Homework 1

Mateus Silva Aragao - msa8779

1) Median age of the houses in the block (in years) 2) Total number of rooms in a given block 3) Number of bedrooms in a given block 4) Population in the block 5) Number of households in the block 6) Median household income in the block (in thousands of dollars) 7) Proximity to the ocean (rated on a scale from 0 = closest to 4 = farthest) 8) Median house value in the block (in dollars)(outcome)

Loading the data and Libraries

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression


data = pd.read_csv('housingUnits.csv')
data.head(5)
```

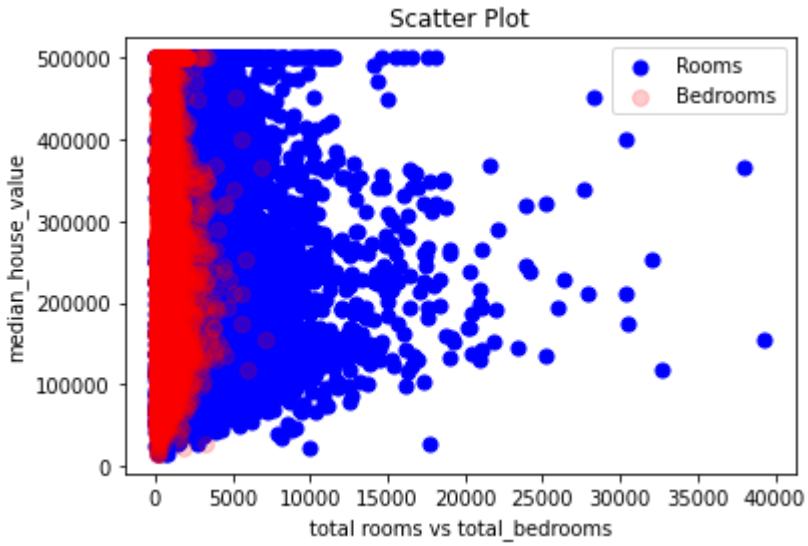
Out[ ]:

	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
0	41	880	223	322	126	208.1300
1	21	7099	1338	2401	1138	207.5350
2	52	1467	328	496	177	181.4350
3	52	1274	293	558	219	141.0775
4	52	1627	357	565	259	96.1550

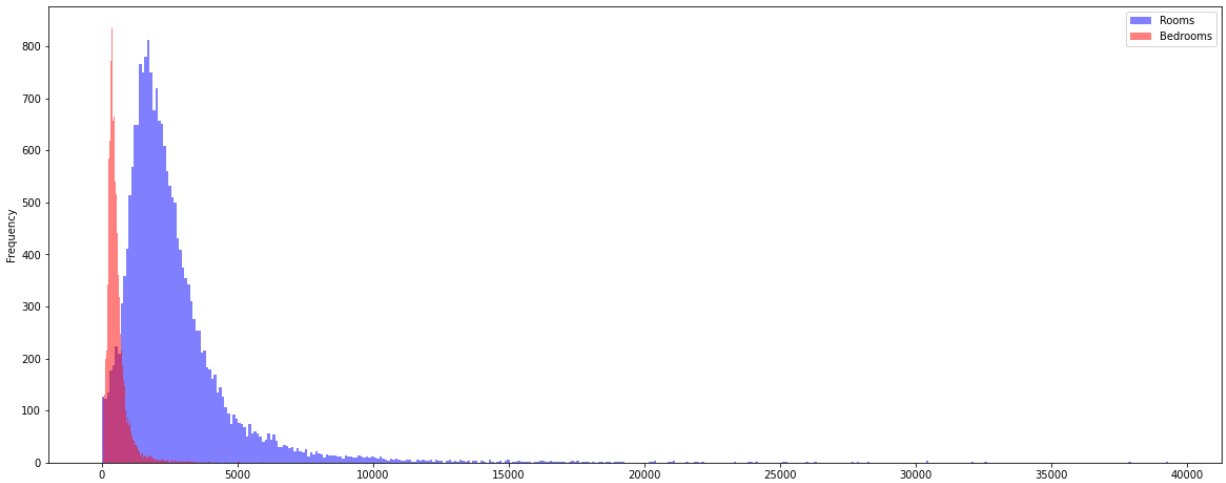
1. Why is it a good idea to standardize/normalize the predictor variables 2 and 3 and why are predictor variables 4 and 5 probably not very useful by themselves to predict median house values in a block?

In [ ]:
```
# Create two data series with random values
x_rooms = data['total_rooms']
x_bed = data['total_bedrooms']
y = data['median_house_value']

# Create a scatter plot of the data
plt.figure()
plt.scatter(x_rooms, y, s=50, c='blue', label="Rooms")
plt.scatter(x_bed, y, s=60, c='red',alpha=0.2,label="Bedrooms")

# Set the axis labels and title
plt.xlabel('total rooms vs total_bedrooms')
plt.ylabel('median_house_value')
```

```
plt.title('Scatter Plot')
plt.legend()



# Display the plot
plt.show()

plt.figure()
plt.figure(figsize=(20, 8))
data['total_rooms'].plot.hist(bins=400,color = 'blue',alpha=.5, label="Rooms"
data['total_bedrooms'].plot.hist(bins=400,color = 'red',alpha=.5,label="Bedro
plt.legend()
plt.show()
```




```
<Figure size 432x288 with 0 Axes>
```



Answer 1:

For this first question, I plotted both variables in relation to our outcome variables and then created a bar graph for both variables. My approach was to understand each variable. I realized that all of the variables were on different scales in relation to the median value of a house/unit in a block.

Variables 2 and 3 refer to the total number of rooms and bedrooms in a given block. We must find a way to normalize these values so they reflect the median value of a house in the block as closely as possible. Essentially, we are trying to predict the price of a unit based on the total number of rooms/bedrooms in the block where that unit is located.

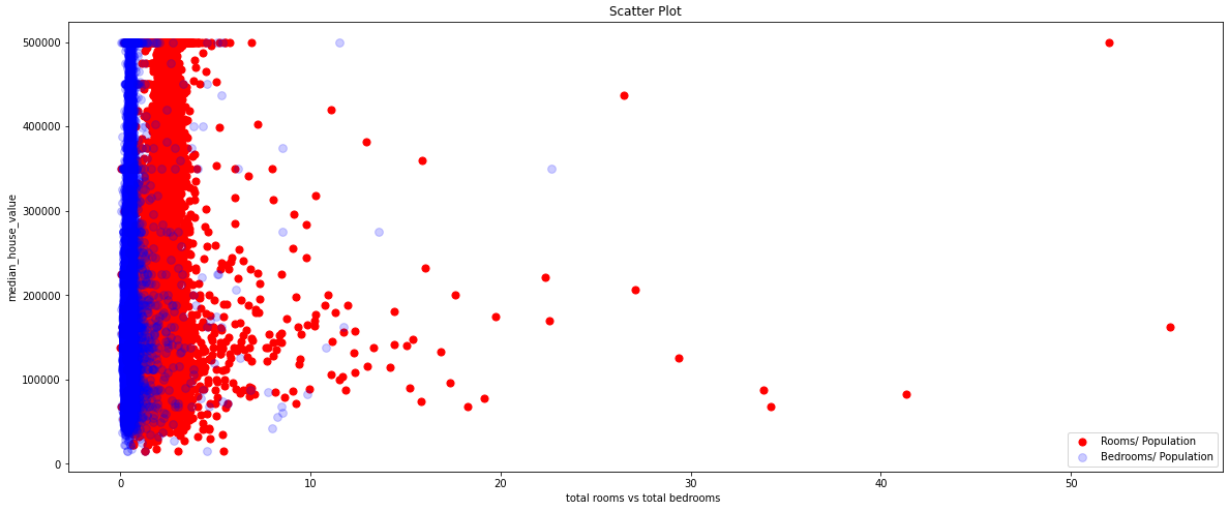
Variables 4 and 5 cannot be used as standalone measurements either, as they refer to the whole of a block (total population/households), while our predictor is the price representation of a single unit in that block.

2. To meaningfully use predictor variables 2 (number of rooms) and 3 (number of bedrooms), you will need to standardize/normalize them. Using the data, is it better to normalize them by population (4) or number of households (5)?

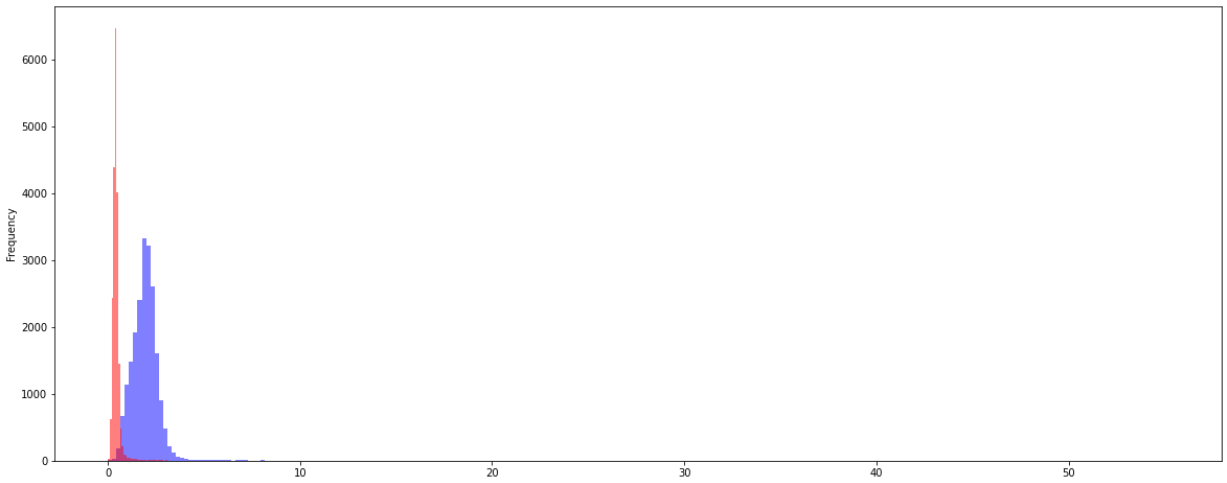
In [ ]:

```
# lets try to normalize by both of the the columns.

# First with the population column:

normalized_room_pop = data['total_rooms']/data['population']
normalized_bed_pop = data['total_bedrooms']/data['population']


y = data['median_house_value']

# Create a scatter plot of the data
plt.figure()
plt.figure(figsize=(20, 8))
plt.scatter(normalized_room_pop, y, s=50, c='red',label='Rooms/ Population')
plt.scatter(normalized_bed_pop, y, s=60, c='blue',alpha=0.2,label='Bedrooms/ 

# Set the axis labels and title
plt.xlabel('total rooms vs total bedrooms ')
plt.ylabel('median_house_value')
plt.title('Scatter Plot')

plt.legend()
# Display the plot
plt.show()

plt.figure()
plt.figure(figsize=(20, 8))
normalized_room_pop.plot.hist(bins=250,color = 'blue',alpha=.5)
normalized_bed_pop.plot.hist(bins=250,color = 'red',alpha=.5)

plt.show()

model = LinearRegression()
model.fit(np.array(normalized_room_pop).reshape(-1,1), np.array(y).reshape(-1
print("rooms normalized by pop")
print("Model Coefficients: ", model.coef_)

model = LinearRegression()
model.fit(np.array(normalized_bed_pop).reshape(-1,1), np.array(y).reshape(-1,
print("bedrooms normalized by pop")
print("Model Coefficients: ", model.coef_)
```

```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```



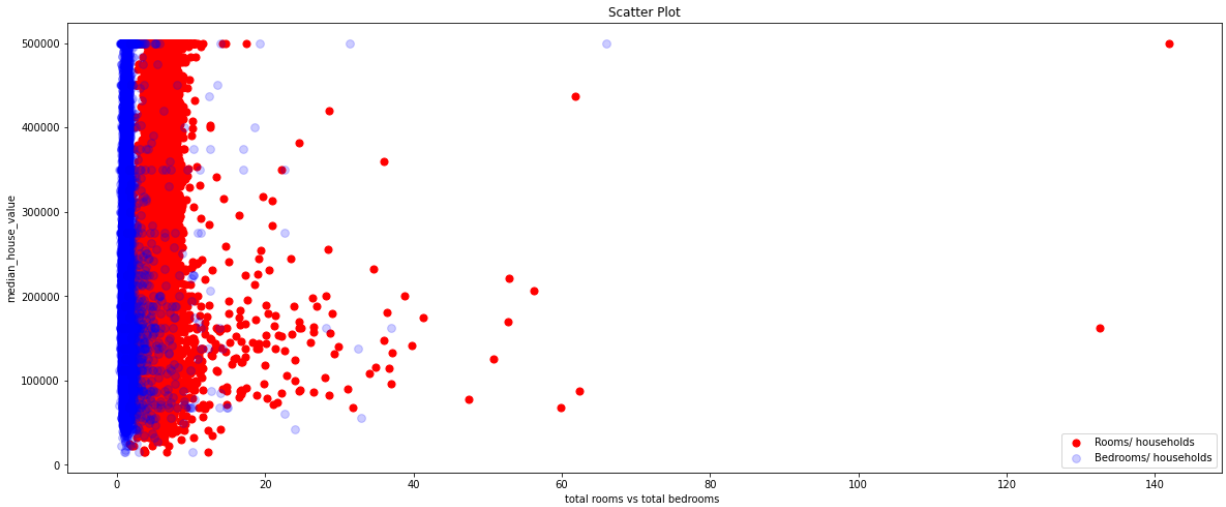

```
rooms normalized by pop
Model Coefficients:  [[21093.26808193]]
bedrooms normalized by pop
Model Coefficients:  [[33435.00816963]]
```

In [ ]:

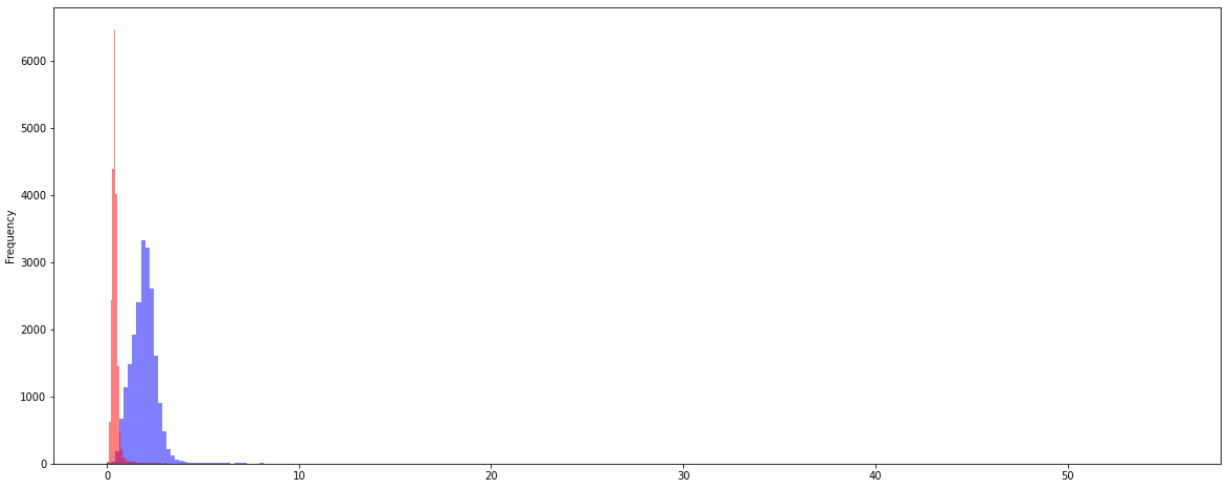
```
# Now with the households column:

normalized_room_house = data['total_rooms']/data['households']
normalized_bed_house = data['total_bedrooms']/data['households']


y = data['median_house_value']

# Create a scatter plot of the data
plt.figure()
plt.figure(figsize=(20, 8))
plt.scatter(normalized_room_house, y, s=50, c='red',label='Rooms/ households'
plt.scatter(normalized_bed_house, y, s=60, c='blue',alpha=0.2,label='Bedrooms

# Set the axis labels and title
plt.xlabel('total rooms vs total bedrooms ')
plt.ylabel('median_house_value')
plt.title('Scatter Plot')

plt.legend()
# Display the plot
plt.show()

plt.figure()
plt.figure(figsize=(20, 8))
normalized_room_pop.plot.hist(bins=250,color = 'blue',alpha=.5)
normalized_bed_pop.plot.hist(bins=250,color = 'red',alpha=.5)
```

```
plt.show()

model = LinearRegression()
model.fit(np.array(normalized_room_house).reshape(-1,1), np.array(y).reshape(
print("rooms normalized by pop")
print("Model Coefficients: ", model.coef_)

model = LinearRegression()
model.fit(np.array(normalized_bed_house).reshape(-1,1), np.array(y).reshape(-
print("bedrooms normalized by pop")
print("Model Coefficients: ", model.coef_)
```

```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```




```
rooms normalized by pop
Model Coefficients:  [[7086.87932804]]
bedrooms normalized by pop
Model Coefficients:  [[6532.07283508]]
```

Answer 2:

I initially assumed that normalizing the total number of rooms and number of bedrooms in a given block by the number of households in the block would be the best representation of the housing unit in that block. To test this, I experimented with both variables 4 and 5, normalizing them and plotting the linear regression to see how they affected variables 2 and 3.

It turns out that using the population variable as a medium for normalization was the best option. Examining the coefficients from both regressions, we realized that normalizing with

the population variable made variables 2 and 3 more correlated to our target variable.

Therefore, it is best to continue normalizing these values with the population variable going forward.

3. Which of the seven variables is most *and* least predictive of housing value, from asimple linear regression perspective? [Hints: a) Make sure to use thestandardized/normalized variables from 2. above; b) Make sure to inspect the scatterplots and comment on a potential issue – would the best predictor be even more predictive if not for an unfortunate limitation of the data?]

In [ ]:

```
# pre processing step
from sklearn.preprocessing import MinMaxScaler

# normalizing the outcome variable
# Assuming the column to be normalized is stored in a NumPy array X
column_to_normalize = np.array(data['median_house_value'])  # Select the colu
scaler = MinMaxScaler()          # Create a MinMaxScaler object
outcome = scaler.fit_transform(column_to_normalize.reshape(-1, 1)).reshape(-1

column_to_normalize = np.array(data['housing_median_age'])  # Select the colu
scaler = MinMaxScaler()          # Create a MinMaxScaler object
median_age_val = scaler.fit_transform(column_to_normalize.reshape(-1, 1))  #

column_to_normalize = np.array(normalized_room_pop)  # Select the column you
scaler = MinMaxScaler()          # Create a MinMaxScaler object
room_amount_val = scaler.fit_transform(column_to_normalize.reshape(-1, 1))  #

column_to_normalize = np.array(normalized_bed_pop)  # Select the column you w
scaler = MinMaxScaler()          # Create a MinMaxScaler object
bed_amount_val = scaler.fit_transform(column_to_normalize.reshape(-1, 1))  #

column_to_normalize = np.array(data['population'])  # Select the column you w
scaler = MinMaxScaler()          # Create a MinMaxScaler object
pop_val = scaler.fit_transform(column_to_normalize.reshape(-1, 1))  # Normali

column_to_normalize = np.array(data['households'])  # Select the column you w
scaler = MinMaxScaler()          # Create a MinMaxScaler object
house_val = scaler.fit_transform(column_to_normalize.reshape(-1, 1))  # Norma

column_to_normalize = np.array(data['median_income'])  # Select the column yo
scaler = MinMaxScaler()          # Create a MinMaxScaler object
income_val = scaler.fit_transform(column_to_normalize.reshape(-1, 1))  # Norm

column_to_normalize = np.array(data['ocean_proximity'])  # Select the column
scaler = MinMaxScaler()          # Create a MinMaxScaler object
ocean_val = scaler.fit_transform(column_to_normalize.reshape(-1, 1))  # Norma


df = pd.DataFrame({"median_age_val":median_age_val.reshape(-1),
                   "room_amount_val":room_amount_val.reshape(-1),
```

```
                        "bed_amount_val":bed_amount_val.reshape(-1),
                        "pop_val":pop_val.reshape(-1),
                        "house_val":house_val.reshape(-1),
                        "income_val":income_val.reshape(-1),
                        "ocean_val":ocean_val.reshape(-1)})
```

In [ ]:

```
# predicting using the median age of household

# x =np.array(data['housing_median_age']).reshape(-1,1)
# y = np.array(data['median_house_value'])
# Create a linear regression model and fit the data
model = LinearRegression()
model.fit(median_age_val, outcome)
print("For Housing median age")
print("Model Coefficients: ", model.coef_)
y_pred = model.predict(median_age_val)

# Calculate the R-squared value
r2 = r2_score(outcome, y_pred)

# Print the R-squared value
print('R-squared:', r2)


# x =np.array(normalized_room_house).reshape(-1,1)
# y = np.array(data['median_house_value'])
# Create a linear regression model and fit the data
model = LinearRegression()
model.fit(room_amount_val, outcome)
print("For Room")
print("Model Coefficients: ", model.coef_)
y_pred = model.predict(room_amount_val)

# Calculate the R-squared value
r2 = r2_score(outcome, y_pred)

# Print the R-squared value
print('R-squared:', r2)

# x =np.array(normalized_bed_house).reshape(-1,1)
# y = np.array(data['median_house_value'])
# Create a linear regression model and fit the data
model = LinearRegression()
model.fit(bed_amount_val, outcome)
print("For bedrooms")
print("Model Coefficients: ", model.coef_)
y_pred = model.predict(bed_amount_val)

# Calculate the R-squared value
r2 = r2_score(outcome, y_pred)

# Print the R-squared value
print('R-squared:', r2)

# x =np.array(data['population']).reshape(-1,1)
# y = np.array(data['median_house_value'])
# Create a linear regression model and fit the data
model = LinearRegression()
model.fit(pop_val, outcome)
print("For population")
print("Model Coefficients: ", model.coef_)
y_pred = model.predict(pop_val)
```

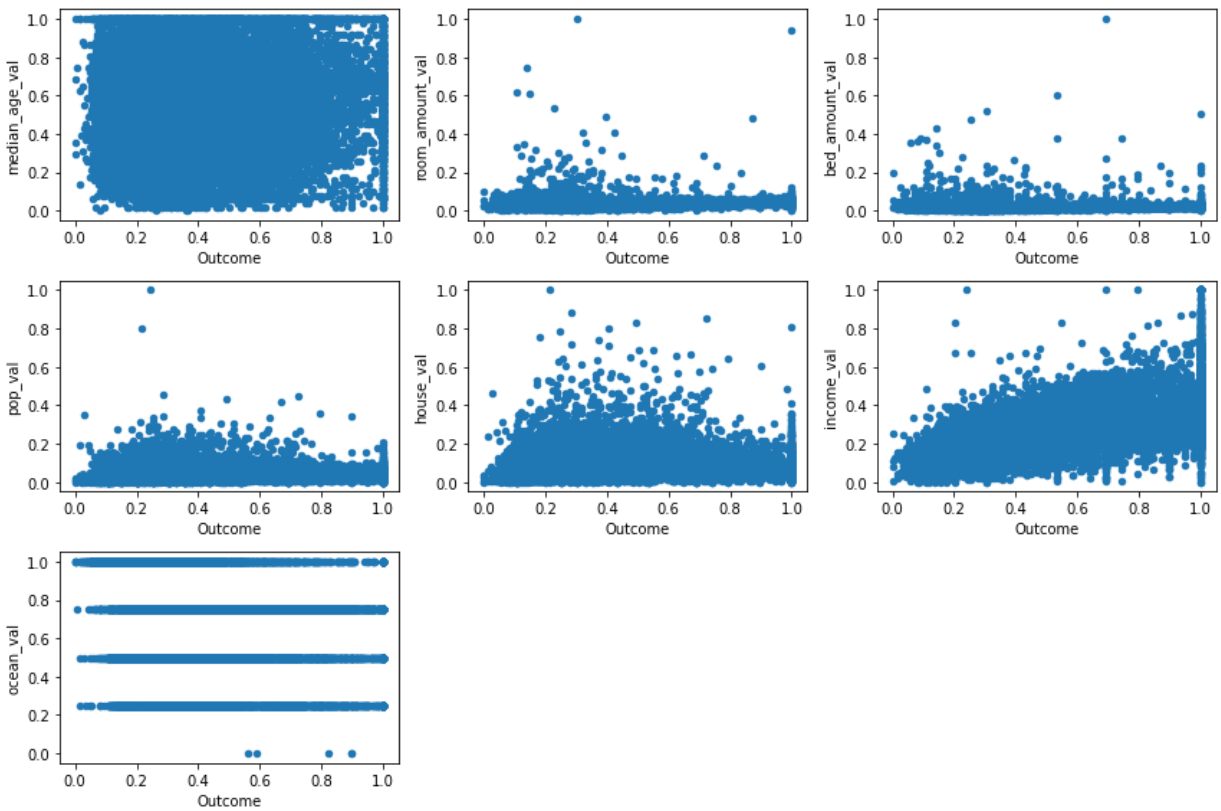
```
# Calculate the R-squared value
r2 = r2_score(outcome, y_pred)

# Print the R-squared value
print('R-squared:', r2)

# x =np.array(data['households']).reshape(-1,1)
# y = np.array(data['median_house_value'])
# Create a linear regression model and fit the data
model = LinearRegression()
model.fit(house_val, outcome)
print("For household")
print("Model Coefficients: ", model.coef_)
y_pred = model.predict(house_val)

# Calculate the R-squared value
r2 = r2_score(outcome, y_pred)

# Print the R-squared value
print('R-squared:', r2)

# x =np.array(data['median_income']).reshape(-1,1)
# y = np.array(data['median_house_value'])
# Create a linear regression model and fit the data
model = LinearRegression()
model.fit(income_val, outcome)
print("For median_income")
print("Model Coefficients: ", model.coef_)
y_pred = model.predict(income_val)

# Calculate the R-squared value
r2 = r2_score(outcome, y_pred)

# Print the R-squared value
print('R-squared:', r2)

# x =np.array(data['ocean_proximity']).reshape(-1,1)
# y = np.array(data['median_house_value'])
# Create a linear regression model and fit the data
model = LinearRegression()
model.fit(ocean_val, outcome)
print("For ocean_proximity")
print("Model Coefficients: ", model.coef_)
y_pred = model.predict(ocean_val)

# Calculate the R-squared value
r2 = r2_score(outcome, y_pred)

# Print the R-squared value
print('R-squared:', r2)
```

```
For Housing median age
Model Coefficients:  [0.10183655]
R-squared: 0.011156305266710742
For Room
Model Coefficients:  [2.40156415]
R-squared: 0.04388269533891953
For bedrooms
Model Coefficients:  [1.56196349]
R-squared: 0.012790501296178758
For population
Model Coefficients:  [-0.1847762]
R-squared: 0.0006076066693256887
For household
Model Coefficients:  [0.2491667]
```

```
R-squared: 0.0043352546340905684
For median_income
Model Coefficients:  [1.24951891]
R-squared: 0.47344749180719903
For ocean_proximity
Model Coefficients:  [-0.40051884]
R-squared: 0.15780848616855125
```

In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Create a Pandas DataFrame with 7 columns of random data
# df = pd.DataFrame(np.random.randn(100, 7), columns=['A', 'B', 'C', 'D', 'E'

# Create a figure with 7 subplots in a 3x3 grid
fig, axes = plt.subplots(nrows=3, ncols=3, figsize=(12, 8))

# Flatten the axes into a 1D array
axes = axes.flatten()

# Create a scatter plot on each subplot
for i, col in enumerate(df.columns):
    ax = axes[i]
    ax.scatter(outcome, df[col], s=20)
    ax.set_xlabel('Outcome')
    ax.set_ylabel(col)

# Hide any unused subplots
for i in range(7, 9):
    fig.delaxes(axes[i])

# Adjust the layout of the subplots
plt.tight_layout()

# Display the plot
plt.show()
```



Answer 3:

For this question, I standardized all of the predictors to be on the same scale for our simple linear regressions, to ensure that the coefficients were not absurdly large or small. After running a simple linear regression for each predictor, I found that the median income variable was the best predictor, and the population variable was the worst. To ensure that the best predictor was as reliable as possible, I took into account any skewness or outliers that may have been present. Even with this extra step, the median income variable continued to be the best predictor and the population variable was the worst. However, it is still important to note that our best predictor could be even better if it weren't so skewed and didn't contain so many outliers.

4. Putting all predictors together in a multiple regression model – how well do these predictors taken together predict housing value? How does this full model compare to the model that just has the single best predictor from 3. ?

In [ ]:
```
from sklearn.metrics import r2_score


model = LinearRegression()
model.fit(df, outcome)
print("For all predictors")
print("Model Coefficients: ", model.coef_)

y_pred = model.predict(df)

# Calculate the R-squared value
r2 = r2_score(outcome, y_pred)

# Print the R-squared value
print('R-squared:', r2)
```

```
For all predictors
Model Coefficients:  [ 0.13759106  0.2737357   0.33283516 -2.56052514  1.55910
7    1.18968504
 -0.23233138]
R-squared: 0.6006645246293566
```

Answer 4

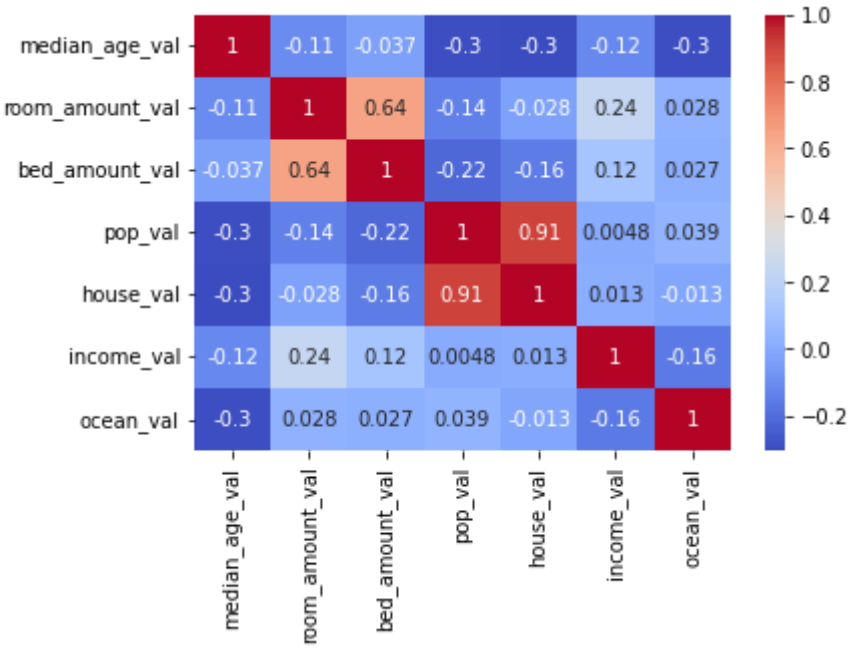
The multiple regression model is a significant improvement over individual linear regressions. This is likely due to the combination of strong predictors, such as proximity to the ocean and the normalized number of rooms, which significantly increase the R-squared of the model. When comparing the R^2 from the best simple linear regression and the R^2 of this model, the multiple regression model shows almost a 50% increase in performance. It is possible, however, that a simpler model with the 3 best performing individual predictors could outperform this model with all of the predictors. This could be due to issues with collinearity,

which is when two independent variables are highly correlated and thus create multicollinearity in the model. This is an important factor that should be taken into consideration when assessing the performance of a regression model. The next section of the research will discuss this further.

5. Considering the relationship between the (standardized) variables 2 and 3, is there potentially a concern regarding collinearity? Is there a similar concern regarding variables 4 and 5, if you were to include them in the model?

In [ ]:
```
import pandas as pd
import seaborn as sns

# Load the dataset into a pandas DataFrame

# Create a correlation matrix using the corr() method
corr_matrix = df.corr()

# Use Seaborn's heatmap function to visualize the correlation matrix
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
```

Out[ ]: <AxesSubplot:>



Answer 5

By understanding the meaning of variables 2 and 3, we can infer that there is a positive correlation between the amount of room and the number of bedrooms in a given block. To answer this question, I decided to test the possible multicolinearity present in these predictors. To do so, I created a correlation matrix of all predictors with each other and plotted it in a heatmap using the Seaborn library.

I chose to plot the correlation matrix as my strategy because it is a straightforward way of exposing the risks of colinearity, such as:

- Decreased predictive accuracy: Highly correlated predictor variables can lead to overfitting of the model, which can reduce its predictive accuracy. The model may fit the training data well but may not generalize well to new data.
- Unreliable coefficients: When predictor variables are highly correlated, it can be difficult to estimate the contribution of each variable to the outcome. This can lead to unstable or unreliable coefficient estimates, making it difficult to interpret the relationship between each predictor and the response variable.

Therefore, it is concerning to have both variables 2 and 3 in the same multiple regression model, since they present a correlation coefficient of 0.52, which is too high. The same applies to the population in a block and the number of households in a block. These two variables are even more correlated and may be causing our model to overfit to the training data.

Extra credit:

a) Does any of the variables (predictor or outcome) follow a distribution that can reasonablybe described as a normal distribution?

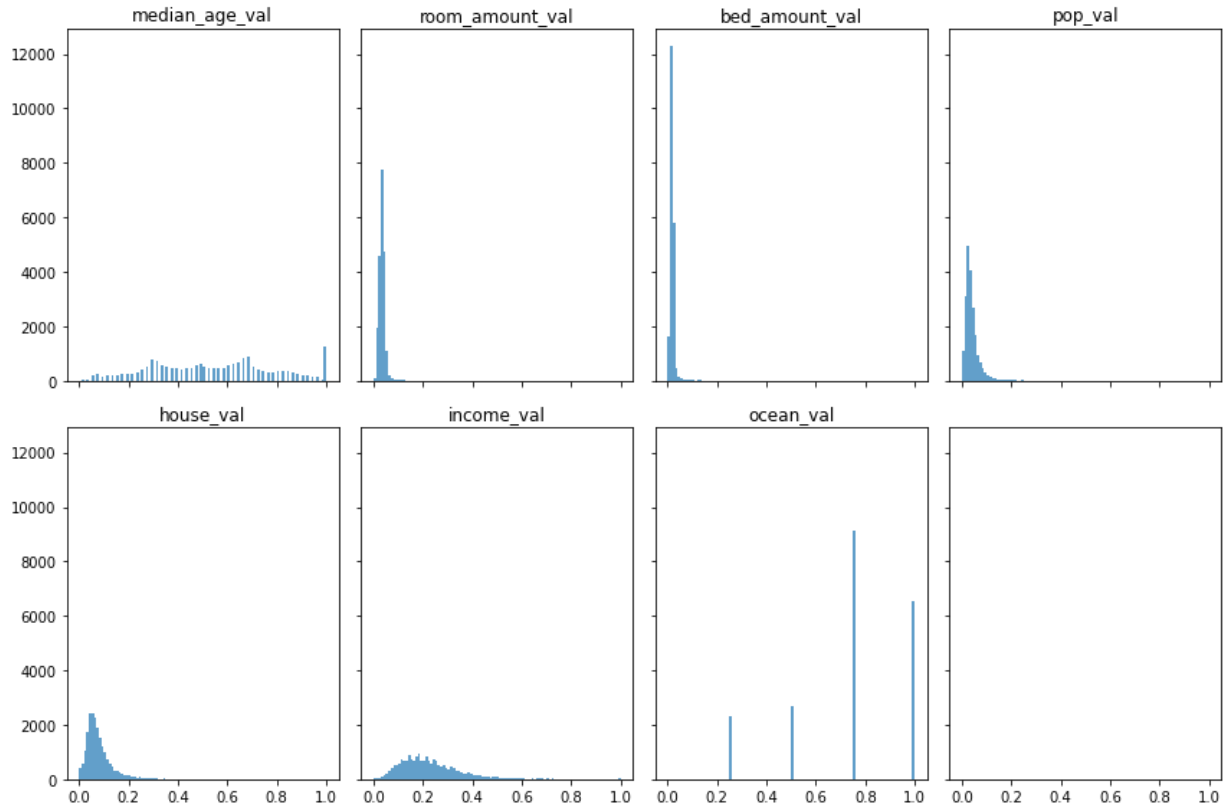
b) Examine the distribution of the outcome variable. Are there any characteristics of thisdistribution that might limit the validity of the conclusions when answering the questions above? If so, please comment on this characteristic.

Extra credit A)

Almost none of the predictor variables have a normal distribution, as is clearly visible from the plot. Moreover, the plot of the outcome variable which is further below, also shows that it does not follow a normal distribution. This observation can be seen from the shape of the plot and it is quite evident to the naked eye that it is not distributed in a normal manner. This is an important factor to take into consideration when predicting the outcome of a model, as it highlights the importance of selecting variables that are normally distributed.

In [ ]:
```
# Extra credit A


fig, axs = plt.subplots(nrows=2, ncols=4, figsize=(12, 8), sharex=True, share

# create histogram for each column
for i, col in enumerate(df.columns):
    ax = axs[i // 4, i % 4]
    ax.hist(df[col], alpha=0.7, stacked=True,bins=100)
    ax.set_title(col)

plt.tight_layout()
plt.show()

# # Display the plot
# plt.show()
# plt.figure()
# plt.figure(figsize=(20, 8))
# df['median_age_val'].plot.hist(bins=100,color = 'red',alpha=.5)
```

```
# df['room_amount_val'].plot.hist(bins=100,color = 'blue',alpha=.5)
# plt.show()
```

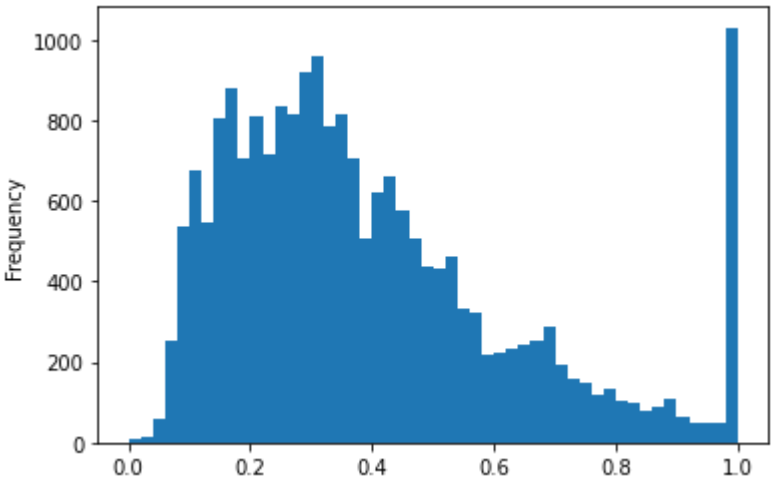


Extra Credit B)

After plotting the distribution of the outcome variable, we find that it is highly skewed to the left, exhibiting a large number of outliers. This is significant, as these outliers can have a profound effect on the regression model and lead to incorrect conclusions. Moreover, the skewed nature of the outcome suggests that it does not follow a normal distribution, and that the outcome does not have a linear relation to the predictor variables. As such, it is clear that fitting a linear regression model may not be the most effective approach for this dataset. Therefore, alternative methods should be considered to ensure the greatest accuracy of the model. This could include the use of transformations to the data, or using a nonlinear regression model. Whichever approach is chosen, it is essential to carefully consider the effect of outliers and the skewness of the data.

In [ ]:
```
# Extra Credit B
pd.Series(outcome).plot.hist(bins=50)
```

Out[ ]:
```
<AxesSubplot:ylabel='Frequency'>
```


Frequency
1000
800
600
400
200
0
0.0
0.2
0.4
0.6
0.8
1.0