       'New_Listing_Count_Mm', 'New_Listing_Count_Yy', 'Pending_Listing_Count',
       'Pending_Listing_Count_Mm', 'Pending_Listing_Count_Yy',
       'Price_Increased_Count', 'Price_Increased_Count_Mm',
       'Price_Increased_Count_Yy', 'Price_Reduced_Count',
       'Price_Reduced_Count_Mm', 'Price_Reduced_Count_Yy',
       'Total_Listing_Count', 'Total_Listing_Count_Mm',
       'Total_Listing_Count_Yy', 'year', 'month', 'Rank', '2021_estimate',
       '2020_Census', '%_change'],
      dtype='object')
```

In [116...
```
from sklearn.ensemble import ExtraTreesClassifier
#feature reduction
# Building the model --> extra tree classifier for feature reduction
#finds Gini Importance as selection criteria
extra_tree_forest = ExtraTreesClassifier(n_estimators = 5,
                                        criterion ='entropy', max_features = 2)

# Training the model
extra_tree_forest.fit(X, y)

# Computing the importance of each feature
feature_importance = extra_tree_forest.feature_importances_

# Normalizing the individual importances
feature_importance_normalized = np.std([tree.feature_importances_ for tree in
                                        extra_tree_forest.estimators_],
                                        axis = 0)
```

In [117...
```
# Plotting a Bar Graph to compare the models
plt.bar(X.columns, feature_importance_normalized)
plt.xlabel('Feature Labels')
plt.ylabel('Feature Importances')
plt.title('Comparison of different Feature Importances')
plt.show()
```



I moved the features and their corresponding feature importance values to a separate document, and found that there are five values with an importance measure greater than 0.008 which I am using as my cutoff.

These are the following top five features:

1. Median_Dom_Vs_Us
2. Cbsa_code
3. 2021_estimate
4. Median_Listing_Price_Mm
5. Total_Listing_Count_Yy

I am now going to reduce the model to only use these features and continue with modeling.

In [118...
```
model_metro = model_metro.drop(model_metro.columns.difference(['Hotness_Rank','Median_Dom_Vs_Us',
'Cbsa_code',
'2021_estimate',
'Median_Listing_Price_Mm',
'Total_Listing_Count_Yy']),1)
```

```
C:\Users\phill\AppData\Local\Temp/ipykernel_25828/1758456819.py:1: FutureWarning: In a future version of pandas
all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  model_metro = model_metro.drop(model_metro.columns.difference(['Hotness_Rank','Median_Dom_Vs_Us',
```

In [119...
```
print(model_metro.shape)
```

```
(4472, 5)
```

In [120...
```
#split into features vs. target once more
#matrices of features
X_reduced = model_metro.drop(labels=['Hotness_Rank'],axis=1)
y_reduced = model_metro['Hotness_Rank']
col=X_reduced.columns
```

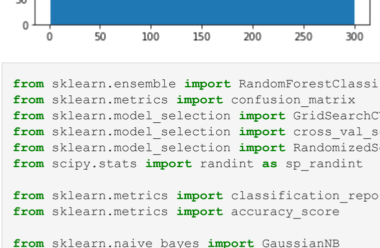
In [121...
```
#train, test, split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_reduced,y_reduced,test_size=0.2,random_state= 10)
```

In [122...
```
#scale data to [-1,1]

from sklearn.preprocessing import MinMaxScaler
scaling = MinMaxScaler(feature_range=(-1,1)).fit(X_train)
X_train = scaling.transform(X_train)
X_test = scaling.transform(X_test)
```

In [123...
```
print("X_train shape: ",X_train.shape)
print("y train shape: ",y_train.shape)
print("X test shape: ", X_test.shape)
print("y test shape: ",y_test.shape)
```

```
X_train shape:  (3577, 4)
y train shape:  (3577,)
X test shape:  (895, 4)
y test shape:  (895,)
```

In [124...
```
#looking at class distribution
#histogram
plt.hist(y_train)
plt.title('Histogram of Hotness Rank')
plt.show()
```



In [125...
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint as sp_randint

from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC

# training a linear SVM classifier
from sklearn.svm import SVC
svm_model_linear = SVC(kernel = 'linear', C = 1).fit(X_train, y_train)
svm_predictions = svm_model_linear.predict(X_test)
```

Support Vector Machine (SVM)

In [126…

```
from sklearn.metrics import precision_recall_fscore_support

# model accuracy for X_test
svm_accuracy = svm_model_linear.score(X_test, y_test)
print(svm_accuracy)

# creating a confusion matrix
svm_cm = confusion_matrix(y_test, svm_predictions)
print(svm_cm)

#getting precision and recall
svm_metrics = precision_recall_fscore_support(y_test, svm_predictions, average='macro',labels=np.unique(svm_pre
print(svm_metrics)
```

```
0.004469273743016759
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 1 0]
 [0 0 0 ... 0 2 0]
 [0 0 0 ... 0 1 0]]
(0.025314070351758797, 0.08333333333333333, 0.03187344139650873, None)
```

K-Nearest Neighbor

In [127…

```
# training a KNN classifier
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 7).fit(X_train, y_train)
```

In [128…

```
# accuracy on X_test
knn_accuracy = knn.score(X_test, y_test)
print(knn_accuracy)

# creating a confusion matrix
knn_predictions = knn.predict(X_test)
knn_cm = confusion_matrix(y_test, knn_predictions)

#getting precision and recall
knn_metrics = precision_recall_fscore_support(y_test, knn_predictions, average='macro',labels=np.unique(knn_pre
print(knn_metrics)
```

```
0.008938547486033519
(0.00953065134099617, 0.01958128078817734, 0.007683671692292383, None)
```

Naive Bayes Classifier

In [129…

```
# training a Naive Bayes classifier
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB().fit(X_train, y_train)
gnb_predictions = gnb.predict(X_test)
```

In [130…

```
# accuracy on X_test
gnb_accuracy = gnb.score(X_test, y_test)
print(gnb_accuracy)

# creating a confusion matrix
gnb_cm = confusion_matrix(y_test, gnb_predictions)

#getting precision and recall
gnb_metrics = precision_recall_fscore_support(y_test, gnb_predictions, average='macro',labels=np.unique(gnb_pre
print(gnb_metrics)
```

```
0.00782122905027933
(0.005754585488041371, 0.03063725490196078, 0.007884959539371305, None)
```

Multinomial Logistic Regression

In [131…

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RepeatedStratifiedKFold

# define the multinomial logistic regression model
model = LogisticRegression(multi_class='multinomial', solver='lbfgs')
# define the model evaluation procedure
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
# evaluate the model and collect the scores
n_scores = cross_val_score(model, X_train, y_train, scoring='accuracy', cv=cv, n_jobs=-1)
# report the model performance
print('Mean Accuracy: %.3f (%.3f)' % (np.mean(n_scores), np.std(n_scores)))
```

```
C:\Users\phill\anaconda3\lib\site-packages\sklearn\model_selection\_split.py:676: UserWarning: The least popula
ted class in y has only 5 members, which is less than n_splits=10.
  warnings.warn(
C:\Users\phill\anaconda3\lib\site-packages\sklearn\model_selection\_split.py:676: UserWarning: The least popula
ted class in y has only 5 members, which is less than n_splits=10.
  warnings.warn(
C:\Users\phill\anaconda3\lib\site-packages\sklearn\model_selection\_split.py:676: UserWarning: The least popula
ted class in y has only 5 members, which is less than n_splits=10.
  warnings.warn(
Mean Accuracy: 0.014 (0.006)
```

In [132…

```
# fit the model on the whole dataset
model.fit(X_train, y_train)
#get predictions
yhat = model.predict(X_test)
# summarize the predicted class
print('Predicted Class: %d' % yhat[0])
```

```
Predicted Class: 142
```