Property Tax Prediction Model

Load Libraries

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

Load the dataset

In [2]:
```
file_path = "../Task 1/cleaned_house_data.csv"
df = pd.read_csv(file_path)
```

Data Analysis

In [3]:
```
print("Initial Dataset Info:")
print(df.info())
```

```
Initial Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3684 entries, 0 to 3683
Data columns (total 16 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   MLS               3684 non-null   int64  
 1   sold_price        3684 non-null   float64
 2   zipcode           3684 non-null   int64  
 3   longitude         3684 non-null   float64
 4   latitude          3684 non-null   float64
 5   lot_acres         3684 non-null   float64
 6   taxes             3684 non-null   float64
 7   year_built        3684 non-null   int64  
 8   bedrooms          3684 non-null   int64  
 9   bathrooms         3684 non-null   float64
 10  sqrt_ft           3684 non-null   float64
 11  garage            3684 non-null   float64
 12  kitchen_features  3684 non-null   int64  
 13  fireplaces        3684 non-null   float64
 14  floor_covering    3684 non-null   int64  
 15  HOA               3684 non-null   float64
dtypes: float64(10), int64(6)
memory usage: 460.6 KB
None
```

In [4]:
```
df.head()
```

Out[4]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built
0	21329440	1125000.0	85718	-110.883547	32.329763	1.33	8654.00	1986
1	21500337	1100000.0	85750	-110.866891	32.321968	1.17	6565.93	1994
2	21206450	1180000.0	85750	-110.868487	32.316324	1.30	9590.16	1993
3	21224755	1175500.0	85718	-110.940650	32.347873	1.23	11674.00	2004
4	21703603	1125478.0	85755	-110.973498	32.460529	1.71	3171.39	2017

In [5]:
```
df.isna().sum()
```

Out[5]:
```
MLS                 0
sold_price          0
zipcode             0
longitude           0
latitude            0
lot_acres           0
taxes               0
year_built          0
bedrooms            0
bathrooms           0
sqrt_ft             0
garage              0
kitchen_features    0
fireplaces          0
floor_covering      0
HOA                 0
dtype: int64
```

In [6]:
```
#df = df.drop(columns=['kitchen_features','floor_covering']) # ,'fireplaces','ba
```

In [6]:
```
df
```

Out[6]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_b
0	21329440	1125000.0	85718	-110.883547	32.329763	1.33	8654.00	1
1	21500337	1100000.0	85750	-110.866891	32.321968	1.17	6565.93	1
2	21206450	1180000.0	85750	-110.868487	32.316324	1.30	9590.16	1
3	21224755	1175500.0	85718	-110.940650	32.347873	1.23	11674.00	2
4	21703603	1125478.0	85755	-110.973498	32.460529	1.71	3171.39	2
...	...	...	...	...	...	...	...	
3679	3056450	525000.0	85614	-110.980945	31.824287	3.01	5122.84	2
3680	21908358	565000.0	85750	-110.820216	32.307646	0.83	4568.71	1
3681	21909379	535000.0	85718	-110.922291	32.317496	0.18	4414.00	2
3682	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1
3683	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2

3684 rows × 16 columns

In [7]:

```
df['taxes_sqft'] = df['taxes'] / df['sqrt_ft']
```

In [8]:

```
df.head()
```

Out[8]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built
0	21329440	1125000.0	85718	-110.883547	32.329763	1.33	8654.00	1986
1	21500337	1100000.0	85750	-110.866891	32.321968	1.17	6565.93	1994
2	21206450	1180000.0	85750	-110.868487	32.316324	1.30	9590.16	1993
3	21224755	1175500.0	85718	-110.940650	32.347873	1.23	11674.00	2004
4	21703603	1125478.0	85755	-110.973498	32.460529	1.71	3171.39	2017

In [116…

```
df[['taxes_sqft','price_zone','bedrooms','bathrooms','lot_acres','year_built']]
```

Out[116...

	taxes_sqft	price_zone	bedrooms	bathrooms	lot_acres	year_built
0	1.722875	13.124431	4	5.0	1.33	1986
1	1.696623	12.827988	4	4.0	1.17	1994
2	1.906972	13.760933	4	3.0	1.30	1993
3	2.817765	13.713572	4	5.0	1.23	2004
4	0.922989	13.124343	3	4.0	1.71	2017
...	...	...	...	...	...	...
3679	1.458667	6.132175	3	3.0	3.01	2007
3680	1.624141	6.588921	4	3.0	0.83	1986
3681	2.095916	6.241396	3	2.0	0.18	2002
3682	2.080246	6.413994	4	3.0	1.42	1990
3683	1.563622	6.414368	4	4.0	1.01	2009

3684 rows × 6 columns

In [117...

```
df_sorted = df.sort_values(by='taxes_sqft', ascending=True)
print("Sorted DataFrame:")
print(df_sorted)
```

```
Sorted DataFrame:
           MLS  sold_price  zipcode   longitude   latitude  lot_acres  \
325   21224893   1000000.0    85749 -110.736241  32.258951       1.52
2481  21620101    616755.0    85658 -111.106396  32.469242       0.30
952   21402357    780000.0    85749 -110.707938  32.272160       3.28
3387  21832887    575359.5    85641 -110.687945  32.081978       1.07
2450  21626928    605000.0    85749 -110.799494  32.292655       0.55
...        ...         ...      ...         ...        ...        ...
1004  21812907    735480.0    85755 -110.976564  32.460364       1.00
771   21723890    775000.0    85755 -110.980627  32.459894       1.04
223   21804547    900000.0    85755 -110.986178  32.475395       2.93
636   21614519    786620.0    85619 -110.754519  32.443022       0.37
353   21518331    880000.0    85755 -111.010677  32.468029       2.07

         taxes  year_built  bedrooms  bathrooms  ...  garage  \
325     794.01        2012         5        5.0  ...     3.0
2481    459.53        2016         3        3.0  ...     3.0
952     540.00        1990         4        4.0  ...     3.0
3387    625.00        2019         5       36.0  ...     3.0
2450    612.25        2015         3        4.0  ...     3.0
...        ...         ...       ...        ...  ...     ...
1004  10923.39        2013         4        3.0  ...     2.0
771   10520.75        2015         3        3.0  ...     3.0
223    9215.41        2013         3        3.0  ...     3.0
636    9407.98        2005         4        3.0  ...     0.0
353   10722.00        2014         3        3.0  ...     2.0

      kitchen_features  fireplaces  floor_covering     HOA  taxes_sqft  \
325                  3         0.0               2    0.00    0.158802
2481                 4         0.0               2   44.00    0.166738
952                  5         3.0               2  137.00    0.167754
3387                 8         1.0               2   56.00    0.168011
2450                 5         0.0               3  105.00    0.172903
...                ...         ...             ...     ...         ...
1004                14         2.0               2  213.00    3.637493
771                  2         1.0               2  167.00    3.642919
223                  3         0.0               2  213.88    3.646779
636                  5         1.0               2   25.00    3.737775
353                  5         1.0               3  166.00    3.786017

      cat_qcut cat_pdCut cat_pdCut_ls price_zone
325          1         1            1  11.661944
2481         1         1            1   7.200203
952          1         1            1   9.096316
3387         1         1            1   6.718272
2450         1         1            1   7.055476
...        ...       ...          ...        ...
1004         5         5            5   8.576526
771          5         5            5   9.037374
223          5         5            5  10.495015
636          5         5            5   9.187447
353          5         5            5  10.261792

[3684 rows x 21 columns]
```

In [10]:
```
labels = [f'{i}' for i in range(1, 6)]
labels
```

Out[10]:
```
['1', '2', '3', '4', '5']
```

In [11]:
```python
# Equal-sized bins
df['cat_qcut'] = pd.qcut(df['taxes_sqft'], q=5, labels=labels)
print("\nEqual Rows Category:\n", df[['taxes_sqft', 'cat_qcut']])
```

```
Equal Rows Category:
       taxes_sqft cat_qcut
0       1.722875        3
1       1.696623        3
2       1.906972        4
3       2.817765        5
4       0.922989        1
...          ...      ...
3679    1.458667        2
3680    1.624141        2
3681    2.095916        4
3682    2.080246        4
3683    1.563622        2

[3684 rows x 2 columns]
```

In [12]:
```python
df.head()
```

Out[12]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_built
0	21329440	1125000.0	85718	-110.883547	32.329763	1.33	8654.00	1986
1	21500337	1100000.0	85750	-110.866891	32.321968	1.17	6565.93	1994
2	21206450	1180000.0	85750	-110.868487	32.316324	1.30	9590.16	1993
3	21224755	1175500.0	85718	-110.940650	32.347873	1.23	11674.00	2004
4	21703603	1125478.0	85755	-110.973498	32.460529	1.71	3171.39	2017

In [13]:
```python
# Categorize data into bins
df['cat_pdCut'] = pd.cut(df['taxes_sqft'], bins=5, labels=labels, include_lowest
```

In [14]:
```python
bins = np.linspace(min(df['taxes_sqft']), max(df['taxes_sqft']), 6)  # 20 bins w
```

In [15]:
```python
bins
```

Out[15]:
```
array([0.158802  , 0.88424499, 1.60968798, 2.33513097, 3.06057396,
       3.78601695])
```

In [16]:
```python
# Categorize data into bins
df['cat_pdCut_ls'] = pd.cut(df['taxes_sqft'], bins=bins, labels=labels, include_
```

In [17]:
```python
df['cat_pdCut_ls']
```

Out[17]:
```
0       3
1       3
2       3
3       4
4       2
       ..
3679    2
3680    3
3681    3
3682    3
3683    2
Name: cat_pdCut_ls, Length: 3684, dtype: category
Categories (5, object): ['1' < '2' < '3' < '4' < '5']
```

In [18]:
```
print(df)
```

```
           MLS  sold_price  zipcode   longitude   latitude  lot_acres  \
0     21329440   1125000.0    85718 -110.883547  32.329763       1.33
1     21500337   1100000.0    85750 -110.866891  32.321968       1.17
2     21206450   1180000.0    85750 -110.868487  32.316324       1.30
3     21224755   1175500.0    85718 -110.940650  32.347873       1.23
4     21703603   1125478.0    85755 -110.973498  32.460529       1.71
...        ...         ...      ...         ...        ...        ...
3679   3056450    525000.0    85614 -110.980945  31.824287       3.01
3680  21908358    565000.0    85750 -110.820216  32.307646       0.83
3681  21909379    535000.0    85718 -110.922291  32.317496       0.18
3682  21908591    550000.0    85750 -110.858556  32.316373       1.42
3683  21900515    550000.0    85745 -111.055528  32.296871       1.01

         taxes  year_built  bedrooms  bathrooms  sqrt_ft  garage  \
0      8654.00        1986         4        5.0   5023.0     3.0
1      6565.93        1994         4        4.0   3870.0     3.0
2      9590.16        1993         4        3.0   5029.0     3.0
3     11674.00        2004         4        5.0   4143.0     3.0
4      3171.39        2017         3        4.0   3436.0     3.0
...        ...         ...       ...        ...      ...     ...
3679   5122.84        2007         3        3.0   3512.0     3.0
3680   4568.71        1986         4        3.0   2813.0     2.0
3681   4414.00        2002         3        2.0   2106.0     2.0
3682   4822.01        1990         4        3.0   2318.0     3.0
3683   5822.93        2009         4        4.0   3724.0     3.0

      kitchen_features  fireplaces  floor_covering    HOA  taxes_sqft  \
0                    6         3.0               3  179.0    1.722875
1                    5         2.0               2   58.0    1.696623
2                    5         3.0               2   40.0    1.906972
3                    5         1.0               2  159.0    2.817765
4                    2         1.0               2   56.0    0.922989
...                ...         ...             ...    ...         ...
3679                 8         1.0               2   37.0    1.458667
3680                10         2.0               2    6.0    1.624141
3681                10         1.0               1  198.0    2.095916
3682                10         1.0               2   43.0    2.080246
3683                 9         1.0               2   56.0    1.563622

     cat_qcut cat_pdCut cat_pdCut_ls
0           3         3            3
1           3         3            3
2           4         3            3
3           5         4            4
4           1         2            2
...       ...       ...          ...
3679        2         2            2
3680        2         3            3
3681        4         3            3
3682        4         3            3
3683        2         2            2

[3684 rows x 20 columns]
```

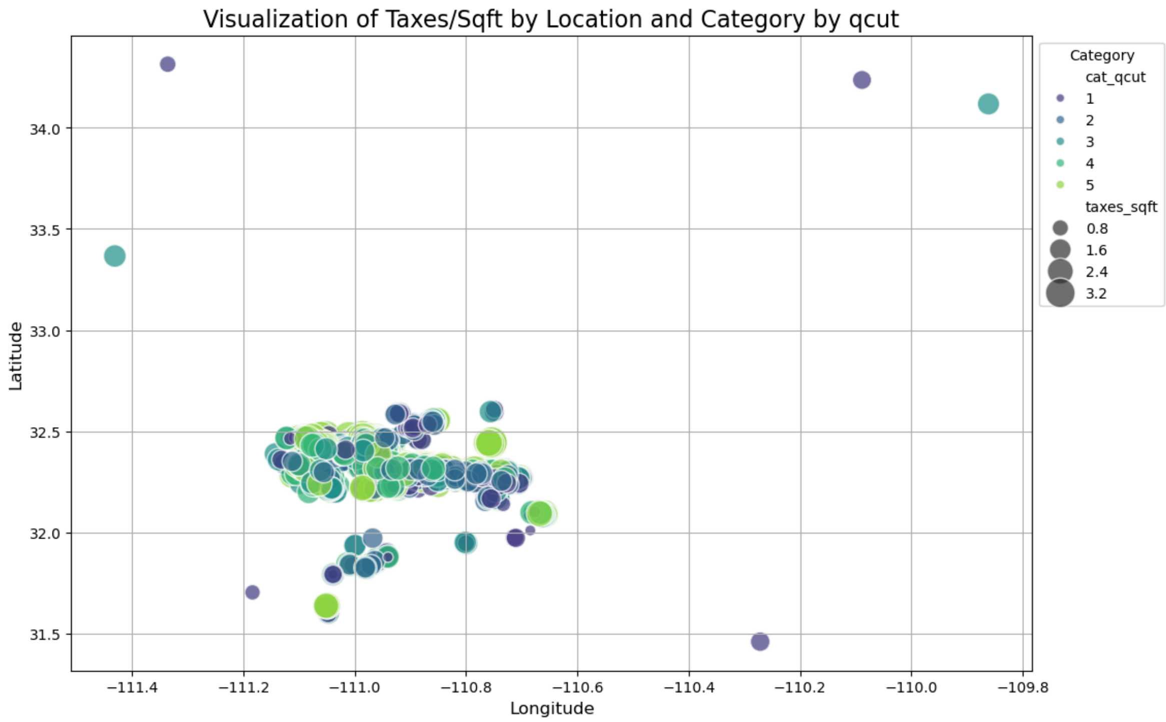
In [19]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3684 entries, 0 to 3683
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype   
---  ------            --------------  -----   
 0   MLS               3684 non-null   int64   
 1   sold_price        3684 non-null   float64 
 2   zipcode           3684 non-null   int64   
 3   longitude         3684 non-null   float64 
 4   latitude          3684 non-null   float64 
 5   lot_acres         3684 non-null   float64 
 6   taxes             3684 non-null   float64 
 7   year_built        3684 non-null   int64   
 8   bedrooms          3684 non-null   int64   
 9   bathrooms         3684 non-null   float64 
 10  sqrt_ft           3684 non-null   float64 
 11  garage            3684 non-null   float64 
 12  kitchen_features  3684 non-null   int64   
 13  fireplaces        3684 non-null   float64 
 14  floor_covering    3684 non-null   int64   
 15  HOA               3684 non-null   float64 
 16  taxes_sqft        3684 non-null   float64 
 17  cat_qcut          3684 non-null   category
 18  cat_pdCut         3684 non-null   category
 19  cat_pdCut_ls      3684 non-null   category
dtypes: category(3), float64(11), int64(6)
memory usage: 500.8 KB
```

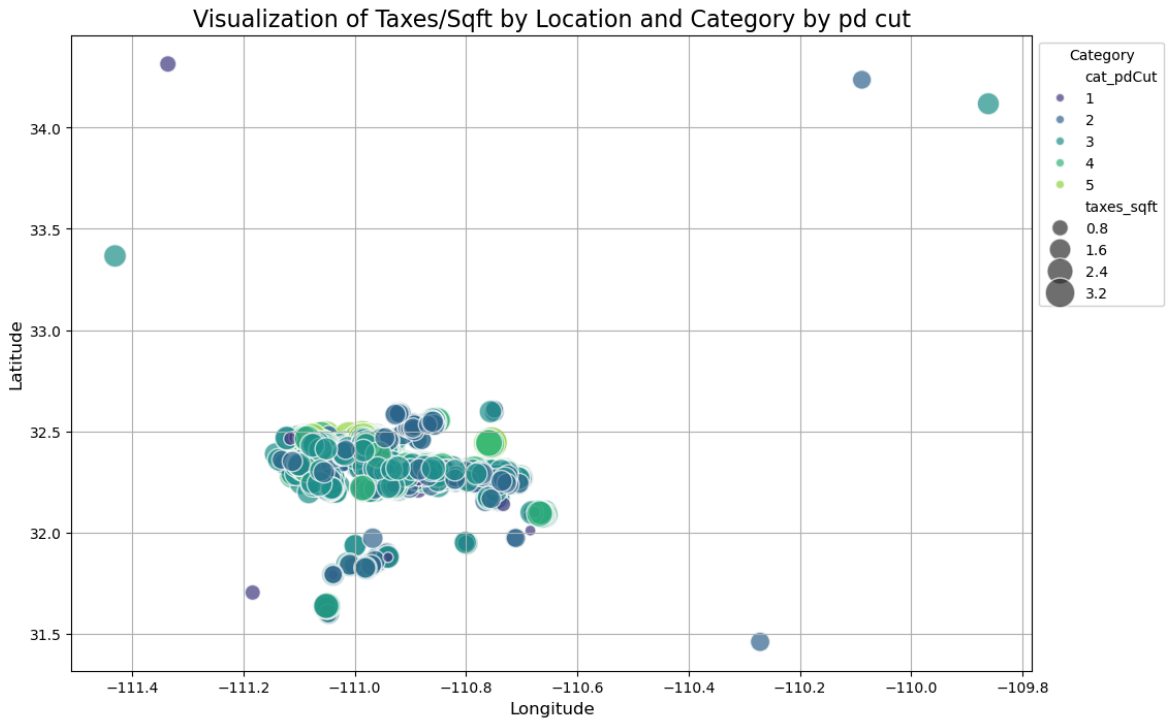
Data Visualization

In [25]:

```
plt.figure(figsize=(12, 8))
sns.scatterplot(
    data=df,
    x='longitude',
    y='latitude',
    size='taxes_sqft',
    hue='cat_qcut',
    palette='viridis',
    sizes=(50, 500),
    alpha=0.7
)
plt.title('Visualization of Taxes/Sqft by Location and Category by qcut', fontsi
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)
plt.legend(title='Category', loc='upper left', bbox_to_anchor=(1, 1))
plt.grid(True)
plt.show()
```

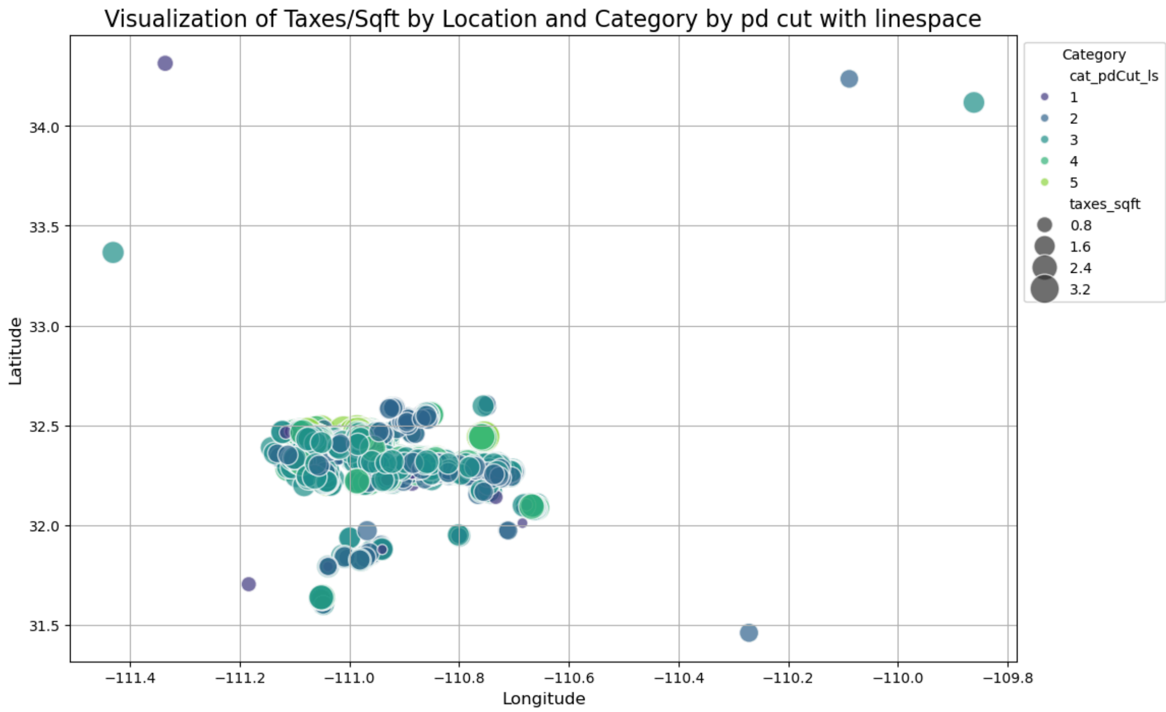


In [26]:
```
plt.figure(figsize=(12, 8))
sns.scatterplot(
    data=df,
    x='longitude',
    y='latitude',
    size='taxes_sqft',
    hue='cat_pdCut',
    palette='viridis',
    sizes=(50, 500),
    alpha=0.7
)
plt.title('Visualization of Taxes/Sqft by Location and Category by pd cut', font
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)
plt.legend(title='Category', loc='upper left', bbox_to_anchor=(1, 1))
plt.grid(True)
plt.show()
```



In [28]:

```
plt.figure(figsize=(12, 8))
sns.scatterplot(
    data=df,
    x='longitude',
    y='latitude',
    size='taxes_sqft',
    hue='cat_pdCut_ls',
    palette='viridis',
    sizes=(50, 500),
    alpha=0.7
)
plt.title('Visualization of Taxes/Sqft by Location and Category by pd cut with l
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)
plt.legend(title='Category', loc='upper left', bbox_to_anchor=(1, 1))
plt.grid(True)
plt.show()
```

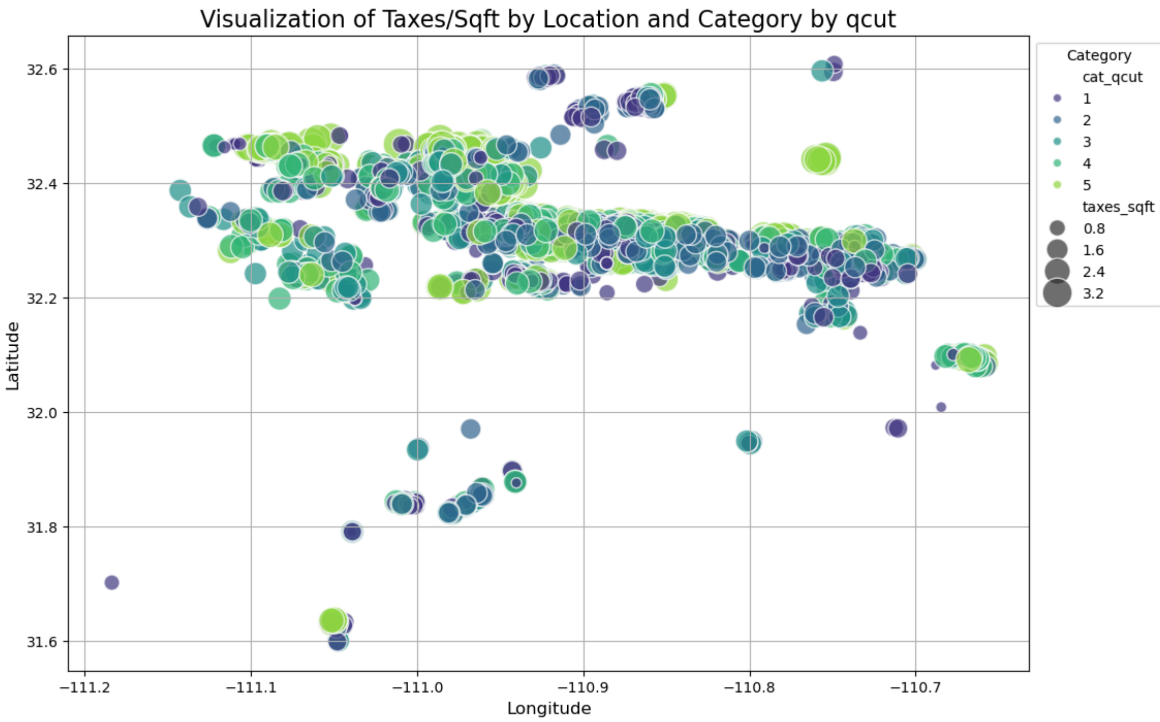


In [29]:

```
lat_min, lat_max = 31.50, 33.0
long_min, long_max = -111.20, -110.6

df_filtered = df[
    (df['latitude'] >= lat_min) & (df['latitude'] <= lat_max) &
    (df['longitude'] >= long_min) & (df['longitude'] <= long_max)
]
```

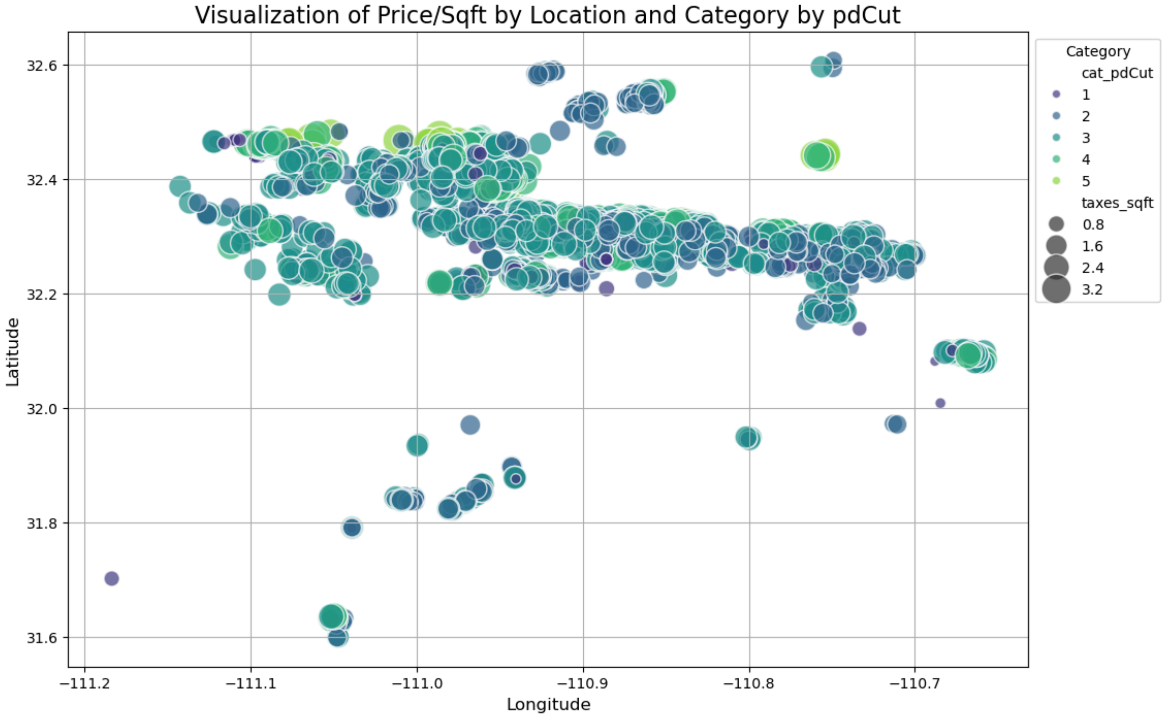
In [30]:

```
plt.figure(figsize=(12, 8))
sns.scatterplot(
    data=df_filtered,
    x='longitude',
    y='latitude',
    size='taxes_sqft',
    hue='cat_qcut',
    palette='viridis',
    sizes=(50, 500),
    alpha=0.7
)
plt.title('Visualization of Taxes/Sqft by Location and Category by qcut', fontsi
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)
plt.legend(title='Category', loc='upper left', bbox_to_anchor=(1, 1))
plt.grid(True)
plt.show()
```



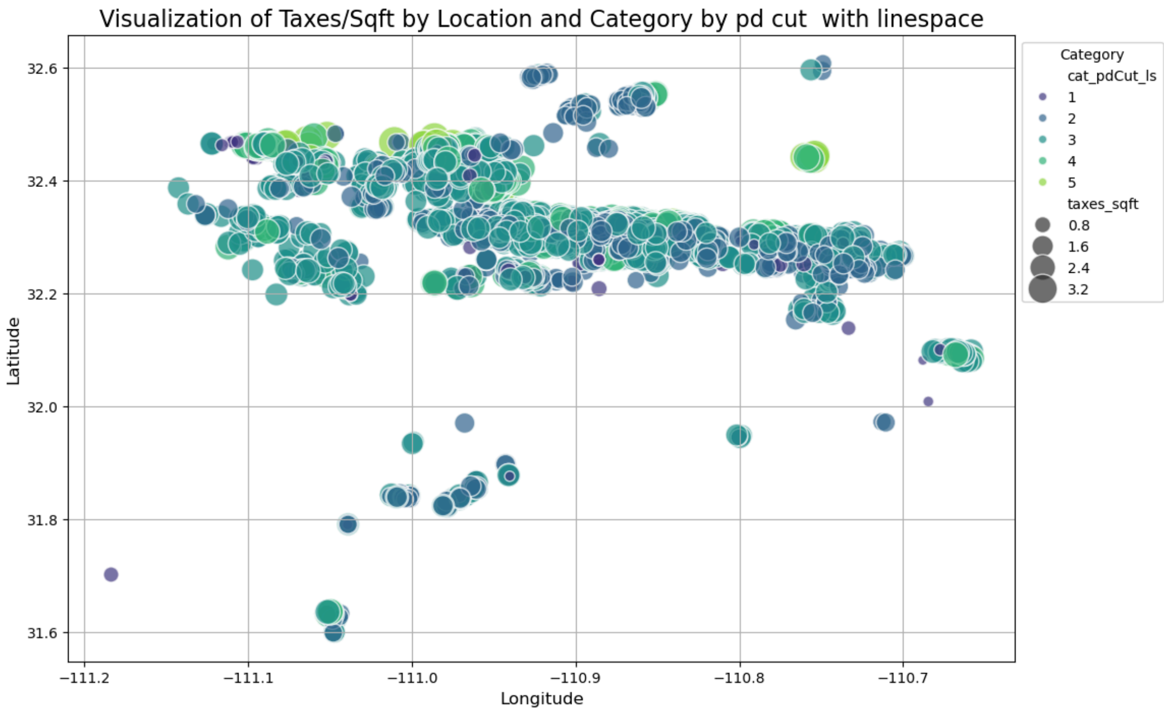
In [131…

```python
# Scatter plot: Price per sqft vs Latitude/Longitude
plt.figure(figsize=(12, 8))
sns.scatterplot(
    data=df_filtered,
    x='longitude',
    y='latitude',
    size='taxes_sqft',
    hue='cat_pdCut',
    palette='viridis',
    sizes=(50, 500),
    alpha=0.7
)
plt.title('Visualization of Price/Sqft by Location and Category by pdCut', fonts
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)
plt.legend(title='Category', loc='upper left', bbox_to_anchor=(1, 1))
plt.grid(True)
plt.show()
```



In [118…

```
# Scatter plot: Price per sqft vs Latitude/Longitude
plt.figure(figsize=(12, 8))
sns.scatterplot(
    data=df_filtered,
    x='longitude',
    y='latitude',
    size='taxes_sqft',
    hue='cat_pdCut_ls',
    palette='viridis',
    sizes=(50, 500),
    alpha=0.7
)
plt.title('Visualization of Taxes/Sqft by Location and Category by pd cut  with
plt.xlabel('Longitude', fontsize=12)
plt.ylabel('Latitude', fontsize=12)
plt.legend(title='Category', loc='upper left', bbox_to_anchor=(1, 1))
plt.grid(True)
plt.show()
```



KNN Regression Model

In [32]:
```
class KNNClassifier():
    def fit(self, X, y):# lazy learner just perform operation
        self.X = X
        self.y = y

    def predict(self, X, K , epsilon=1e-1):
        N = len(X) #number of observations
        y_hat = np.zeros(N) #

        for i in range(N):
            dist2 = np.sum((self.X - X[i])**2,axis =1)
            idxt = np.argsort(dist2)[:K] # give a list of index from lowest dist
            gamma_k = 1/(np.sqrt(dist2[idxt] + epsilon))

            y_hat[i] =np.bincount(self.y[idxt],weights=gamma_k).argmax() # to gi

        return y_hat # return outside for loop
```

In [33]:
```
def accuracy(y, y_hat):
    return np.mean(y==y_hat)
```

In [34]:
```
knn_instance = KNNClassifier()
```

In [35]:
```
df
```

Out[35]:

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_b
0	21329440	1125000.0	85718	-110.883547	32.329763	1.33	8654.00	1
1	21500337	1100000.0	85750	-110.866891	32.321968	1.17	6565.93	1
2	21206450	1180000.0	85750	-110.868487	32.316324	1.30	9590.16	1
3	21224755	1175500.0	85718	-110.940650	32.347873	1.23	11674.00	2
4	21703603	1125478.0	85755	-110.973498	32.460529	1.71	3171.39	2
...	...	...	...	...	...	...	...	
3679	3056450	525000.0	85614	-110.980945	31.824287	3.01	5122.84	2
3680	21908358	565000.0	85750	-110.820216	32.307646	0.83	4568.71	1
3681	21909379	535000.0	85718	-110.922291	32.317496	0.18	4414.00	2
3682	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1
3683	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2

3684 rows × 20 columns

In [36]:

```
Cols = ["longitude","latitude"]
X_f = df_filtered[Cols].values

y_f=df_filtered["cat_qcut"].astype(int).values


knn_instance.fit(X_f,y_f)
y_hat_f = knn_instance.predict(X_f,K=3)

accuracy(y_f,y_hat_f)
```

Out[36]: np.float64(0.801032889372112)

In [37]:

```
Cols = ["longitude","latitude"]
X = df[Cols].values
X

y=df["cat_qcut"].astype(int).values
y

knn_instance.fit(X,y)
y_hat = knn_instance.predict(X,K=3)
```

In [38]:

```
accuracy(y,y_hat)
```

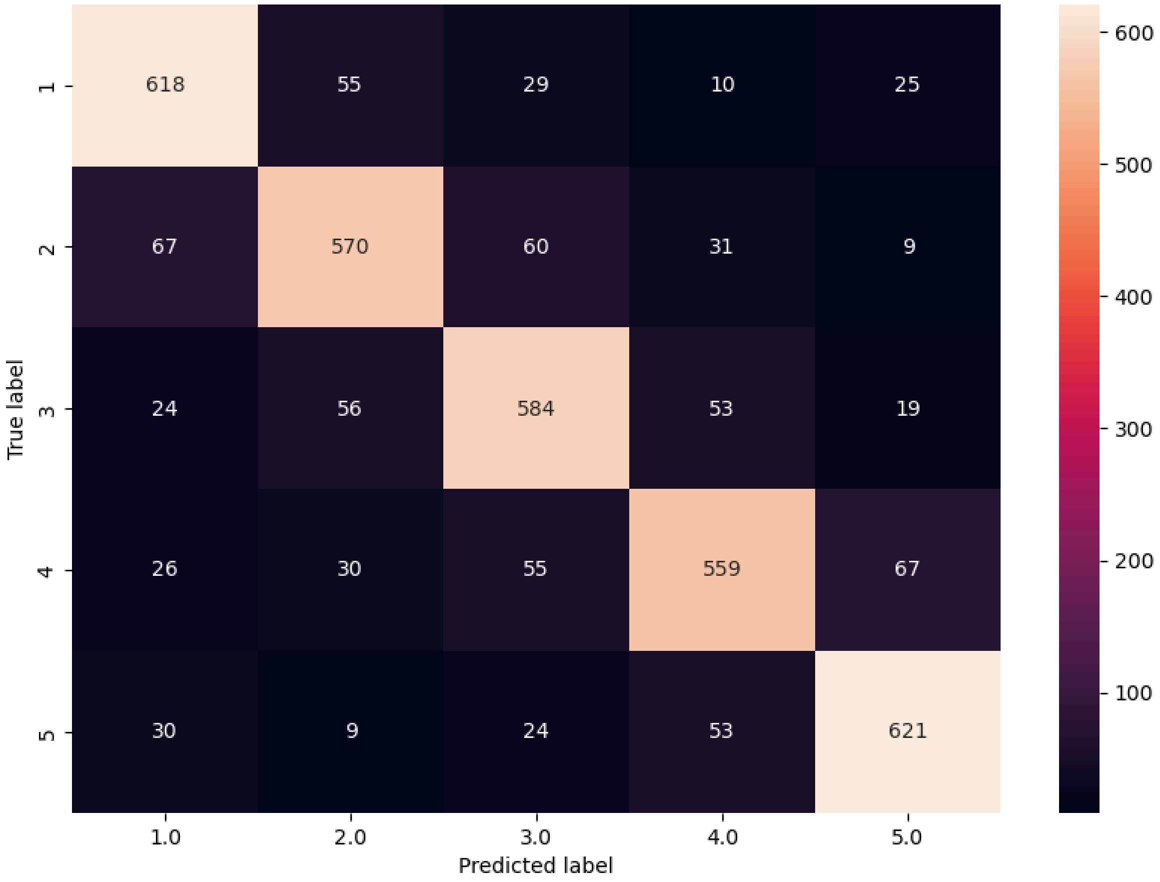
Out[38]: np.float64(0.8013029315960912)

In [39]:

```
plt.figure(figsize=(10,7))
ytest_actu = pd.Series(y, name='Actual')
ytest_pred = pd.Series(y_hat, name='Predicted')
cm = pd.crosstab(ytest_actu, ytest_pred)
ax = sns.heatmap(cm, annot=True, fmt="d")
```

```
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

Out[39]: Text(0.5, 47.7222222222222, 'Predicted label')



In [40]:
```
Cols = ["longitude","latitude"]
X_f1 = df_filtered[Cols].values

y_f1=df_filtered["cat_pdCut"].astype(int).values


knn_instance.fit(X_f1,y_f1)
y_hat_f1 = knn_instance.predict(X_f1,K=3)

accuracy(y_f1,y_hat_f1)
```

Out[40]: np.float64(0.806469149225333)

In [41]:
```
y1=df["cat_pdCut"].astype(int).values
y1

knn_instance.fit(X,y1)
y_hat1 = knn_instance.predict(X,K=3)
```

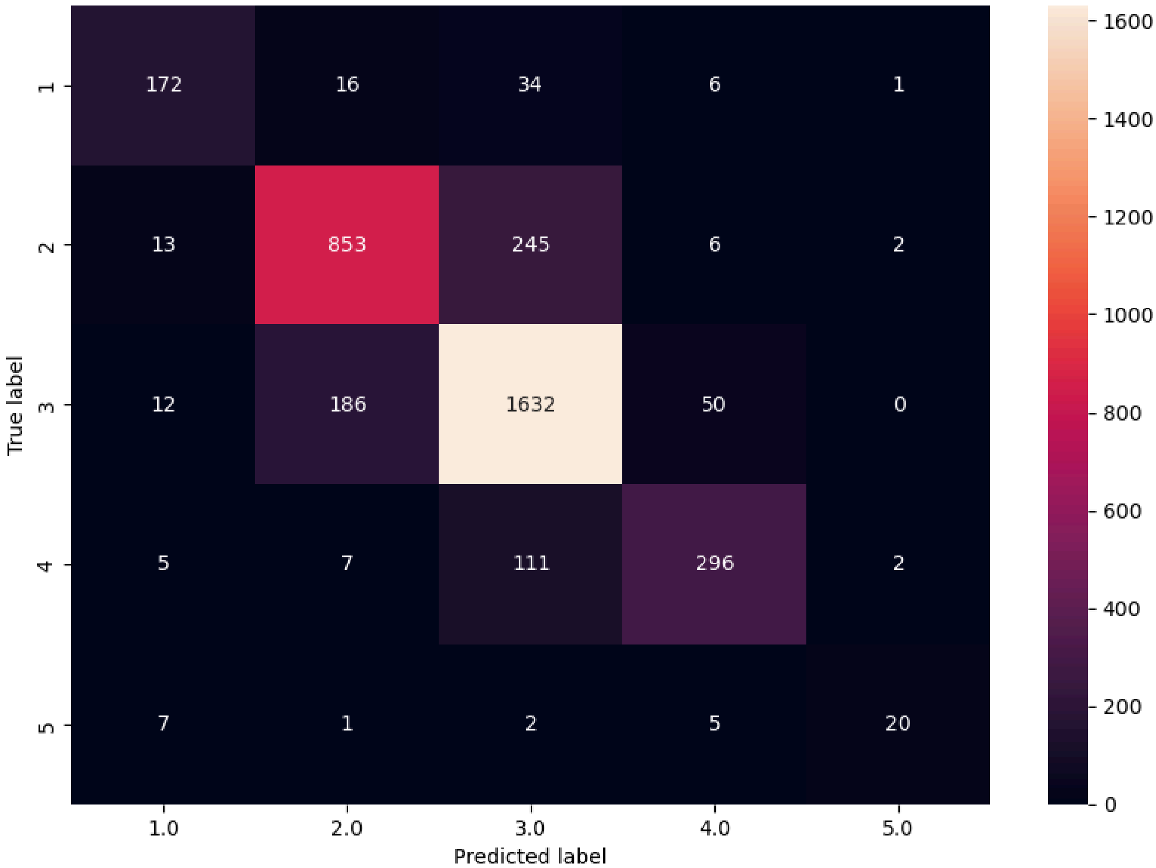
In [42]:
```
accuracy(y1,y_hat1)
```

Out[42]: np.float64(0.8070032573289903)

In [44]:
```
plt.figure(figsize=(10,7))
ytest_actu1 = pd.Series(y1, name='Actual')
ytest_pred1 = pd.Series(y_hat1, name='Predicted')
cm = pd.crosstab(ytest_actu1, ytest_pred1)
```

```
ax = sns.heatmap(cm, annot=True, fmt="d")
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

Out[44]: Text(0.5, 47.7222222222222, 'Predicted label')



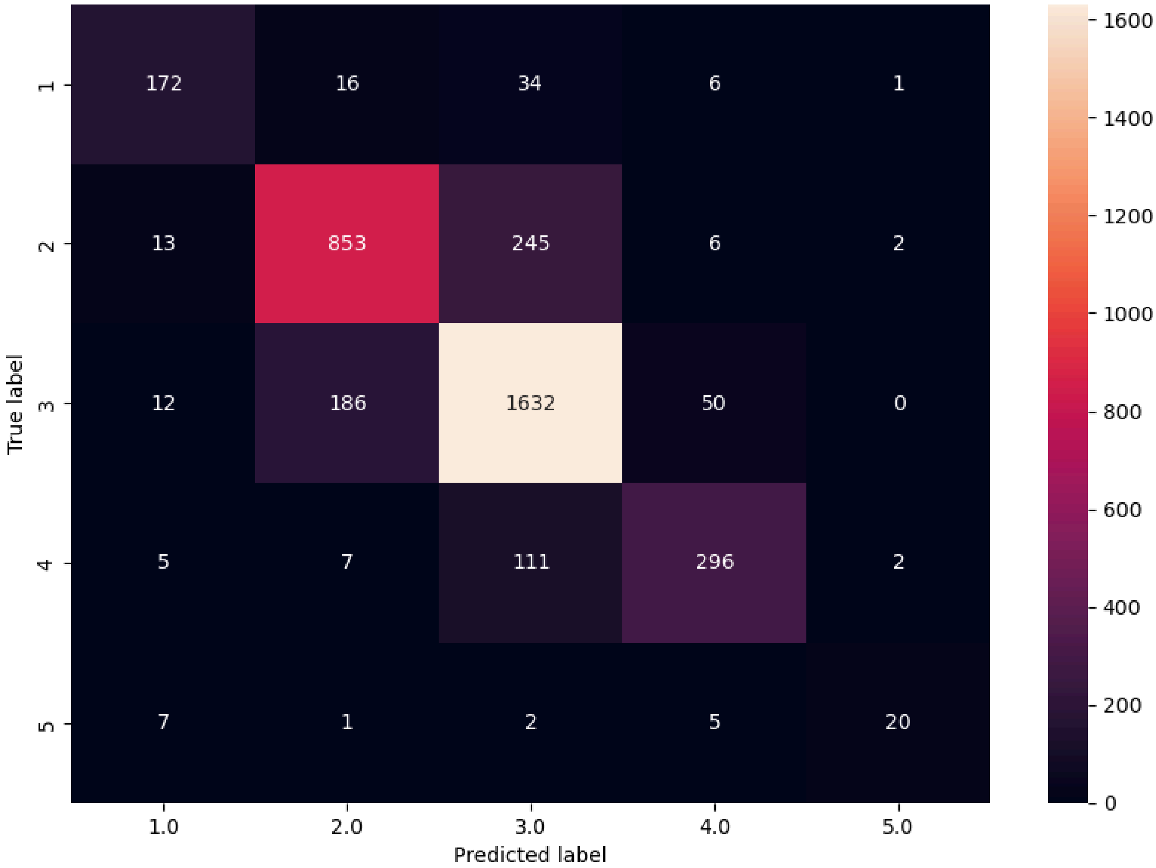
In [45]:
```
y2=df["cat_pdCut_ls"].astype(int).values
y2

knn_instance.fit(X,y2)
y_hat2 = knn_instance.predict(X,K=3)
```

In [46]:
```
accuracy(y2,y_hat2)
```

Out[46]: np.float64(0.8070032573289903)

In [47]:
```
plt.figure(figsize=(10,7))
ytest_actu2 = pd.Series(y2, name='Actual')
ytest_pred2 = pd.Series(y_hat2, name='Predicted')
cm = pd.crosstab(ytest_actu2, ytest_pred2)
ax = sns.heatmap(cm, annot=True, fmt="d")
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

Out[47]: Text(0.5, 47.7222222222222, 'Predicted label')



Simple Linear Regression

In [121…

```
class SimpleLinearReg():
  def fit(self,X,y):
    self.y=y
    self.denominator = np.mean(X**2)- np.mean(X)**2
    self.w1= (np.mean(X*y)-(np.mean(X)*np.mean(y))) / self.denominator
    self.w0 = (np.mean(y)*np.mean(X**2) - np.mean(X)*np.mean(X*y))/self.denomina

  def predict(self, X, show=0):
    y_hat = self.w1 * X  + self.w0

    if show:
      plt.figure()
      plt.scatter(X, self.y, s=8)
      plt.plot(X, y_hat,color="#FF0070")
      plt.xlabel('Taxes per Square Foot')
      plt.ylabel('Price Zone')
      plt.title('Scatter Plot with Linear Regression')
      plt.legend()

    return y_hat
```

Model Fit

In [122…

```
df['price_zone'] = df['sold_price'] / df['zipcode']
```

In [123…

```
df
```

Out[123...

	MLS	sold_price	zipcode	longitude	latitude	lot_acres	taxes	year_b
0	21329440	1125000.0	85718	-110.883547	32.329763	1.33	8654.00	1
1	21500337	1100000.0	85750	-110.866891	32.321968	1.17	6565.93	1
2	21206450	1180000.0	85750	-110.868487	32.316324	1.30	9590.16	1
3	21224755	1175500.0	85718	-110.940650	32.347873	1.23	11674.00	2
4	21703603	1125478.0	85755	-110.973498	32.460529	1.71	3171.39	2
...	...	...	...	...	...	...	...	
3679	3056450	525000.0	85614	-110.980945	31.824287	3.01	5122.84	2
3680	21908358	565000.0	85750	-110.820216	32.307646	0.83	4568.71	1
3681	21909379	535000.0	85718	-110.922291	32.317496	0.18	4414.00	2
3682	21908591	550000.0	85750	-110.858556	32.316373	1.42	4822.01	1
3683	21900515	550000.0	85745	-111.055528	32.296871	1.01	5822.93	2

3684 rows × 21 columns

In [124...

```
#Cols = ["longitude","latitude"]
X = df['taxes_sqft'].values
X

y=df["price_zone"].values
y
```

Out[124...

```
array([13.12443127, 12.82798834, 13.76093294, ...,  6.24139621,
        6.41399417,  6.41436818], shape=(3684,))
```

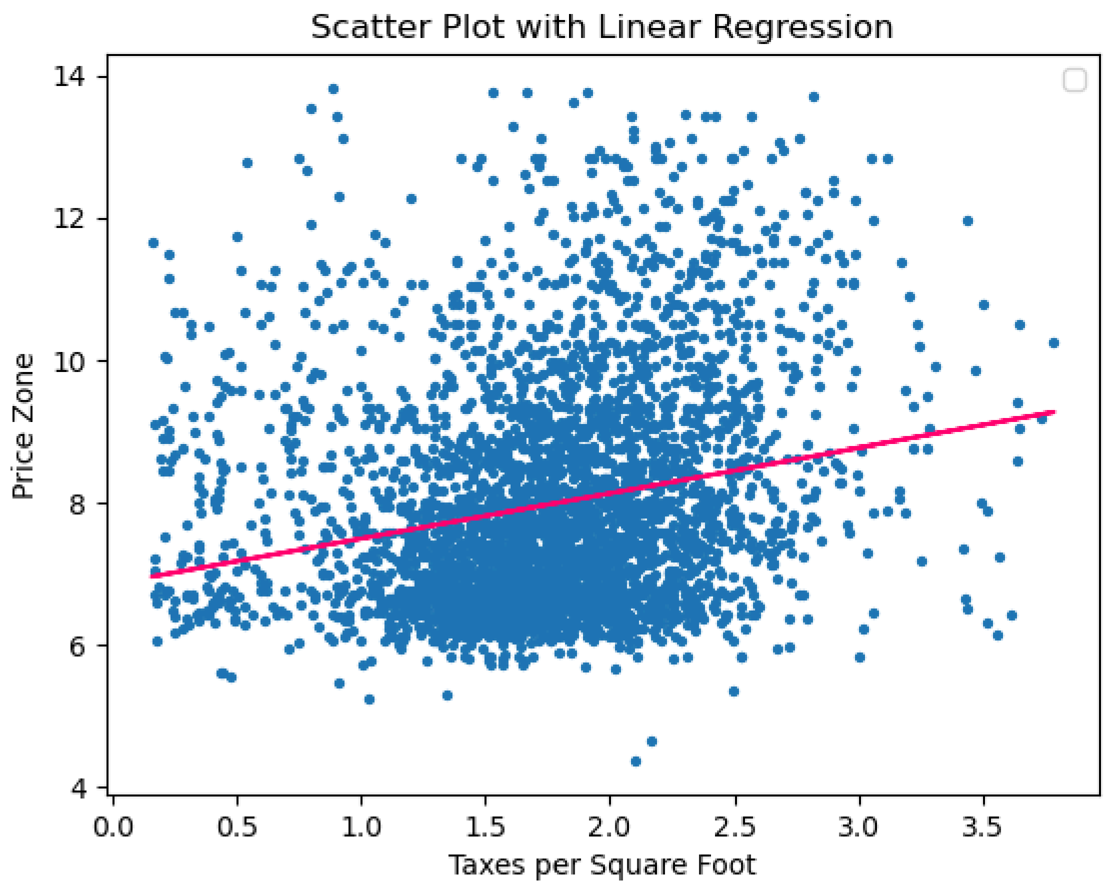
In [125...

```
slr = SimpleLinearReg()
slr.fit(X,y)
```

In [126...

```
y_hat = slr.predict(X,show=1)
```

```
C:\Users\Vaishali\AppData\Local\Temp\ipykernel_2816\3840397035.py:18: UserWarnin
g: No artists with labels found to put in legend.  Note that artists whose label
start with an underscore are ignored when legend() is called with no argument.
  plt.legend()
```



Ordinary Least Square (OLS) Metric

In [75]:
```
def OLS(Y,Y_hat):
  N = Y_hat.shape[0]
  return ((1/(2*N))*np.sum((Y-Y_hat)**2))
```

In [76]:
```
OLS(y,y_hat)
```

Out[76]:
```
np.float64(1.20207561022373)
```

Known Function

$y = w1 \ * F(X) + w0$

In [77]:
```
y2=y**2
```

In [78]:
```
slr_known = SimpleLinearReg()
slr_known.fit(X,y2)
```

In [79]:
```
y_hat2= slr_known.predict(X)
```

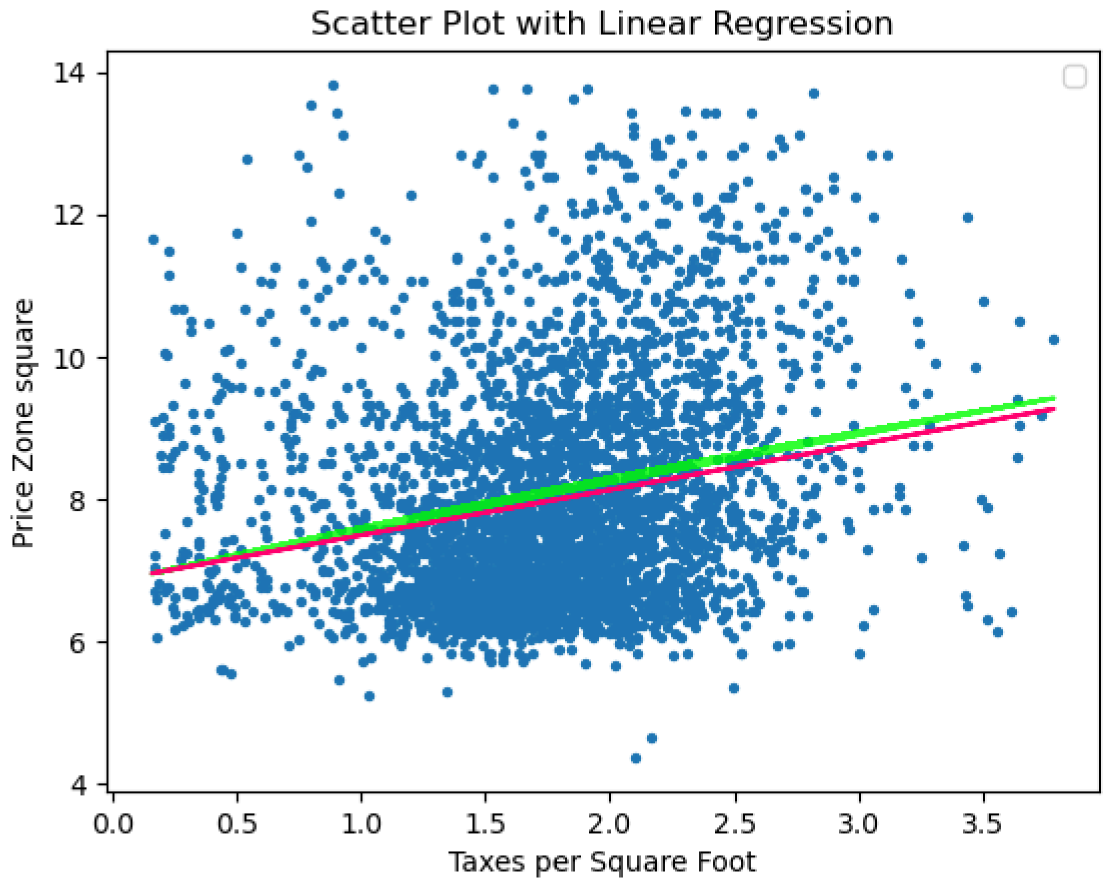
In [127…
```
plt.figure()
plt.scatter(X,y,s=8)
plt.plot(X,np.sqrt(y_hat2),color="#00FF00", alpha=0.8)
plt.plot(X,y_hat,color="#FF0070")
plt.xlabel('Taxes per Square Foot')
plt.ylabel('Price Zone square')
```

```python
plt.title('Scatter Plot with Linear Regression')
plt.legend()
```

```
C:\Users\Vaishali\AppData\Local\Temp\ipykernel_2816\1900821809.py:8: UserWarning: No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored when legend() is called with no argument.
  plt.legend()
```

Out[127…

```
<matplotlib.legend.Legend at 0x23cf46785c0>
```



In [81]:

```python
OLS(y,np.sqrt(y_hat2))
```

Out[81]:

```
np.float64(1.2176200462033464)
```

KNN Regressor Class

In [86]:

```python
class KNNRegressor():
  def fit(self, X, y):
    self.X = X
    self.y = y

  def predict(self, X,K, epsilon = 1e-3):
    N = len(X)
    y_hat = np.zeros(N)

    for i in range(N):
      dist2 = np.sum((self.X-X[i])**2, axis =1)
      idxt = np.argsort(dist2)[:K]
      gamma_k = np.exp(-dist2[idxt])/ (np.exp(-dist2[idxt]).sum()+epsilon)
      y_hat[i] = gamma_k.dot(self.y[idxt])
```

```python
    return y_hat
```

Prediction using KNN Regressor

In [84]:
```python
dat_training = ContValData()
X_train,y_train = dat_training.create(1,1000,noise=0.2)
dat_training.show()
```

Out[84]:
```
array([13.12443127, 12.82798834, 13.76093294, ...,  6.24139621,
        6.41399417,  6.41436818], shape=(3684,))
```

In [ ]:
```python
y_hat_fast = fast.predict(X)
```

OLS Regressor with Gradient Descent

In [185…
```python
def MAE(Y,Y_hat):
  return np.sum(np.abs((Y-Y_hat)))/len(Y)

def R2(Y,Y_hat):
  N=len(Y)
  return 1-((np.sum((Y-Y_hat)**2)/np.sum((Y_hat-np.mean(Y))**2)))

def OLS(Y, Y_hat,N):
   return ((1/(2*N))*np.sum((Y-Y_hat)**2))
```

OLS Multivariante Linear Regression Class

In [186…
```python
class MVLinearRegression():

  def fit(self, X, y, eta=1e-3, epochs=1e3, show_curve=True):
    epochs= int(epochs)# eta or lr as learning rate
    N,D=X.shape
    Y=y

    #Begin Optimizationwith SGD
    self.W = np.random.randn(D)
    self.J= np.zeros(epochs)

    # Start Gradient Descent Progression
    for epoch in range(epochs):
      Y_hat = self.predict(X)
      self.J[epoch] = OLS(Y,Y_hat,N)

      # Weight Update Rule
      self.W -= eta * (1/N) * (X.T@(Y_hat-Y))

    if show_curve:
       plt.figure()
       plt.plot(self.J)
       plt.xlabel("epoches")
       plt.ylabel("J")
       plt.title("Training Curve")
```

```python
  def predict(self,X):
    return X@self.W
```

In [187…

```python
df.columns
```

Out[187…

```
Index(['MLS', 'sold_price', 'zipcode', 'longitude', 'latitude', 'lot_acres',
       'taxes', 'year_built', 'bedrooms', 'bathrooms', 'sqrt_ft', 'garage',
       'kitchen_features', 'fireplaces', 'floor_covering', 'HOA', 'taxes_sqft',
       'cat_qcut', 'cat_pdCut', 'cat_pdCut_ls', 'price_zone'],
      dtype='object')
```

In [188…

```python
X=df[['taxes_sqft','price_zone','bedrooms','bathrooms','lot_acres','year_built']
```

In [189…

```python
y=X[:,0]
X=X[:,1:]
```

In [190…

```python
X_scaled = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
```

In [191…

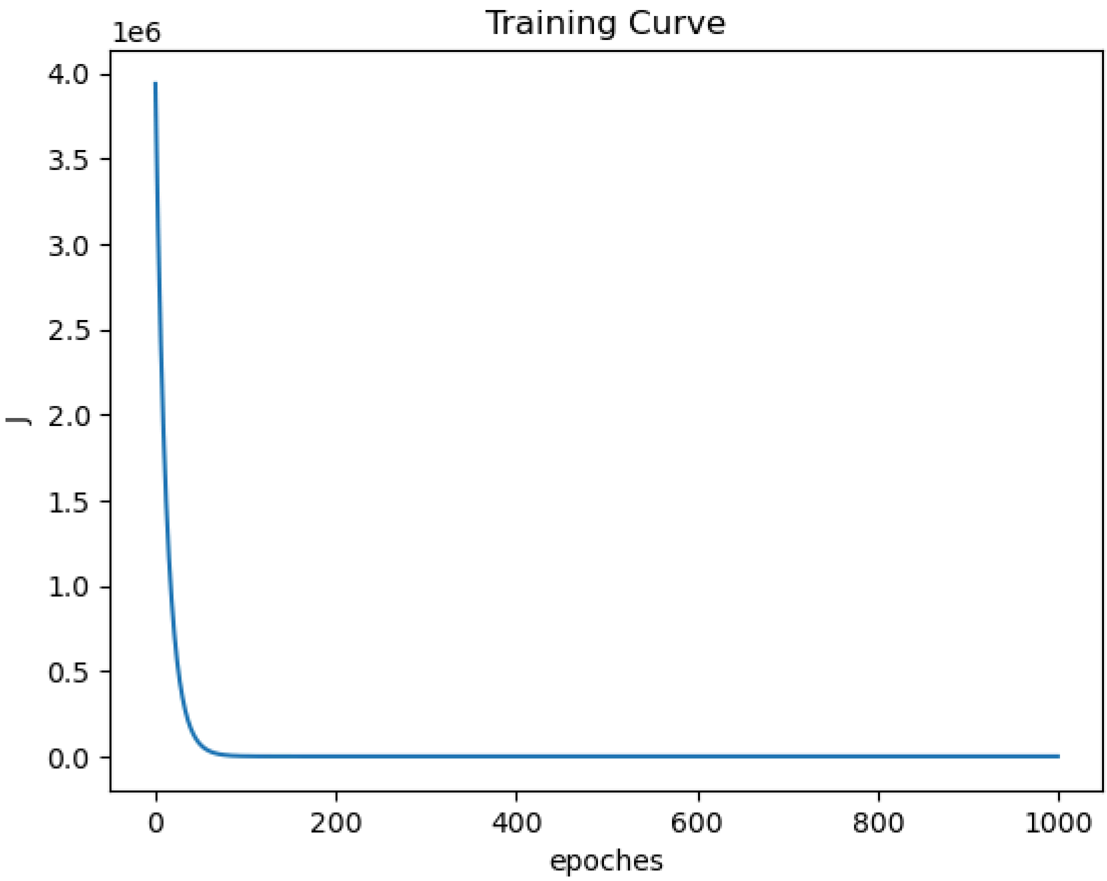
```python
y
```

Out[191…

```
array([1.72287478, 1.69662274, 1.90697156, ..., 2.09591643, 2.0802459 ,
       1.56362245], shape=(3684,))
```

In [196…

```python
my_reg = MVLinearRegression()
my_reg.fit(X,y,eta=1e-8,epochs=1e3)
```



Inference Function

In [220…

```python
def predict_house(X_test,model):
  y_Out = model.predict(X_test)
```

```
    print("The Tax of the home is predicted to be ",round(y_Out[0],2))
    return y_Out
```

In [198…

```
col = ["taxes_sqft","price_zone","bedrooms","bathrooms","lot_acres", "year_built
```

In [199…

```
df.columns
```

Out[199…

```
Index(['MLS', 'sold_price', 'zipcode', 'longitude', 'latitude', 'lot_acres',
       'taxes', 'year_built', 'bedrooms', 'bathrooms', 'sqrt_ft', 'garage',
       'kitchen_features', 'fireplaces', 'floor_covering', 'HOA', 'taxes_sqft',
       'cat_qcut', 'cat_pdCut', 'cat_pdCut_ls', 'price_zone'],
      dtype='object')
```

In [200…

```
df[col].head()
```

Out[200…

	taxes_sqft	price_zone	bedrooms	bathrooms	lot_acres	year_built
0	1.722875	13.124431	4	5.0	1.33	1986
1	1.696623	12.827988	4	4.0	1.17	1994
2	1.906972	13.760933	4	3.0	1.30	1993
3	2.817765	13.713572	4	5.0	1.23	2004
4	0.922989	13.124343	3	4.0	1.71	2017

In [221…

```
#'taxes_sqft','price_zone','bedrooms','bathrooms','lot_acres','year_built'
X_test = np.array([[13.124431,4,5.0,1.33,1986]])
```

In [222…

```
float(my_reg.predict(X_test)[0])
```

Out[222…

```
2.571645594320823
```

In [223…

```
predict_house(X_test,my_reg)
```

```
The Tax of the home is predicted to be  2.57
```

Out[223…

```
array([2.57164559])
```

In [224…

```
PredictedTax = t_sq * 5023.0
```

In [225…

```
PredictedTax
```

Out[225…

```
array([57868799.67615443])
```

In [226…

```
X_test1 = np.array([[13.124431,4,5.0,1.33,1986]])
X_test2 = np.array([[12.827988,4,4.0,1.17,1994]])
X_test3 = np.array([[13.760933,4,3.0,1.30,1993]])
X_test4 = np.array([[13.713572,4,5.0,1.23,2004]])
X_test5 = np.array([[13.124343,3,4.0,1.71,2017]])
```

In [227…

```
float(my_reg.predict(X_test1)[0])
```

Out[227…

```
2.571645594320823
```

In [228…

```
predict_house(X_test1,my_reg)
```

```
The Tax of the home is predicted to be  2.57
```

Out[228… array([2.57164559])

In [229…
```
float(my_reg.predict(X_test2)[0])
predict_house(X_test2,my_reg)
```

The Tax of the home is predicted to be 1.93

Out[229… array([1.92747161])

In [230…
```
float(my_reg.predict(X_test3)[0])
predict_house(X_test3,my_reg)
```

The Tax of the home is predicted to be 1.31

Out[230… array([1.31464421])

In [231…
```
float(my_reg.predict(X_test4)[0])
predict_house(X_test4,my_reg)
```

The Tax of the home is predicted to be 2.53

Out[231… array([2.52677025])

In [232…
```
float(my_reg.predict(X_test5)[0])
predict_house(X_test5,my_reg)
```

The Tax of the home is predicted to be 1.83

Out[232… array([1.82934706])

In [ ]:
```
#[2.57164559,1.92747161,1.31464421,2.52677025,1.82934706]
```

In [ ]:
```
```