MAT 328 Project: Malique Russell

Structure and Shape of the Data

The dataset is structured in a rectangular (tabular) format, consisting of rows and columns. It includes a mix of quantitative and qualitative data.

Quantitative Data:

- Extremely Low Income Units: Units with rents at 0–30% of the area median income
- Very Low Income Units: Rents at 31–50% of the area median income
- Low Income Units: Rents at 51–80% of the area median income
- Moderate Income Units: Rents at 81–120% of the area median income
- Middle Income Units: Rents at 121–165% of the area median income
- Other Income Units: Units reserved for building superintendents
- Counted Rental Units: Units counted under the Housing New York plan where assistance was provided to landlords
- Counted Homeownership Units: Units counted under the Housing New York plan where assistance was provided directly to homeowners
- All Counted Units: Total affordable units counted under the Housing New York plan
- Total Units: Sum of all units in the dataset
- Senior Units: Units specifically designated for senior households

Qualitative Data:

- Project ID: Unique identifier for each project
- Project Name: Name assigned by the Housing Preservation and Development (HPD).
- Program Group: Type of housing initiative
- Project Start Date: Date of project loan or agreement closure
- Project Completion Date: Date of the last building completion in a project
- Extended Affordability Only: Indicates whether the project qualifies for extended affordability
- Prevailing Wage Status: Specifies if the project adheres to prevailing wage requirements (e.g., Davis-Bacon Act)
- Planned Tax Benefit: Expected tax incentives associated with the project

Granularity of the Data:

The dataset has a low level of granularity, as each row represents aggregated unit data rather than individual housing units. A more granular dataset would provide detailed information at the unit level rather than summaries by category

Scope and Completeness of the Data

The dataset is well-suited for analyzing affordable housing trends in New York City. However, its scope is too broad for hyper-localized questions (e.g., borough-specific trends) and too narrow for state-wide analysis

Temporality of the Data

The dataset spans eight years, covering January 1, 2014, to December 31, 2021. It is managed by the Department of Housing Preservation and Development (HPD) and was last updated on March 3, 2025

Faithfulness of the Data

The dataset appears highly reliable, as it is compiled by a reputable city agency with direct oversight and access to housing records, ensuring accuracy and completeness

In [3]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import statsmodels.formula.api as smf
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import statsmodels.formula.api as smf
import plotly.graph_objects as go
from scipy.special import expit
from scipy.stats import logistic
from sklearn.preprocessing import MinMaxScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import confusion_matrix
```

In [4]:

```
affordable_housing = pd.read_csv("Affordable_Housing_Production_by_Project.csv")
affordable_housing["Project Completion Date"] = pd.to_datetime(affordable_housing["
affordable_housing = affordable_housing.sort_values(by='Project Completion Date', a
affordable_housing.head()
```

Out[4]:

	Project ID	Project Name	Program Group	Project Start Date	Project Completion Date	Extended Affordability Only	Prevaili Wa Stat
549	55759	CONFIDENTIAL	CONFIDENTIAL	01/03/2014	2014-01-03	No	N Prevaili Wa
523	55647	CONFIDENTIAL	CONFIDENTIAL	01/07/2014	2014-01-07	No	N Prevaili Wa
555	55773	CONFIDENTIAL	CONFIDENTIAL	01/10/2014	2014-01-10	No	N Prevaili Wa
641	57341	CONFIDENTIAL	CONFIDENTIAL	01/10/2014	2014-01-10	No	N Prevaili Wa
533	55697	CONFIDENTIAL	CONFIDENTIAL	01/14/2014	2014-01-14	No	N Prevaili Wa

In [5]:

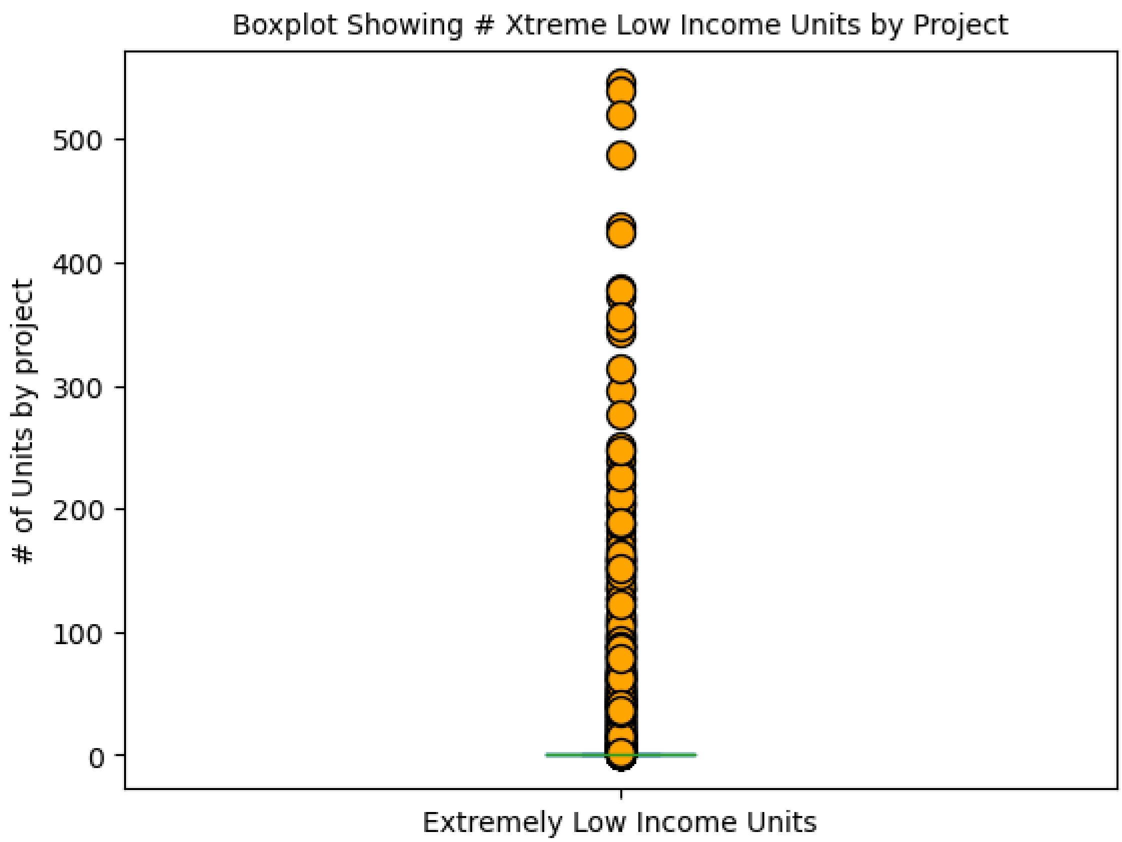
```
# Dropping Incomplete projects
complete_projects = affordable_housing.dropna(subset=['Project Completion Date'])
complete_projects.reset_index(drop=True, inplace=True)
complete_projects.head()
```

Out[5]:

	Project ID	Project Name	Program Group	Project Start Date	Project Completion Date	Extended Affordability Only	Prevailing Wage Status
0	55759	CONFIDENTIAL	CONFIDENTIAL	01/03/2014	2014-01-03	No	Non Prevailing Wage
1	55647	CONFIDENTIAL	CONFIDENTIAL	01/07/2014	2014-01-07	No	Non Prevailing Wage
2	55773	CONFIDENTIAL	CONFIDENTIAL	01/10/2014	2014-01-10	No	Non Prevailing Wage
3	57341	CONFIDENTIAL	CONFIDENTIAL	01/10/2014	2014-01-10	No	Non Prevailing Wage
4	55697	CONFIDENTIAL	CONFIDENTIAL	01/14/2014	2014-01-14	No	Non Prevailing Wage

In [6]:
```
xtreme = complete_projects["Extremely Low Income Units"]
very = complete_projects["Very Low Income Units"]
low = complete_projects["Low Income Units"]
moderate = complete_projects["Moderate Income Units"]
middle = complete_projects["Middle Income Units"]
other = complete_projects["Other Income Units"]
owned = complete_projects["Counted Homeownership Units"]
total = complete_projects["All Counted Units"]
```

In [7]:
```
xtreme.plot(kind = "box", flierprops=dict(marker='o', markersize=10, markerfacecolo
plt.ylabel("# of Units by project")
_=plt.title('Boxplot Showing # Xtreme Low Income Units by Project', fontsize = 10)
```



This graph shows the distribution of completed extremely low-income units by project built in New York City from january 1, 2014 to December 30, 2025

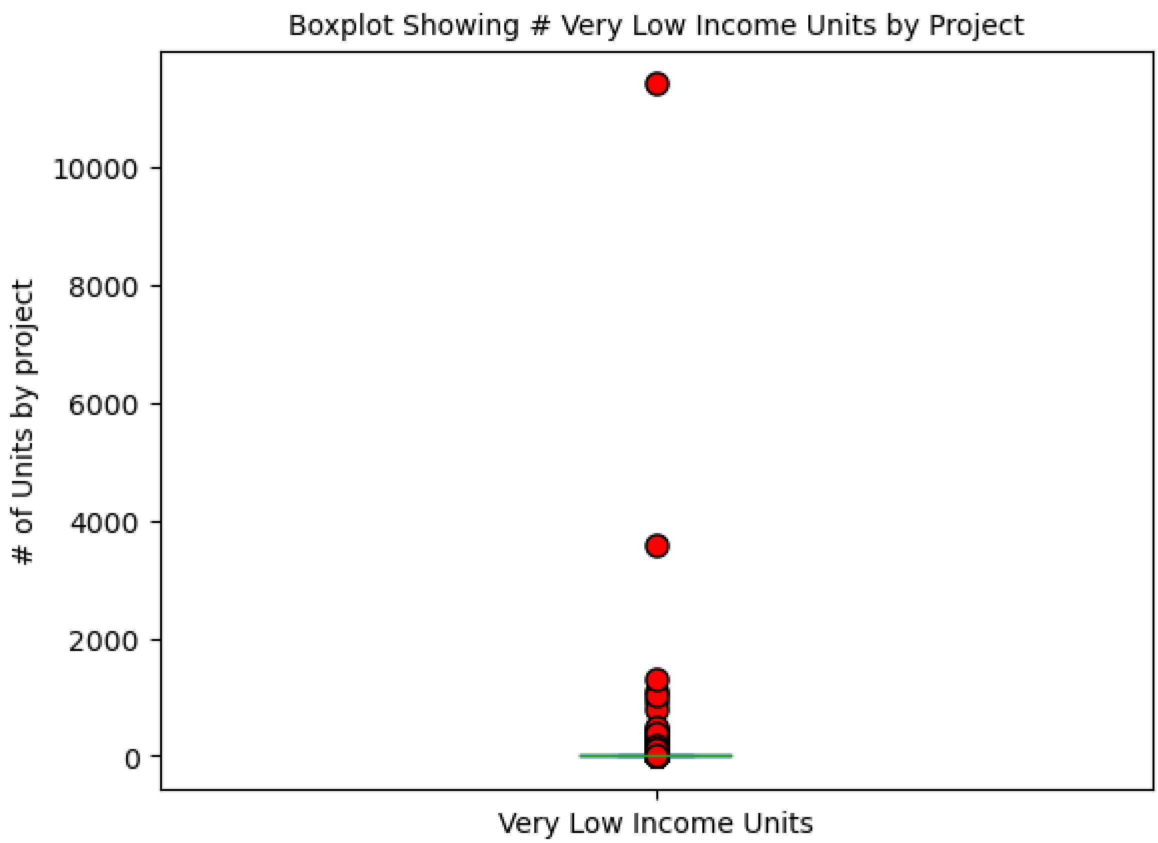
Some notable deductions:

- The majority of projects had under 300 extremely low-income unit
- Most projects had less than 250 of these units

In [9]:
```
# Percent of Completed Extremely Low Income Units
Xtreme =xtreme.sum()/total.sum()
xtreme_per = round(Xtreme*100,2)
xtreme_per
```

Out[9]: 16.84

In [10]:
```
very.plot(kind = "box", flierprops=dict(marker='o', markersize = 8, markerfacecolor
plt.ylabel("# of Units by project")
_=plt.title('Boxplot Showing # Very Low Income Units by Project', fontsize = 10)
```



This graph shows the distribution of completed very low-income units by project built in New York City from january 1, 2014 to December 30, 2025

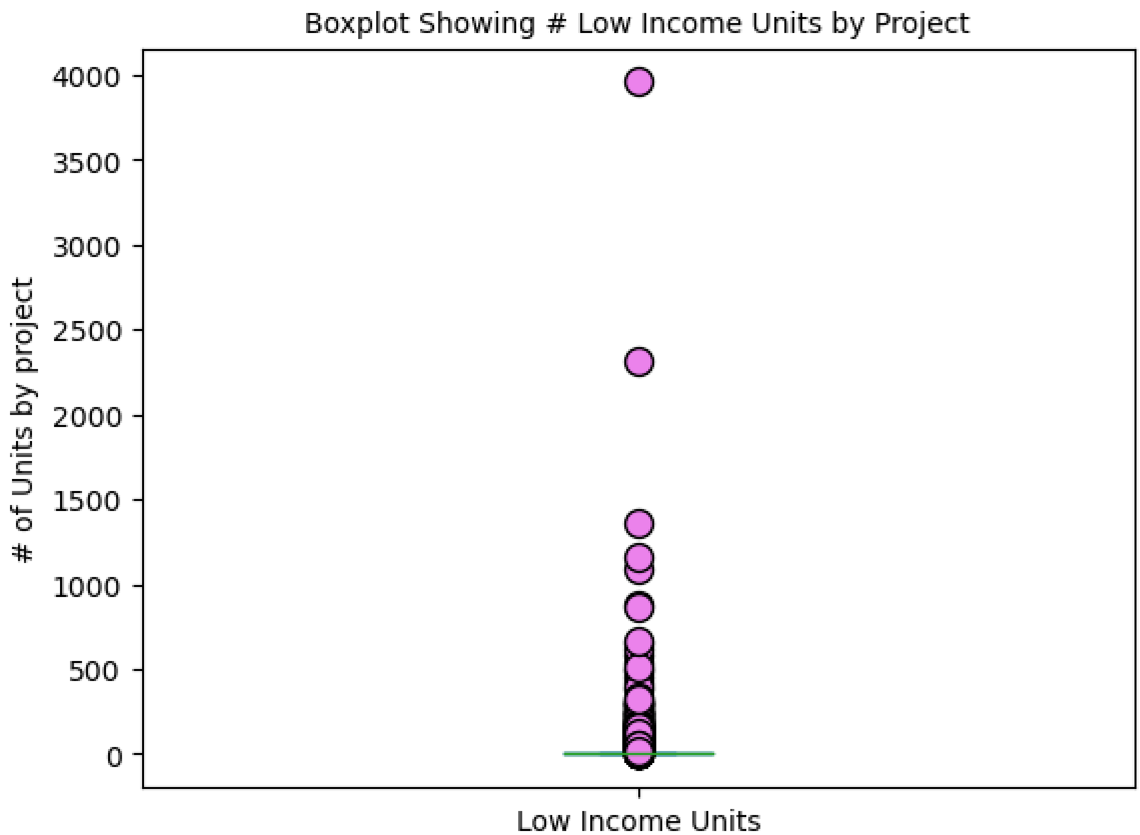
Some notable deductions:

- The majority of projects had under 2000 very low-income units
- One project had 10,000+ of these units

In [12]:
```
# Percent of Completed Very Low Income Units
Very = very.sum()/total.sum()
round(Very*100,2)
very_per = round(Very*100,2)
very_per
```

Out[12]:
```
24.88
```

In [13]:
```
low.plot(kind = "box", flierprops=dict(marker='o', markersize=10, markerfacecolor =
plt.ylabel("# of Units by project")
_=plt.title('Boxplot Showing # Low Income Units by Project', fontsize = 10)
```



This graph shows the distribution of completed low-income units by project built in New York City from January 1, 2014 to December 30, 2025

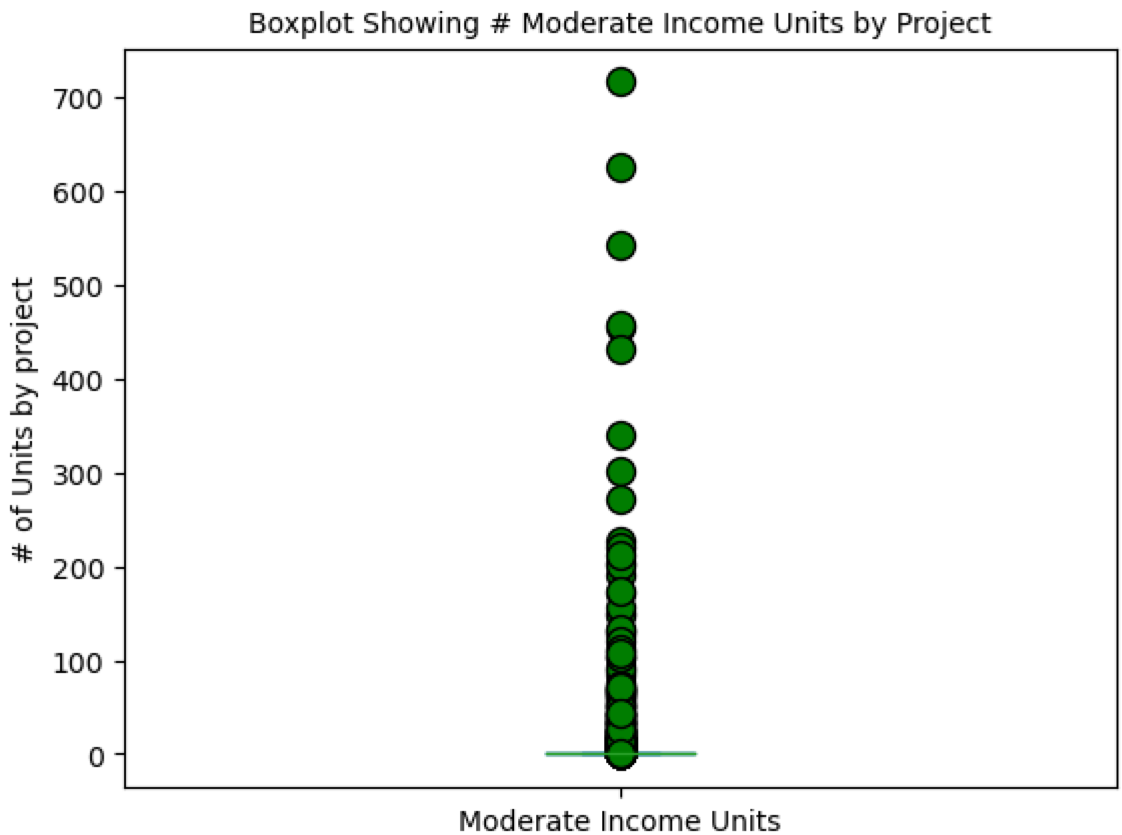
Some notable deductions:

- The majority of projects had under 1000 low-income unit
- Most projects had less than 600 of these units

In [15]:
```
# Percent of Completed Low Income Units
Low = low.sum()/total.sum()
round(Low*100,2)
low_per = round(Low*100,2)
low_per
```

Out[15]: 36.67

In [16]:
```
moderate.plot(kind = "box", flierprops=dict(marker='o', markersize=10, markerfaceco
plt.ylabel("# of Units by project")
_=plt.title('Boxplot Showing # Moderate Income Units by Project', fontsize = 10)
```



This graph shows the distribution of completed moderate income units by project built in New York City from January 1, 2014 to December 30, 2025

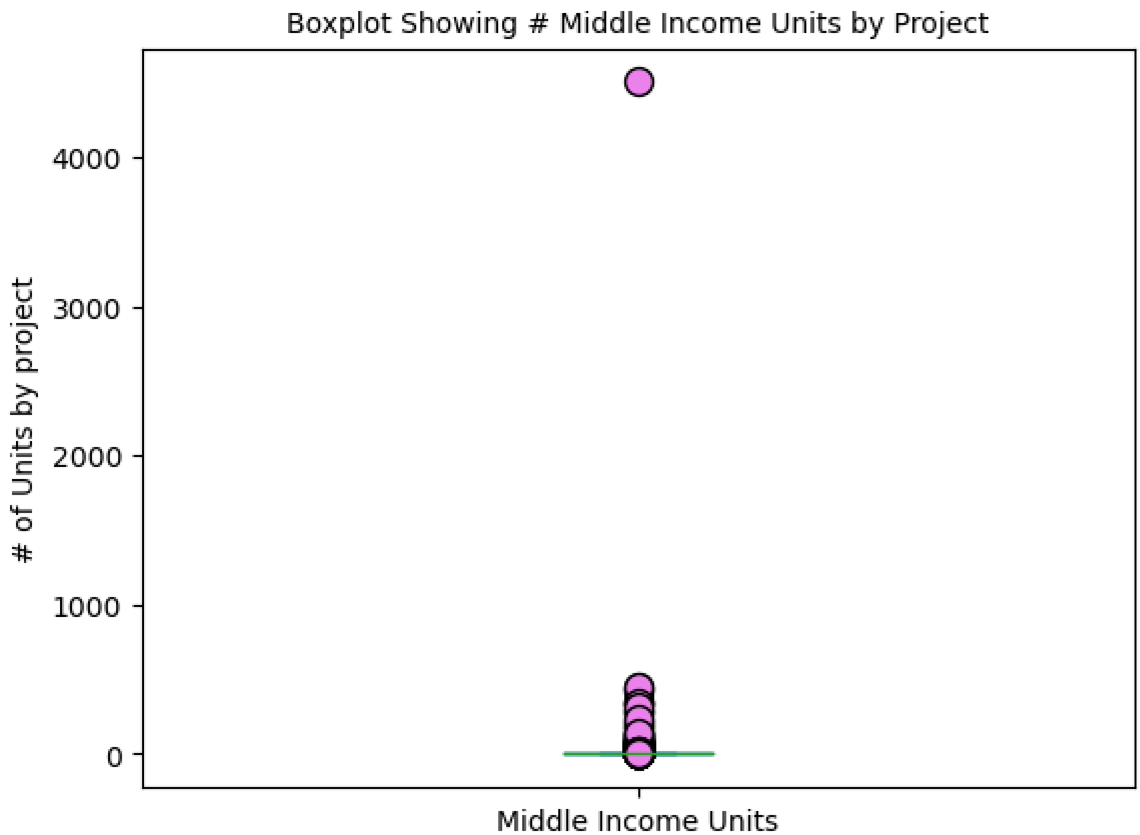
Some notable deductions:

- The majority of projects had under 250 moderate income unit
- Most projects had less than 100 of these units

In [18]:
```
# Percent of Completed Moderate Income Units
Moderate = moderate.sum()/total.sum()
round(Moderate*100,2)
moderate_per = round(Moderate*100,2)
moderate_per
```

Out[18]: 6.85

In [19]:
```
middle.plot(kind = "box", flierprops=dict(marker='o', markersize=10, markerfacecolo
plt.ylabel("# of Units by project")
_=plt.title('Boxplot Showing # Middle Income Units by Project', fontsize = 10)
```



This graph shows the distribution of completed middle income units by project built in New York City from January 1, 2014 to December 30, 2025

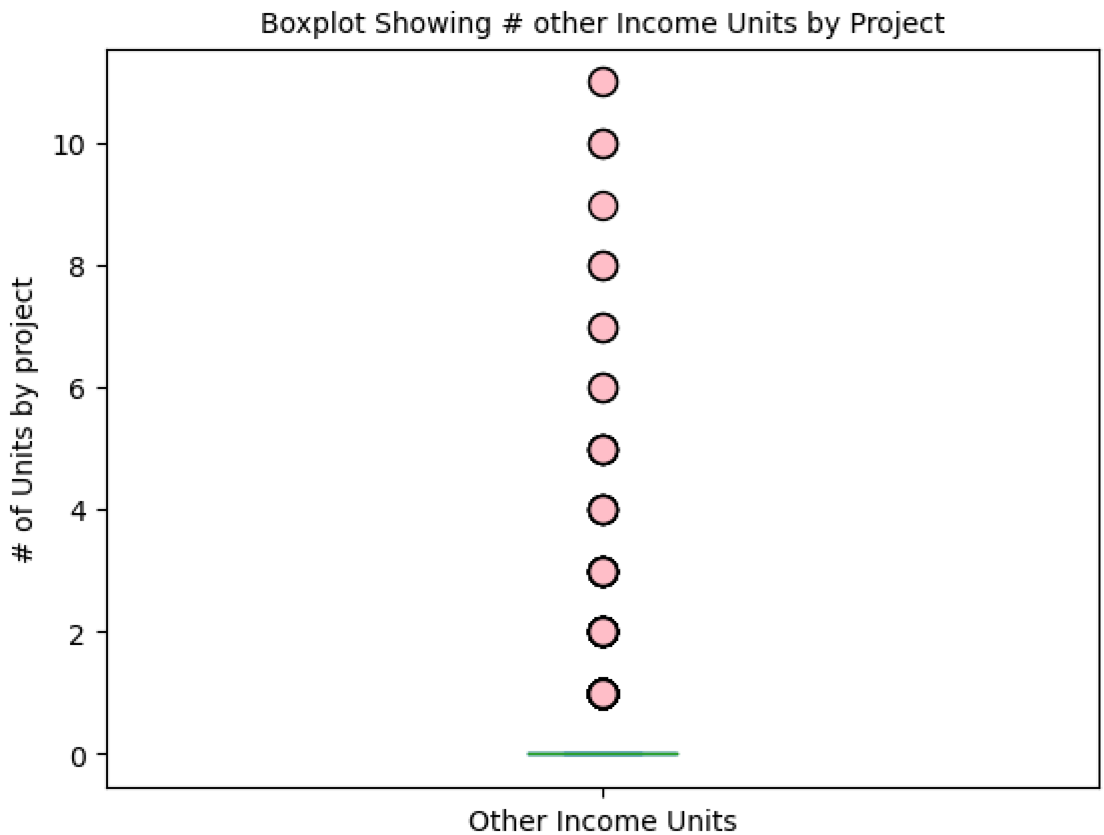
Some notable deductions:

- The majority of projects had less than 150 middle low-income unit

In [22]:
```
# Percent of Completed Middle Income Units
Middle = middle.sum()/total.sum()
round(Middle*100,2)
middle_per = round(Middle*100,2)
middle_per
```

Out[22]: 14.28

In [26]:
```
other.plot(kind = "box", flierprops=dict(marker='o', markersize=10, markerfacecolor
plt.ylabel("# of Units by project")
_=plt.title('Boxplot Showing # other Income Units by Project', fontsize = 10)
```



This graph shows the distribution of completed other income units by project built in New York City from January 1, 2014 to December 30, 2025

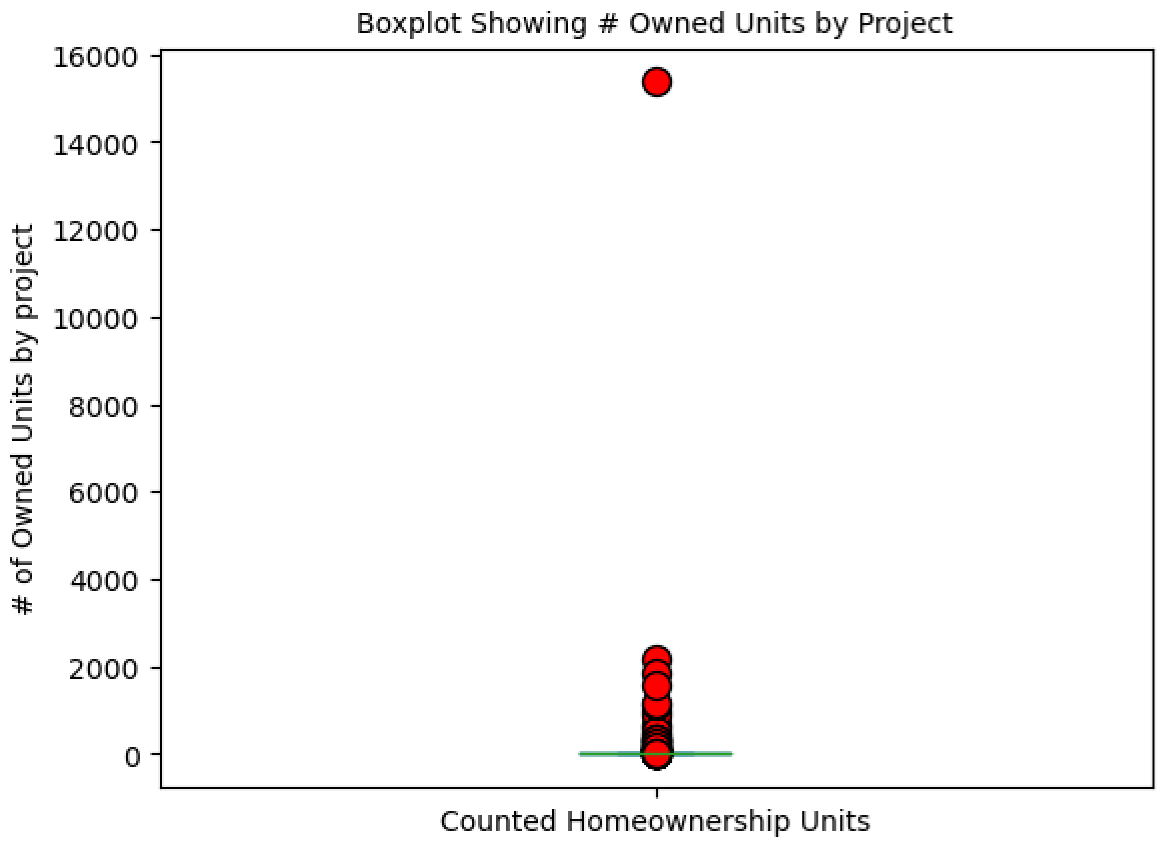
Some notable deductions:

- These units seem to be fairly distributed

In [30]:
```
# Percentage of Completed Other Income Units
Other = other.sum()/total.sum()
round(Other*100,2)
other_per = round(Other*100,2)
other_per
```

Out[30]: 0.47

In [31]:
```
owned.plot(kind = "box" , flierprops=dict(marker='o', markersize=10, markerfacecolo
plt.ylabel("# of Owned Units by project")
_=plt.title('Boxplot Showing # Owned Units by Project', fontsize = 10)
```



This graph shows the distribution of completed owned income units by project built in New York City from January 1, 2014 to December 30, 2025

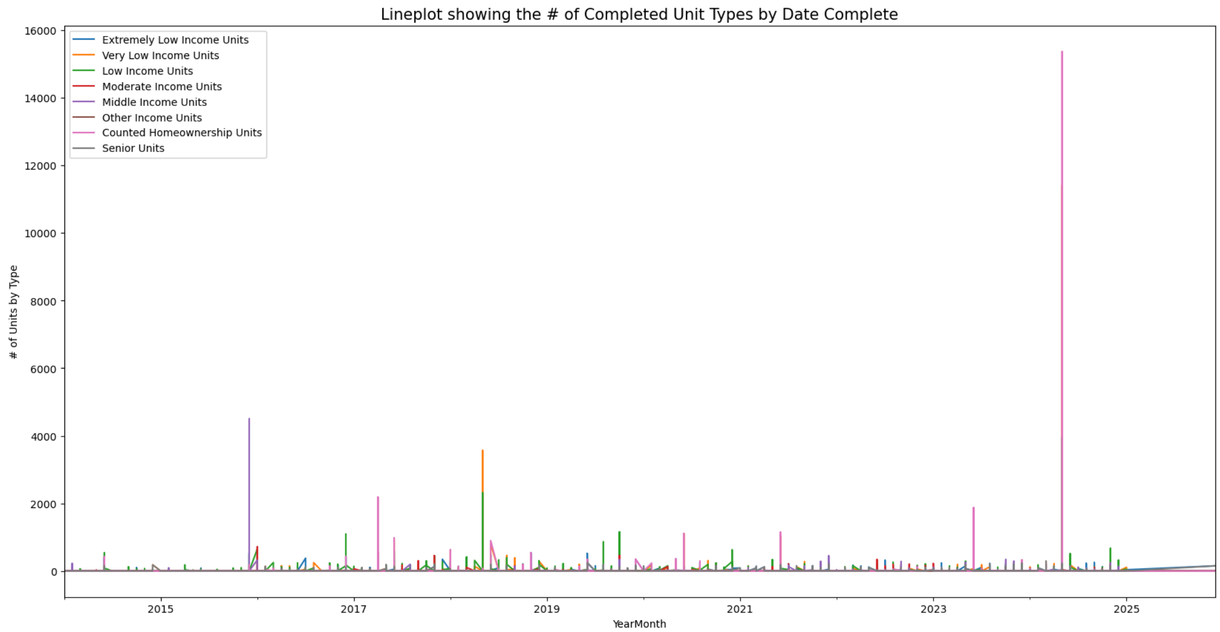
Some notable deductions:

- The majority of projects had less than 2000 owned unit
- One project had 15,000+ units owned, which is an obvious outlier

In [35]:
```
# Percentage of Completed Owned Units
Owned = owned.sum()/total.sum()
round(Owned*100,2)
owned_per = round(Owned*100,2)
owned_per
```

Out[35]: 18.93

In [40]:
```
projects1 = complete_projects.drop(["Project ID", "Total Units", "All Counted Units
projects1['YearMonth'] = projects1['Project Completion Date'].dt.to_period('M')
projects1.drop(["Project Completion Date"], axis = 1).plot(x="YearMonth",  figsize
plt.ylabel("# of Units by Type")
_=plt.title('Lineplot showing the # of Completed Unit Types by Date Complete', font
```



This line graph shows the number of completed units by type built in New York City from January 1, 2014 to December 30, 2025

Some key points shown in the graph are:

- The majority of projects had under 3000 completed units regardless of type
- At least 4 projects had over 2000 units completed which makes them outliers in the data
- The most units completed for a single project type fall under the home-owner category, completed after 2024

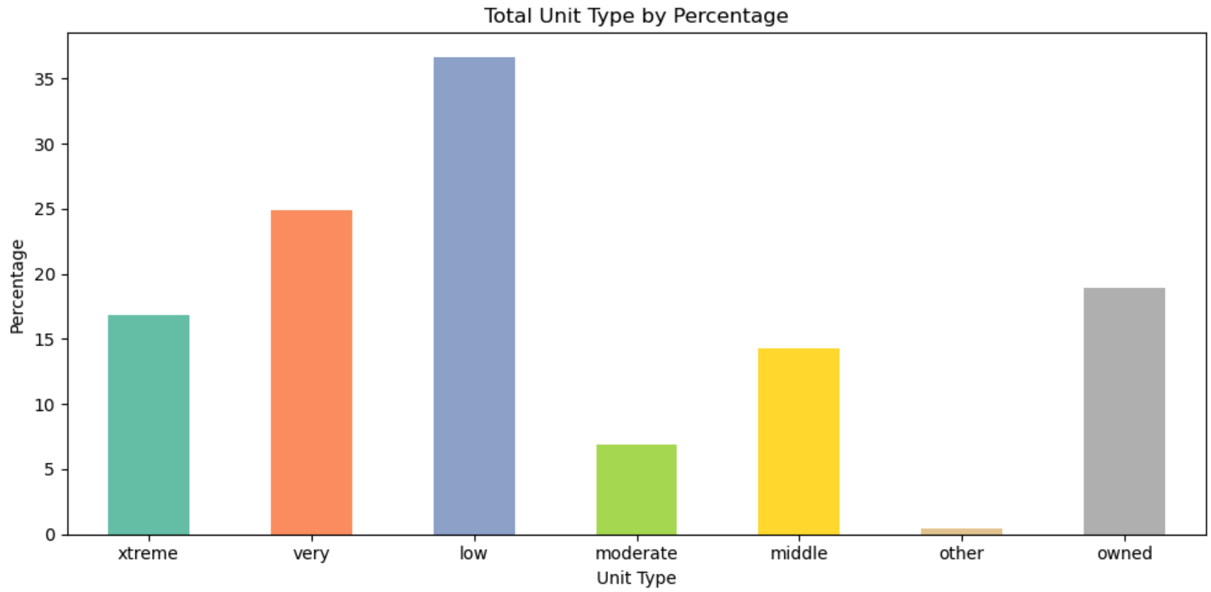
In [44]:

```
data = {
    'Unit Type': ["xtreme", "very", "low", "moderate", "middle", "other", "owned"],
    'Percentages': [16.84, 24.88, 36.67, 6.85, 14.28, 0.47, 18.93]}

df = pd.DataFrame(data)
colors = plt.cm.Set2(np.linspace(0, 1, len(data['Unit Type'])))

df.plot(kind='bar', x='Unit Type', y='Percentages', color=colors, legend=False, fig

plt.title("Total Unit Type by Percentage")
plt.ylabel("Percentage")
plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
plt.savefig("total.png", dpi=300, bbox_inches='tight')
```




```
<Figure size 640x480 with 0 Axes>
```

This bar graph shows the percentage of completed units by type built in New York City from January 1, 2014 to December 30, 2025

Some keys points shown on the chart are:

- Low income units were the most built built in New York City during the 10 year period
- Other income units units were the least built in the period
- Units falling under the xtreme, very and low uncome categories account for the bulk of units builts 78.39%
- Only 19% of units completed are owned

Average Unit Type Completed Completed By Year

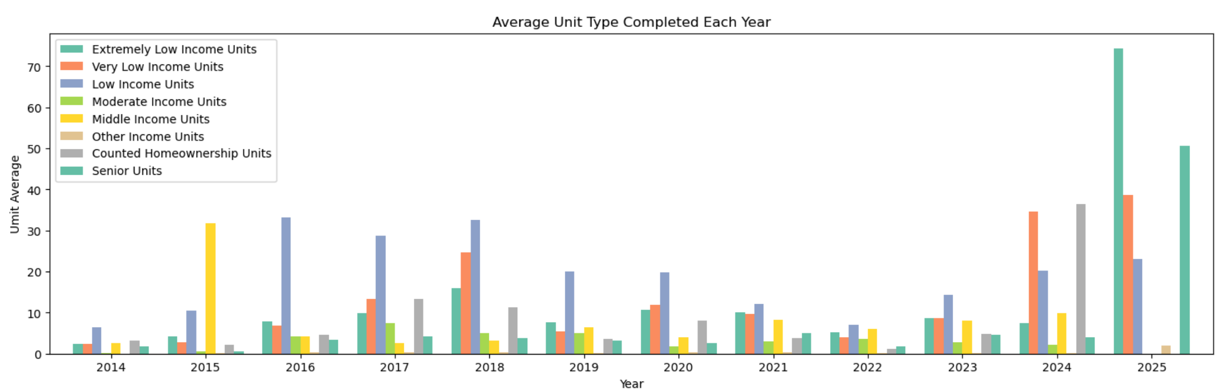
In [48]:
```
# Extract year from date
projects1["Year"] = projects1['Project Completion Date'].dt.year
# Group by year and get average of numeric columns
year_units_avg = projects1.groupby("Year").mean(numeric_only=True).reset_index()
year_units_avg
```

Out[48]:

	Year	Extremely Low Income Units	Very Low Income Units	Low Income Units	Moderate Income Units	Middle Income Units	Other Income Units	Countec Homeownershi Unit
0	2014	2.428571	2.443769	6.528875	0.212766	2.653495	0.042553	3.18845(
1	2015	4.285714	2.809524	10.401361	0.571429	31.700680	0.149660	2.25170
2	2016	7.941441	6.819820	33.184685	4.283784	4.171171	0.256757	4.57207
3	2017	9.892405	13.382911	28.775316	7.522152	2.677215	0.405063	13.22784
4	2018	15.977707	24.738854	32.627389	5.003185	3.121019	0.321656	11.23248
5	2019	7.550117	5.375291	20.053613	5.069930	6.526807	0.240093	3.58974
6	2020	10.588235	11.823529	19.852941	1.860294	3.963235	0.327206	8.10661
7	2021	10.034985	9.769679	12.145773	3.069971	8.172012	0.297376	3.74927
8	2022	5.297189	3.967871	7.056225	3.500000	5.965863	0.112450	1.06827
9	2023	8.575592	8.714026	14.431694	2.723133	8.151184	0.214936	4.75956
10	2024	7.468303	34.697342	20.159509	2.130879	9.860941	0.198364	36.42535
11	2025	74.333333	38.666667	23.000000	0.000000	0.000000	2.000000	0.00000

In [49]:

```
_=year_units_avg.plot(kind='bar', x = "Year", width = .8, color=colors, legend=Fals
_=plt.title("Average Unit Type Completed Each Year")
_=plt.ylabel("Umit Average")
_=plt.xticks(rotation = 0)
```



The bar graph displays the average unit types completed each year

- Low-income units make up the largest portion, at approximately 36%.
- Very low-income units follow at around 25%.
- Owned units represent about 19%.
- Extreme low-income units make up about 17%.

- Middle-income units are at 14%.
- Moderate-income units account for roughly 7%.
- Other unit types make up a very small portion, just under 1%.

Overall, the chart highlights that the majority of units fall within the low- and very low-income categories, indicating a significant focus on lower cost housing.

In [52]:
```
# Data Summary
projects1.drop(["Project Completion Date", "Year"], axis = 1).describe()
```

Out[52]:

	Extremely Low Income Units	Very Low Income Units	Low Income Units	Moderate Income Units	Middle Income Units	Other Income Units	Ho
count	3911.000000	3911.000000	3911.000000	3911.000000	3911.000000	3911.000000	
mean	8.211966	12.130913	17.876502	3.341089	6.963436	0.228330	
std	36.071743	196.941309	94.876869	26.751088	76.357427	0.746878	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
50%	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	
75%	0.000000	0.000000	2.000000	0.000000	3.000000	0.000000	
max	545.000000	11413.000000	3959.000000	716.000000	4505.000000	11.000000	

In [53]:
```
# Create New Dataframe for
projects2 = projects1.drop(["Project Name", "Program Group", "Project Start Date",
projects2
```

Out[53]:

	Prevailing Wage Status	Extremely Low Income Units	Very Low Income Units	Low Income Units	Moderate Income Units	Middle Income Units	Other Income Units	Counted Homeownership Units
0	Non Prevailing Wage	0	0	0	0	1	0	[illegible]
1	Non Prevailing Wage	0	0	0	0	1	0	[illegible]
2	Non Prevailing Wage	0	0	0	0	1	0	[illegible]
3	Non Prevailing Wage	0	0	1	0	0	0	[illegible]
4	Non Prevailing Wage	0	0	0	0	1	0	[illegible]
...	...	...	...	...	...	...	...	...
3906	Non Prevailing Wage	0	0	0	0	2	0	[illegible]
3907	Non Prevailing Wage	0	0	1	0	0	0	[illegible]
3908	Non Prevailing Wage	35	108	54	0	0	4	[illegible]
3909	Non Prevailing Wage	36	8	15	0	0	1	[illegible]
3910	Prevailing Wage	152	0	0	0	0	1	[illegible]

3911 rows × 9 columns

In [57]:

```
print(projects2.columns.tolist())
projects2 = pd.get_dummies(projects2, columns = ["Prevailing Wage Status"], drop_fi
projects2
```

```
['Prevailing Wage Status', 'Extremely Low Income Units', 'Very Low Income Units', 'L
ow Income Units', 'Moderate Income Units', 'Middle Income Units', 'Other Income Unit
s', 'Counted Homeownership Units', 'Senior Units']
```

Out[57]:

	Extremely Low Income Units	Very Low Income Units	Low Income Units	Moderate Income Units	Middle Income Units	Other Income Units	Counted Homeownership Units	Senior Units	S
0	0	0	0	0	1	0	1	0	
1	0	0	0	0	1	0	1	0	
2	0	0	0	0	1	0	1	0	
3	0	0	1	0	0	0	1	0	
4	0	0	0	0	1	0	1	0	
...	...	...	...	...	...	...	...	...	
3906	0	0	0	0	2	0	0	0	
3907	0	0	1	0	0	0	1	0	
3908	35	108	54	0	0	4	0	0	
3909	36	8	15	0	0	1	0	0	
3910	152	0	0	0	0	1	0	152	

3911 rows × 9 columns

In [58]:

```
projects2.columns = projects2.columns.str.replace(' ', '_')
projects2
```

Out[58]:

	Extremely_Low_Income_Units	Very_Low_Income_Units	Low_Income_Units	Moderate_I
0	0	0	0	
1	0	0	0	
2	0	0	0	
3	0	0	1	
4	0	0	0	
...	...	...	...	
3906	0	0	0	
3907	0	0	1	
3908	35	108	54	
3909	36	8	15	
3910	152	0	0	

3911 rows × 9 columns

In [60]:
```
x = projects2.drop("Prevailing_Wage_Status_Prevailing_Wage", axis = 1)
y = projects2["Prevailing_Wage_Status_Prevailing_Wage"]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size= .2, random_sta
```

In [62]:
```
model = LogisticRegression()
model.fit(x_train, y_train)
```

Out[62]:
```
▾ LogisticRegression
LogisticRegression()
```

In [63]:
```
y_test_pred = model.predict(x_test)
y_test_pred
print("Coefficients:", model.coef_)
```

```
Coefficients: [[ 0.01178669 -0.01377399  0.00336534 -0.16988005  0.00544764  0.38368
812
  -0.00297958  0.019909   ]]
```

Logistic Model 1

In [66]:
```
logit_model = smf.logit("Prevailing_Wage_Status_Prevailing_Wage ~ Extremely_Low_Inc
logit_model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.065639
         Iterations 13
```

Out[66]:

Logit Regression Results

Dep. Variable:	Prevailing_Wage_Status_Prevailing_Wage	**No. Observations:**	3911
Model:	Logit	**Df Residuals:**	3902
Method:	MLE	**Df Model:**	8
Date:	Mon, 12 May 2025	**Pseudo R-squ.:**	0.3290
Time:	12:15:34	**Log-Likelihood:**	-256.71
converged:	True	**LL-Null:**	-382.57
Covariance Type:	nonrobust	**LLR p-value:**	7.430e-50

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	-4.6378	0.172	-26.925	0.000	-4.975	-4.300
Extremely_Low_Income_Units	0.0132	0.002	5.640	0.000	0.009	0.018
Very_Low_Income_Units	-0.0151	0.006	-2.405	0.016	-0.027	-0.003
Low_Income_Units	0.0034	0.001	2.602	0.009	0.001	0.006
Moderate_Income_Units	-0.2176	0.119	-1.835	0.066	-0.450	0.015
Middle_Income_Units	-4.81e-05	0.001	-0.043	0.966	-0.002	0.002
Other_Income_Units	0.3322	0.149	2.223	0.026	0.039	0.625
Counted_Homeownership_Units	-0.0047	0.004	-1.305	0.192	-0.012	0.002
Senior_Units	0.0223	0.003	7.243	0.000	0.016	0.028

In [72]:
```
con_matrix = logit_model.pred_table()
con_matrix
```

Out[72]:
```
array([[3822.,   11.],
       [  56.,   22.]])
```

In [74]:
```
true_neg = con_matrix[0][0]
false_neg = con_matrix[1][0]
false_pos = con_matrix[0][1]
true_pos = con_matrix[1][1]
```

In [76]:
```
accuracy = round((true_pos + true_neg)/len(projects2),2)
accuracy
```

Out[76]:
```
0.98
```

In [78]:
```
sensitivity = round(true_pos/(true_pos + false_neg),2)
sensitivity
```

Out[78]:
```
0.28
```

In [79]:
```
specificity = round(true_neg/(true_neg + false_pos),2)
specificity
```

Out[79]: 1.0

In [82]:
```
precision = round(true_pos/(true_pos + false_pos),2)
precision
```

Out[82]: 0.67

Logistic Model 2 Excluding - Middle_Income_Units & Counted_Homeownership_Units

In [88]:
```
logit_model1 = smf.logit("Prevailing_Wage_Status_Prevailing_Wage ~ Extremely_Low_In
logit_model1.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.065884
         Iterations 12
```

Out[88]:

Logit Regression Results

Dep. Variable:	Prevailing_Wage_Status_Prevailing_Wage	**No. Observations:**	3911
Model:	Logit	**Df Residuals:**	3904
Method:	MLE	**Df Model:**	6
Date:	Mon, 12 May 2025	**Pseudo R-squ.:**	0.3265
Time:	12:15:35	**Log-Likelihood:**	-257.67
converged:	True	**LL-Null:**	-382.57
Covariance Type:	nonrobust	**LLR p-value:**	4.516e-51

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-4.6365	0.171	-27.077	0.000	-4.972	-4.301
Extremely_Low_Income_Units	0.0113	0.002	5.992	0.000	0.008	0.015
Very_Low_Income_Units	-0.0162	0.006	-2.688	0.007	-0.028	-0.004
Low_Income_Units	0.0034	0.001	2.615	0.009	0.001	0.006
Moderate_Income_Units	-0.1745	0.103	-1.693	0.090	-0.376	0.027
Other_Income_Units	0.3561	0.145	2.453	0.014	0.072	0.641
Senior_Units	0.0237	0.003	8.290	0.000	0.018	0.029

In [90]:
```
con_matrix1 = logit_model1.pred_table()
con_matrix1
```

Out[90]:
```
array([[3821.,   12.],
       [  56.,   22.]])
```

In [92]:
```
true_neg1 = con_matrix1[0][0]
false_neg1 = con_matrix1[1][0]
false_pos1 = con_matrix1[0][1]
true_pos1 = con_matrix1[1][1]
```

In [94]:
```
accuracy1 = round((true_pos1 + true_neg1)/len(projects2),2)
accuracy1
```

Out[94]:
```
0.98
```

In [96]:
```
sensitivity1 = round(true_pos1/(true_pos1 + false_neg1),2)
sensitivity1
```

Out[96]:
```
0.28
```

In [98]:
```
specificity1 = round(true_neg1/(true_neg1 + false_pos1),2)
specificity1
```

Out[98]:
```
1.0
```

In [100…
```
precision1 = round(true_pos1/(true_pos1 + false_pos1),2)
precision1
```

Out[100…
```
0.65
```

Logistic Model 2 Excluding - Middle_Income_Units & Counted_Homeownership_Units & Moderate_Income_Units

In [103…
```
logit_model2 = smf.logit("Prevailing_Wage_Status_Prevailing_Wage ~ Extremely_Low_In
logit_model2.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.067185
         Iterations 10
```

Out[103…

Logit Regression Results

Dep. Variable:	Prevailing_Wage_Status_Prevailing_Wage	**No. Observations:**	3911
Model:	Logit	**Df Residuals:**	3905
Method:	MLE	**Df Model:**	5
Date:	Mon, 12 May 2025	**Pseudo R-squ.:**	0.3132
Time:	12:15:37	**Log-Likelihood:**	-262.76
converged:	True	**LL-Null:**	-382.57
Covariance Type:	nonrobust	**LLR p-value:**	9.228e-50

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-4.6660	0.170	-27.406	0.000	-5.000	-4.332
Extremely_Low_Income_Units	0.0102	0.002	5.997	0.000	0.007	0.014
Very_Low_Income_Units	-0.0187	0.006	-2.944	0.003	-0.031	-0.006
Low_Income_Units	0.0027	0.001	2.277	0.023	0.000	0.005
Other_Income_Units	0.2649	0.139	1.913	0.056	-0.007	0.536
Senior_Units	0.0252	0.003	9.144	0.000	0.020	0.031

In [105…

```
con_matrix2 = logit_model2.pred_table()
con_matrix2
```

Out[105…

```
array([[3822.,   11.],
       [  56.,   22.]])
```

Logistic regression Analyis

A logistic regression model was used to determine the prevailing wage status of affordable housing projects. The prevailing wage promotes fair wages for workers to ensure equitable compensation. The primary objective of this analysis was to assess whether factors such as the types of housing units influence the likelihood that a project complies with prevailing wage requirements. To explore this relationship, three logistic regression models were developed and evaluated using different sets of independent variables related to unit types and project characteristics. Key Findings

- Model Fit and Predictive Power:

All models had low Pseudo R-squared values (~0.329) or 32.9%, indicating that they explain only a modest portion of the variation in prevailing wage status. The minimal variation across R-squared values (all in the 0.320 range) suggests limited improvement across models. Despite high overall accuracy (98%), the models suffer from very low sensitivity (28%), meaning they fail to detect most actual cases where prevailing wage standards were met.

- Predictive Accuracy:

The models have 100% perfect specificity, making them highly reliable in identifying when prevailing wage standards were not met. Precision (67%) indicates that when the model predicts a project meets prevailing wage, it is correct about two-thirds of the tie.

- Significant Predictors:

Extremely Low Income Units, Senior Units, and Other Income Units were found to be statistically significant predictors positively associated with prevailing wage compliance.

In contrast, Very Low Income Units showed a negative association, suggesting projects with more of these units may be less likely to meet prevailing wage standards Conclusion While the logistic models provide some insight into which housing unit types may influence prevailing wage outcomes, their overall predictive power is limited. They are better at ruling out projects that do not meet prevailing wage requirements than correctly identifying those that do. Nevertheless, the analysis highlights key factors that could guide targeted interventions or policy adjustments in affordable housing development.

K = 5 Nearest Neighbors Classifier

In [109…

```
knn5 = KNeighborsClassifier(n_neighbors = 5)
knn5.fit(x_train, y_train)
```

Out[109…

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

In [111…

```
y_test_pred = knn5.predict(x_test)
y_test_pred
confusion_matrix(y_test,y_test_pred)
```

Out[111…

```
array([[764,   1],
       [  8,  10]], dtype=int64)
```

In [113…

```
accuracy2 = round((10 + 764)/len(projects2),2)
accuracy
```

Out[113…

```
0.98
```

In [115…

```
sensitivity2 = round(10/(10+8),2)
sensitivity2
```

Out[115…

```
0.56
```

In [117…

```
specificity2 = round(764/(1+764),5)
specificity2
```

Out[117…

```
0.99869
```

In [119…

```
precision2 = round(10/(10 + 1),2)
precision2
```

Out[119…

```
0.91
```

K = 4 Nearest Neighbors Classifier

In [122…

```
knn4 = KNeighborsClassifier(n_neighbors = 4)
knn4.fit(x_train, y_train)
```

Out[122…

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=4)
```

In [124…

```
y_test_predict = knn4.predict(x_test)
y_test_predict
confusion_matrix(y_test,y_test_predict)
```

Out[124…

```
array([[764,   1],
       [  9,   9]], dtype=int64)
```

In [126…

```
accuracy3 = round((9 + 764)/len(projects2),2)
accuracy3
```

Out[126…

```
0.2
```

In [128…

```
sensitivity3 = round(9/(9+9),2)
sensitivity3
```

Out[128…

```
0.5
```

In [130…

```
specificity3 = round(764/(1+764),5)
specificity3
```

Out[130…

```
0.99869
```

In [132…

```
precision3 = round(10/(10 + 1),2)
precision3
```

Out[132…

```
0.91
```

K = 3 Nearest Neighbors Classifier

In [135…

```
knn3 = KNeighborsClassifier(n_neighbors = 3)
knn3.fit(x_train, y_train)
```

Out[135…

```
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=3)
```

In [137…

```
y_test_predict1 = knn3.predict(x_test)
y_test_predict1
confusion_matrix(y_test,y_test_predict1)
```

Out[137…

```
array([[764,   1],
       [  7,  11]], dtype=int64)
```

In [139…

```
accuracy4 = round((11 + 764)/len(projects2),2)
accuracy4
```

Out[139…

```
0.2
```

In [141…

```
sensitivity4 = round(11/(11+1),2)
sensitivity4
```

Out[141…

```
0.92
```

In [143…

```
specificity4 = round(764/(1+764),5)
specificity4
```

Out[143…

```
0.99869
```

In [145…

```
precision4 = round(10/(10 + 1),2)
precision4
```

Out[145…

```
0.91
```

KNN-Classifier

The analysis evaluates the performance of KNN classifiers with different parameters (K = 3, 4, 5) to predict whether affordable housing projects comply with prevailing wage standards. Each model's effectiveness is judged using common classification metrics: Accuracy, Sensitivity (Recall), Specificity, and Precision. Key Observations

- Strengths

All models are extremely reliable at identifying projects that do not meet prevailing wage standards Specificity 99.87%. The model is 91% correct at predicting that projects meet prevailing wage standards, which implies a low false positive rate. The model with K=3 neighbors achieves 92% sensitivity, making it much better at catching true positive cases projects that actually meet prevailing wage standards.

- Weaknesses

Despite strong precision and specificity, the models perform poorly in general classification, only being 20% accurate. This may be due to class imbalance with far more negative than positive cases. K=4 and K=5 have a low sensitivity of 50% and 56% respectively, indicating many false negatives and projects that do meet prevailing wage are misclassified. All models show similar and very low accuracy, suggesting that accuracy alone is not an informative metric for this imbalanced dataset.

Conclusion

While the models are highly effective at detecting when projects do not comply with prevailing wage standards, they struggle to identify those that do, especially in K=4 and K=5 models. The K=3 model strikes the best balance, significantly improving sensitivity while maintaining high precision and specificity. However, the low accuracy and likely class imbalance indicate that further model tuning, resampling or alternative classifiers like Random Forest and Logistic Regression with regularization, may be better to build a more balanced and robust predictive model.

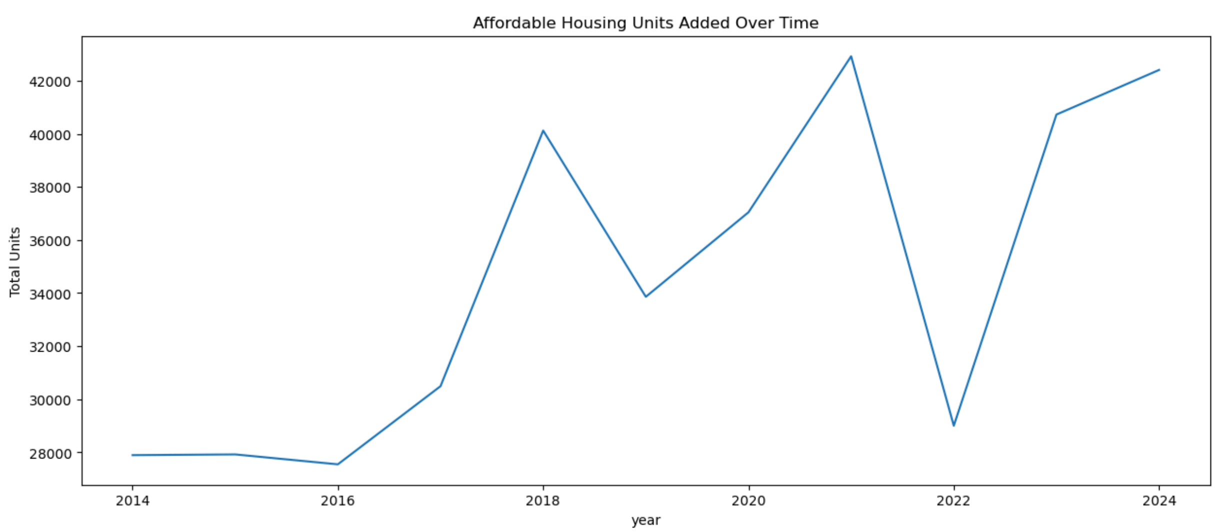
To determine which model is better depends on the desired outcome as each model performs differently, and better than the other in some cases. However, because our goal is to determine whether or not a project is compliant with prevailing wage standards. In this case the KNN classifiers performs better with a 92% sensitivity than the logistic regression and the K=3 is the best model

In [148…

```
affordable_housing['year'] = pd.to_datetime(affordable_housing['Project Start Date'

# Group and plot
yearly_units = affordable_housing.groupby('year')['Total Units'].sum().reset_index(

# Visualization
plt.figure(figsize=(15, 6))
sns.lineplot(data=yearly_units, x='year', y='Total Units')
plt.title("Affordable Housing Units Added Over Time")
plt.show()
```

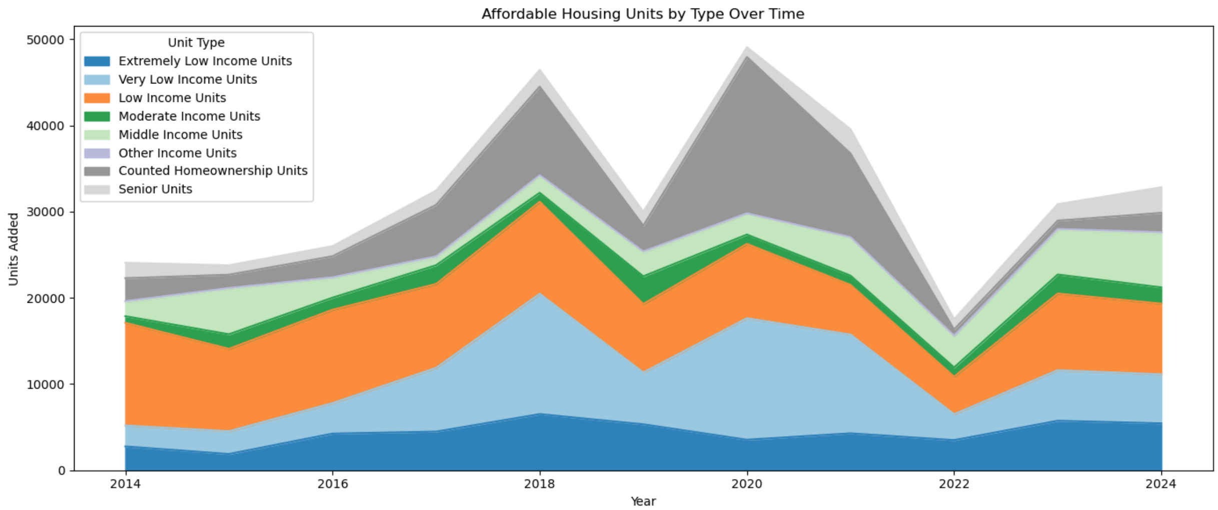


In [160…

```
unit_cols = ['Extremely Low Income Units', 'Very Low Income Units',     'Low Income
grouped = affordable_housing.groupby('year')[unit_cols].sum()

grouped.plot(kind='area', stacked=True, figsize=(14, 6), colormap='tab20c')
plt.title("Affordable Housing Units by Type Over Time")
plt.xlabel("Year")
plt.ylabel("Units Added")
plt.legend(title="Unit Type", loc='upper left')
```

```
plt.tight_layout()
plt.show()
```



In [152…

```
unit_cols = [
    'Extremely Low Income Units', 'Very Low Income Units', 'Low Income Units',
    'Moderate Income Units', 'Middle Income Units', 'Other Income Units',
    'Counted Homeownership Units', 'Senior Units'
]

# Group and sum by year
grouped = affordable_housing.groupby('year')[unit_cols].sum().reset_index()

# Create interactive stacked area chart
fig = go.Figure()

for col in unit_cols:
    fig.add_trace(go.Scatter(
        x=grouped['year'],
        y=grouped[col],
        mode='lines',
        name=col,
        stackgroup='one',  # enables stacking
        hoverinfo='x+y+name'
    ))

fig.update_layout(
    title="Affordable Housing Units by Type Over Time (Interactive)",
    xaxis_title="Year",
    yaxis_title="Units Added",
    hovermode='x unified',
    legend_title="Unit Type",
    template='plotly_white',
    width=1500,
    height=700
)

fig.show()
```

```
In [ ]:
```