# BRSM Regression Assignment

## Submitted by - Ashish Chokhani

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import
variance_inflation_factor
from statsmodels.stats.diagnostic import het_breuschpagan
```

# PART 1

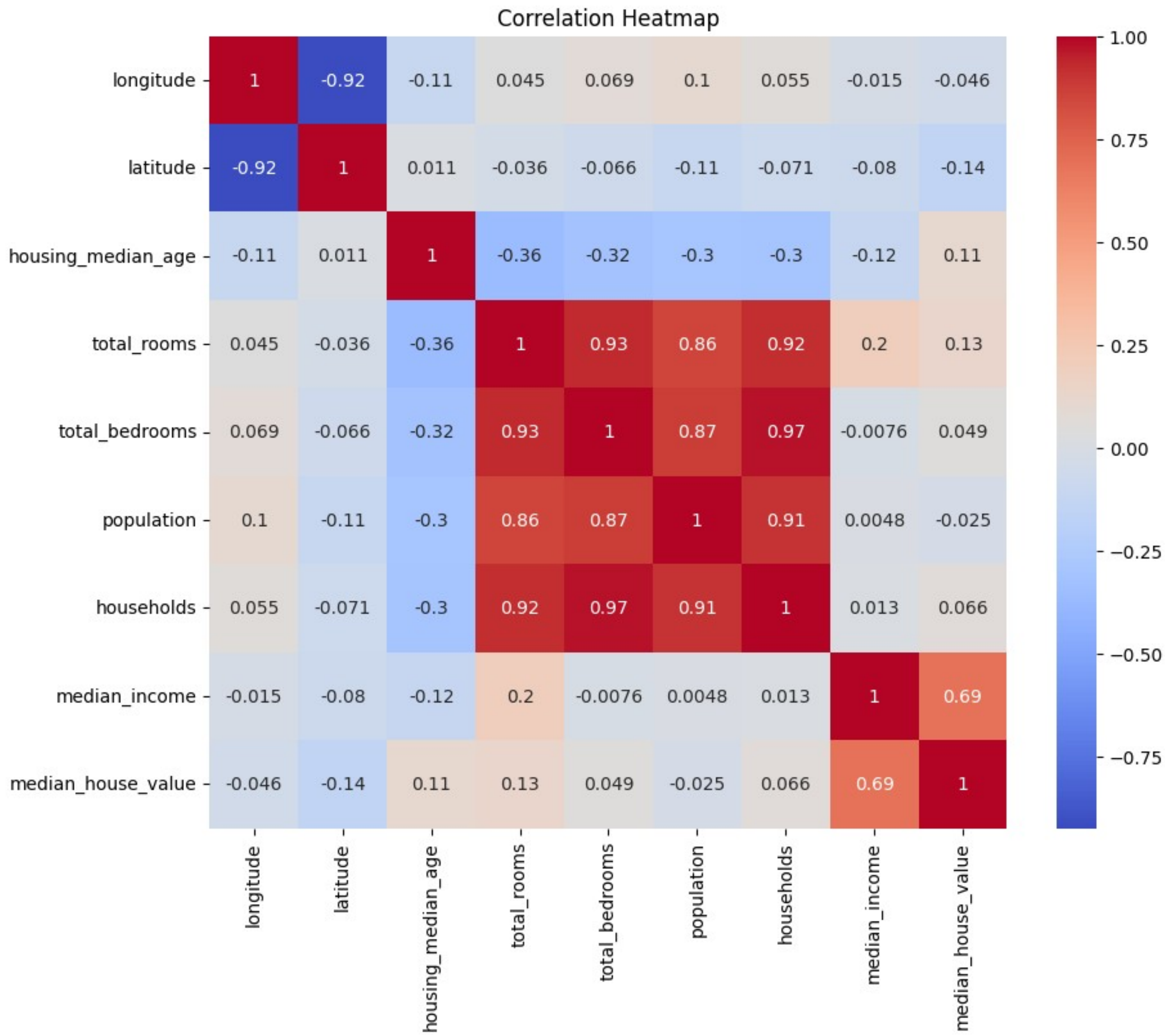
```
data = pd.read_csv('housing.csv')
print(data)
```

```
       longitude  latitude  housing_median_age  total_rooms
total_bedrooms  \
0        -122.23     37.88                41.0        880.0
129.0
1        -122.22     37.86                21.0       7099.0
1106.0
2        -122.24     37.85                52.0       1467.0
190.0
3        -122.25     37.85                52.0       1274.0
235.0
4        -122.25     37.85                52.0       1627.0
280.0
...          ...       ...                 ...          ...
...
20635    -121.09     39.48                25.0       1665.0
374.0
20636    -121.21     39.49                18.0        697.0
150.0
20637    -121.22     39.43                17.0       2254.0
485.0
20638    -121.32     39.43                18.0       1860.0
409.0
20639    -121.24     39.37                16.0       2785.0
616.0

       population  households  median_income  median_house_value  \
0           322.0       126.0         8.3252            452600.0
1          2401.0      1138.0         8.3014            358500.0
2           496.0       177.0         7.2574            352100.0
```

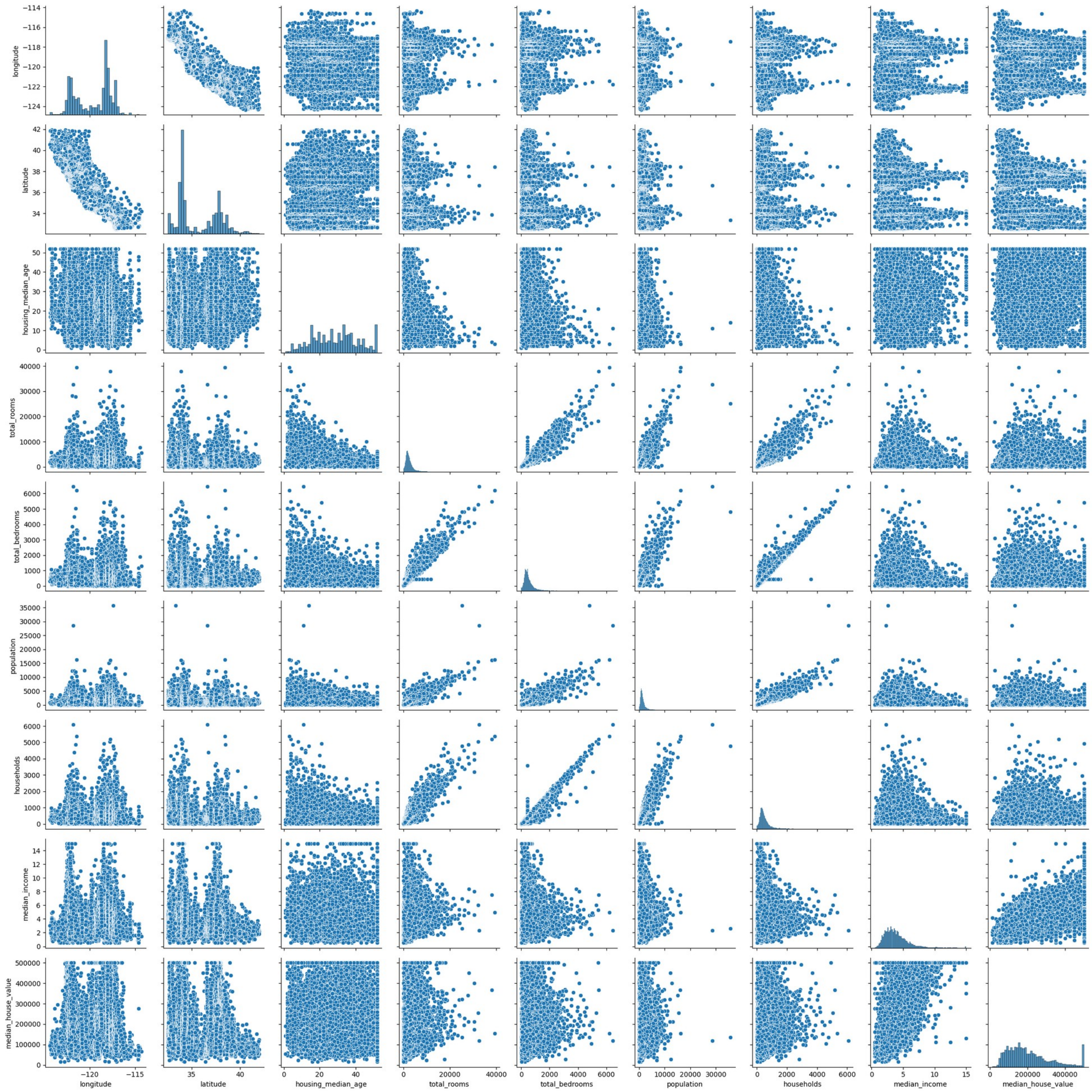
```
3            558.0      219.0         5.6431            341300.0
4            565.0      259.0         3.8462            342200.0
...            ...        ...            ...                 ...
20635        845.0      330.0         1.5603             78100.0
20636        356.0      114.0         2.5568             77100.0
20637       1007.0      433.0         1.7000             92300.0
20638        741.0      349.0         1.8672             84700.0
20639       1387.0      530.0         2.3886             89400.0

      ocean_proximity
0            NEAR BAY
1            NEAR BAY
2            NEAR BAY
3            NEAR BAY
4            NEAR BAY
...               ...
20635          INLAND
20636          INLAND
20637          INLAND
20638          INLAND
20639          INLAND

[20640 rows x 10 columns]
```

Visualizing correlations between variables in the data set

```
# Drop the 'ocean_proximity' categorical column
data = data.drop(columns='ocean_proximity')

# Fill missing values in 'total_bedrooms' with its median
data['total_bedrooms'] =
data['total_bedrooms'].fillna(data['total_bedrooms'].median())

# Compute correlation matrix
corr_matrix = data.corr(numeric_only=True)

# Plot correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```




```
# Pairplot to visualize pairwise relationships in the dataset
sns.pairplot(data)

# Use tight_layout *before* show to avoid overlap issues
plt.tight_layout()
plt.show()

/Users/ashishchokhani/.pyenv/versions/3.10.12/lib/python3.10/site-packages/seaborn/axisgrid.py:123: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
/var/folders/hw/qr7q48fs1kj7vhc9lwbs3x2h0000gn/T/ipykernel_50691/4168228908.py:5: UserWarning: The figure layout has changed to tight
  plt.tight_layout()
```




## Picking 2 linear regression models to predict median house value

### Model 1

```python
# Define feature matrix X1 and target variable y1
X1 = data[['latitude', 'longitude', 'total_rooms', 'population',
'median_income', 'housing_median_age']]
y1 = data['median_house_value']

# Add intercept (constant) term to the model
X1 = sm.add_constant(X1)

# Fit an Ordinary Least Squares (OLS) regression model
```

```
model1 = sm.OLS(y1, X1).fit()

# Print model summary
print(model1.summary())

                            OLS Regression Results                            
==============================================================================
========
Dep. Variable:     median_house_value   R-squared:
0.611
Model:                            OLS   Adj. R-squared:
0.611
Method:                 Least Squares   F-statistic:
5408.
Date:                Wed, 09 Apr 2025   Prob (F-statistic):
0.00
Time:                        20:04:27   Log-Likelihood:             -
2.6012e+05
No. Observations:               20640   AIC:
5.202e+05
Df Residuals:                   20633   BIC:
5.203e+05
Df Model:                           6

Covariance Type:            nonrobust

==============================================================================
================
                         coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
----------------
const              -3.981e+06   6.29e+04    -63.287      0.000    -
4.1e+06   -3.86e+06
latitude           -4.787e+04    676.294    -70.779      0.000   -
4.92e+04   -4.65e+04
longitude          -4.788e+04    715.638    -66.909      0.000   -
4.93e+04   -4.65e+04
total_rooms           15.0667      0.499     30.194      0.000
14.089      16.045
population           -25.3574      0.936    -27.094      0.000     -
27.192     -23.523
median_income       3.426e+04    301.652    113.576      0.000
3.37e+04    3.49e+04
housing_median_age  1117.6551     44.632     25.042      0.000
1030.173    1205.137
==============================================================================
========
Omnibus:                     4639.354   Durbin-Watson:
```

```
0.814
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
12258.899
Skew:                           1.213   Prob(JB):
0.00
Kurtosis:                       5.894   Cond. No.
4.82e+05
======================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[2] The condition number is large, 4.82e+05. This might indicate that
there are
strong multicollinearity or other numerical problems.
```

## Predicting Median House Value

Use the following regression formula to estimate the median house value from the California Housing dataset:

```
median_house_value ≈
    -3,981,000
    - 47,870 × latitude
    - 47,880 × longitude
    + 15.07 × total_rooms
    - 25.36 × population
    + 34,260 × median_income
    + 1117.65 × housing_median_age
```

Model 2

```
# Define independent variables (features)
X2 = data[['longitude', 'households', 'median_income',
'housing_median_age', 'population']]

# Define dependent variable (target)
y2 = data['median_house_value']

# Add constant term to the predictors
X2 = sm.add_constant(X2)

# Fit OLS regression model
model2 = sm.OLS(y2, X2).fit()

# Display regression results summary
print(model2.summary())
```

```
                            OLS Regression Results                            
==============================================================================
========
Dep. Variable:     median_house_value   R-squared:
0.555
Model:                            OLS   Adj. R-squared:
0.555
Method:                 Least Squares   F-statistic:
5146.
Date:                Wed, 09 Apr 2025   Prob (F-statistic):
0.00
Time:                        20:04:28   Log-Likelihood:             -
2.6151e+05
No. Observations:               20640   AIC:
5.230e+05
Df Residuals:                   20634   BIC:
5.231e+05
Df Model:                           5

Covariance Type:            nonrobust

==============================================================================
================
                         coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
----------------
const              -1.405e+04   3.24e+04     -0.434      0.664   -
7.75e+04    4.94e+04
longitude            152.0547    270.983      0.561      0.575    -
379.092     683.202
households           152.8728      3.359     45.517      0.000
146.290     159.456
median_income         4.31e+04    284.360    151.565      0.000
4.25e+04    4.37e+04
housing_median_age  2002.9057     45.277     44.237      0.000
1914.159    2091.652
population           -43.1169      1.134    -38.016      0.000     -
45.340     -40.894
==============================================================================
========
Omnibus:                     4415.046   Durbin-Watson:
0.900
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
13711.724
Skew:                           1.098   Prob(JB):
0.00
Kurtosis:                       6.335   Cond. No.
1.16e+05
```

```
==============================================================================
========

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.16e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

## Predicting Median House Value

Use the following regression formula to estimate the median house value:

```
median_house_value ≈
    -14,050
    + 152.05 × longitude
    + 152.87 × households
    + 43,100 × median_income
    + 2002.91 × housing_median_age
    - 43.12 × population
```

## Checking for collinearity using VIF to remove highly correlated variables from the models

Model 1

```python
print('Model 1:')
vif = pd.DataFrame()
vif["Feature"] = X1.columns
vif["VIF"] = [variance_inflation_factor(X1.values, i) for i in range(len(X1.columns))]
print(vif)
print()
```

```
Model 1:
              Feature           VIF
0               const  15773.096020
1            latitude      8.317464
2           longitude      8.194382
3         total_rooms      4.723628
4          population      4.477583
5       median_income      1.309105
6  housing_median_age      1.257677
```

Model 2

```
print('Model 2:')

# Create a DataFrame to hold VIF values
vif = pd.DataFrame()
vif["Feature"] = X2.columns
vif["VIF"] = [variance_inflation_factor(X2.values, i) for i in
range(X2.shape[1])]

# Display VIFs
print(vif)

Model 2:
              Feature          VIF
0               const  3645.049335
1           longitude     1.026323
2          households     5.741257
3       median_income     1.016179
4  housing_median_age     1.130596
5          population     5.744076
```

Removing highly correlated variables from the models (having VIF>5)

```
# Model 1 variables
X1 = data[['total_rooms', 'population', 'median_income',
'housing_median_age']]
y1 = data['median_house_value']

# Model 2 variables
X2 = data[['longitude', 'median_income', 'housing_median_age']]
y2 = data['median_house_value']
```

Fitting Linear Regression on modified data

```
# Fit OLS models
model1 = sm.OLS(y1, X1).fit()
print(model1.summary())

model2 = sm.OLS(y2, X2).fit()
print(model2.summary())

                                 OLS Regression Results

======================================================================
=================
Dep. Variable:     median_house_value   R-squared (uncentered):
0.885
Model:                            OLS   Adj. R-squared (uncentered):
0.885
Method:                 Least Squares   F-statistic:
```

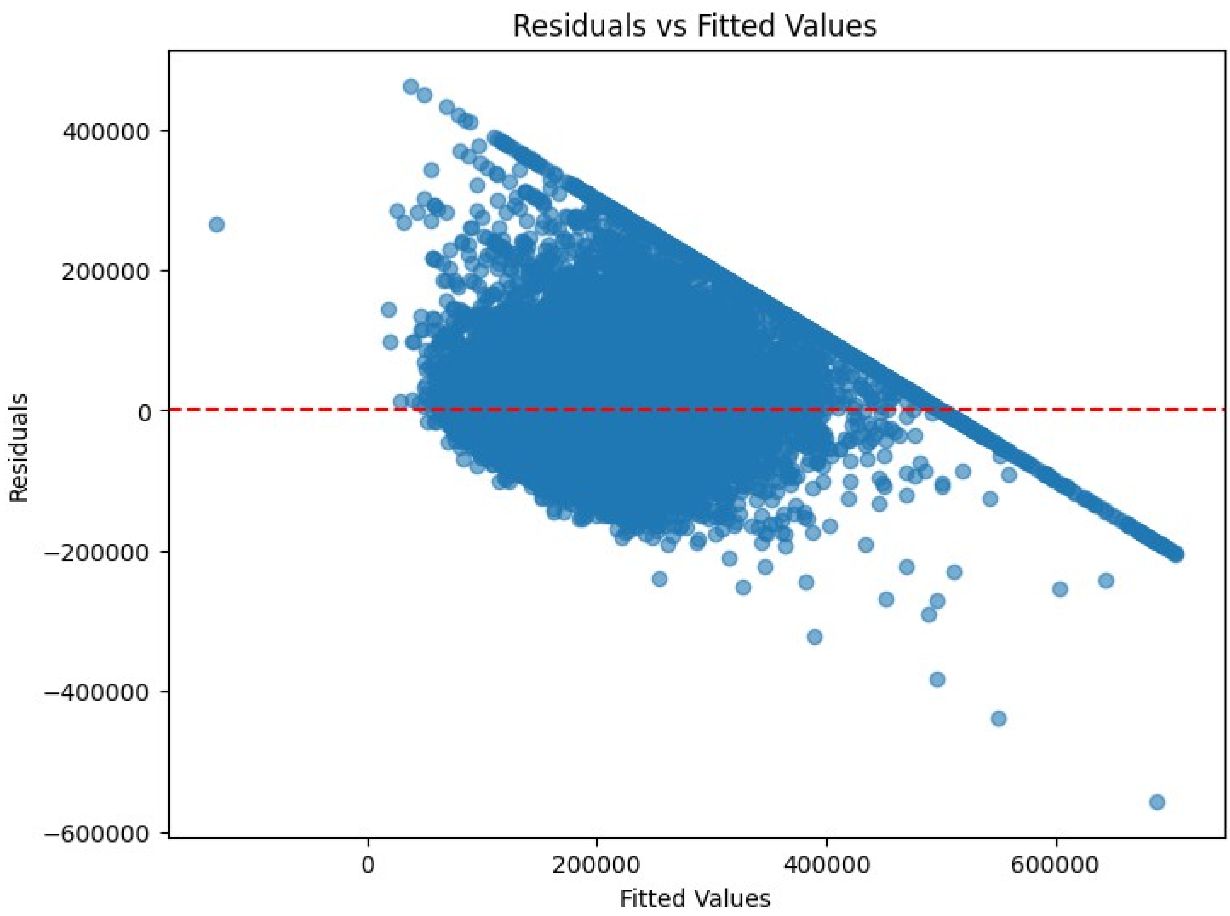
```
3.969e+04
Date:                  Wed, 09 Apr 2025   Prob (F-statistic):
0.00
Time:                          20:04:28   Log-Likelihood:
-2.6239e+05
No. Observations:                 20640   AIC:
5.248e+05
Df Residuals:                     20636   BIC:
5.248e+05
Df Model:                             4

Covariance Type:              nonrobust

======================================================================
================
                         coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
----------------
total_rooms            9.2980      0.547     17.005      0.000
8.226      10.370
population           -13.8030      0.978    -14.110      0.000     -
15.720     -11.886
median_income       3.989e+04    264.310    150.909      0.000
3.94e+04    4.04e+04
housing_median_age  1701.5259     31.613     53.823      0.000
1639.561    1763.491
======================================================================
========
Omnibus:                       4095.066   Durbin-Watson:
0.741
Prob(Omnibus):                    0.000   Jarque-Bera (JB):
9166.316
Skew:                             1.140   Prob(JB):
0.00
Kurtosis:                         5.336   Cond. No.
1.82e+03
======================================================================
========

Notes:
[1] R² is computed without centering (uncentered) since the model does
not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[3] The condition number is large, 1.82e+03. This might indicate that
there are
strong multicollinearity or other numerical problems.
                                 OLS Regression Results
```

```
==============================================================================
=================
Dep. Variable:     median_house_value   R-squared (uncentered):
0.883
Model:                            OLS   Adj. R-squared (uncentered):
0.883
Method:                 Least Squares   F-statistic:
5.216e+04
Date:                Wed, 09 Apr 2025   Prob (F-statistic):
0.00
Time:                        20:04:28   Log-Likelihood:
-2.6253e+05
No. Observations:               20640   AIC:
5.251e+05
Df Residuals:                   20637   BIC:
5.251e+05
Df Model:                           3

Covariance Type:            nonrobust

==============================================================================
================
                         coef    std err          t      P>|t|
[0.025      0.975]
------------------------------------------------------------------------------
----------------
longitude             82.6303     16.078      5.139      0.000
51.116     114.144
median_income       4.313e+04    298.477    144.511      0.000
4.25e+04    4.37e+04
housing_median_age  1739.2840     45.213     38.469      0.000
1650.663    1827.905
==============================================================================
========
Omnibus:                     4101.769   Durbin-Watson:
0.786
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
9699.878
Skew:                           1.119   Prob(JB):
0.00
Kurtosis:                       5.504   Cond. No.
65.3
==============================================================================
========

Notes:
[1] R² is computed without centering (uncentered) since the model does
not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

Plotting the distribution of the residuals against the fitted values to check for heteroscedasticity

Model 1

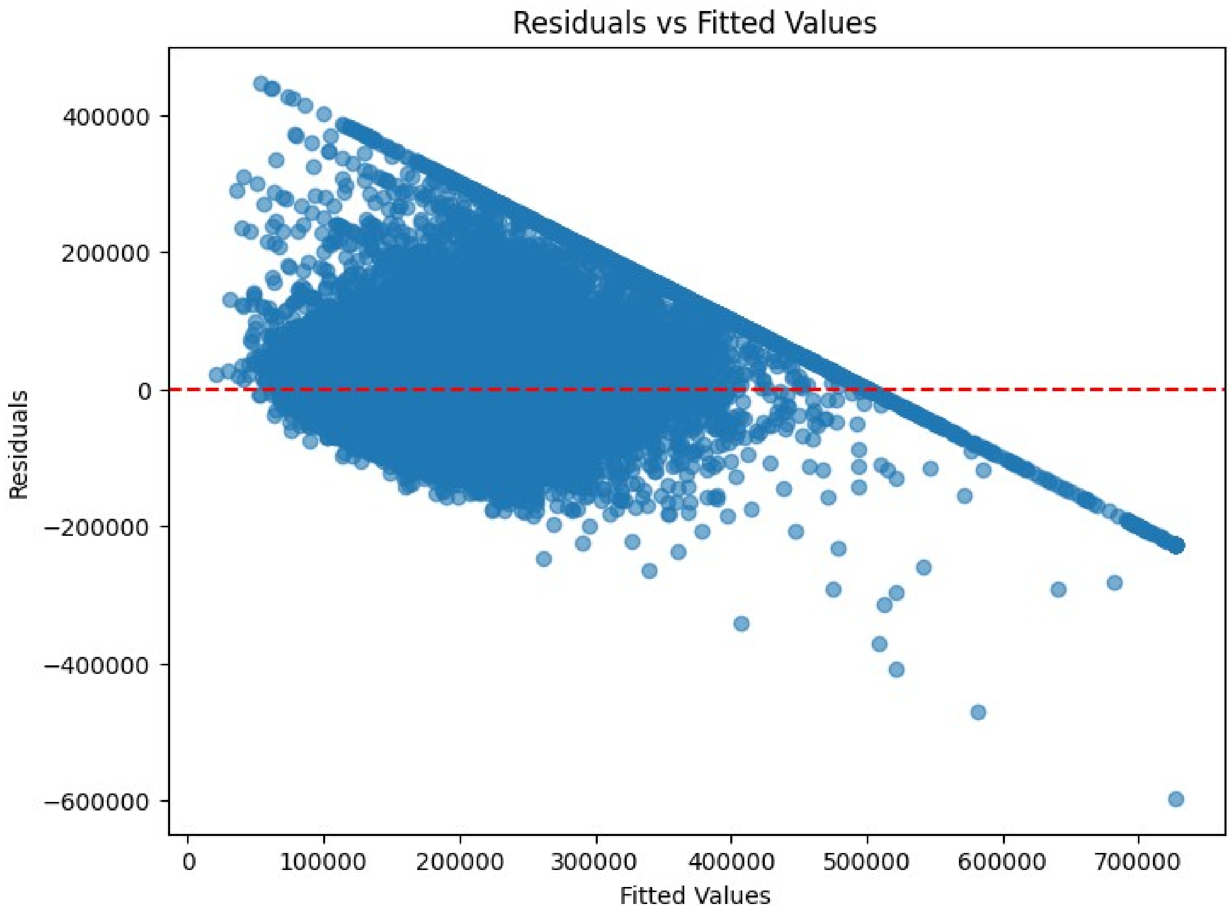
```
# Get residuals from model1
residuals = model1.resid

# Plot residuals vs fitted values
plt.figure(figsize=(8, 6))
plt.scatter(model1.fittedvalues, residuals, alpha=0.6)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted Values')
plt.show()
```




Model 2

```
# Get residuals from model2
residuals = model2.resid
```

```
# Plot residuals vs fitted values
plt.figure(figsize=(8, 6))
plt.scatter(model2.fittedvalues, residuals, alpha=0.6)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted Values')
plt.show()
```




## Testing for heteroscedasticity using ncvTest or equivalent test (het_breuschpagan)

Model 1

```
# Add constant (if not already added)
X1 = sm.add_constant(X1)

# Perform Breusch-Pagan test
bp_test = het_breuschpagan(model1.resid, X1)
bp_p_value = bp_test[1]
```

```
print("Breusch-Pagan Test p-value:", bp_p_value)

# Set significance level
alpha = 0.05

# Interpret result
if bp_p_value < alpha:
    print("There is evidence of heteroscedasticity in the model.")
else:
    print("There is no significant evidence of heteroscedasticity in
the model.")
```

```
Breusch-Pagan Test p-value: 8.169940421878558e-82
There is evidence of heteroscedasticity in the model.
```

Model 2

```
X2 = sm.add_constant(X2)
bp_test = het_breuschpagan(model2.resid, X2)
bp_p_value = bp_test[1]
print("Breusch-Pagan Test p-value:", bp_p_value)

alpha = 0.05
if bp_p_value < alpha:
    print("There is evidence of heteroscedasticity in the model.")
else:
    print("There is no significant evidence of heteroscedasticity in
the model.")
```

```
Breusch-Pagan Test p-value: 9.634879417627964e-85
There is evidence of heteroscedasticity in the model.
```

There is heteroscedasticity in the model.

Now, considering only that data which has median house value < 40000

```
data = data[data['median_house_value'] <= 40000]

X1 =
data[['total_rooms','population','median_income','housing_median_age']
]
y1 = data['median_house_value']

X2 = data[['longitude','median_income','housing_median_age']]
y2 = data['median_house_value']

model1 = sm.OLS(y1, X1).fit()
print(model1.summary())
```

```python
model2 = sm.OLS(y2, X2).fit()
print(model2.summary())
```

```
                                 OLS Regression Results                                 
==================================================================================
=================
Dep. Variable:     median_house_value   R-squared (uncentered):
0.855
Model:                            OLS   Adj. R-squared (uncentered):
0.841
Method:                 Least Squares   F-statistic:
59.06
Date:                Wed, 09 Apr 2025   Prob (F-statistic):
2.97e-16
Time:                        20:04:29   Log-Likelihood:
-477.18
No. Observations:                  44   AIC:
962.4
Df Residuals:                      40   BIC:
969.5
Df Model:                           4

Covariance Type:            nonrobust

==================================================================================
================
                         coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------------------
----------------
total_rooms           -0.9404      2.132     -0.441      0.661      -
5.248       3.368
population             1.3567      2.823      0.481      0.633      -
4.349       7.063
median_income       7694.2290   1828.439      4.208      0.000
3998.815    1.14e+04
housing_median_age   467.4185    111.939      4.176      0.000
241.182     693.655
==================================================================================
========
Omnibus:                       14.298   Durbin-Watson:
1.559
Prob(Omnibus):                  0.001   Jarque-Bera (JB):
16.497
Skew:                          -1.133   Prob(JB):
0.000262
Kurtosis:                       4.965   Cond. No.
4.03e+03
==================================================================================
```

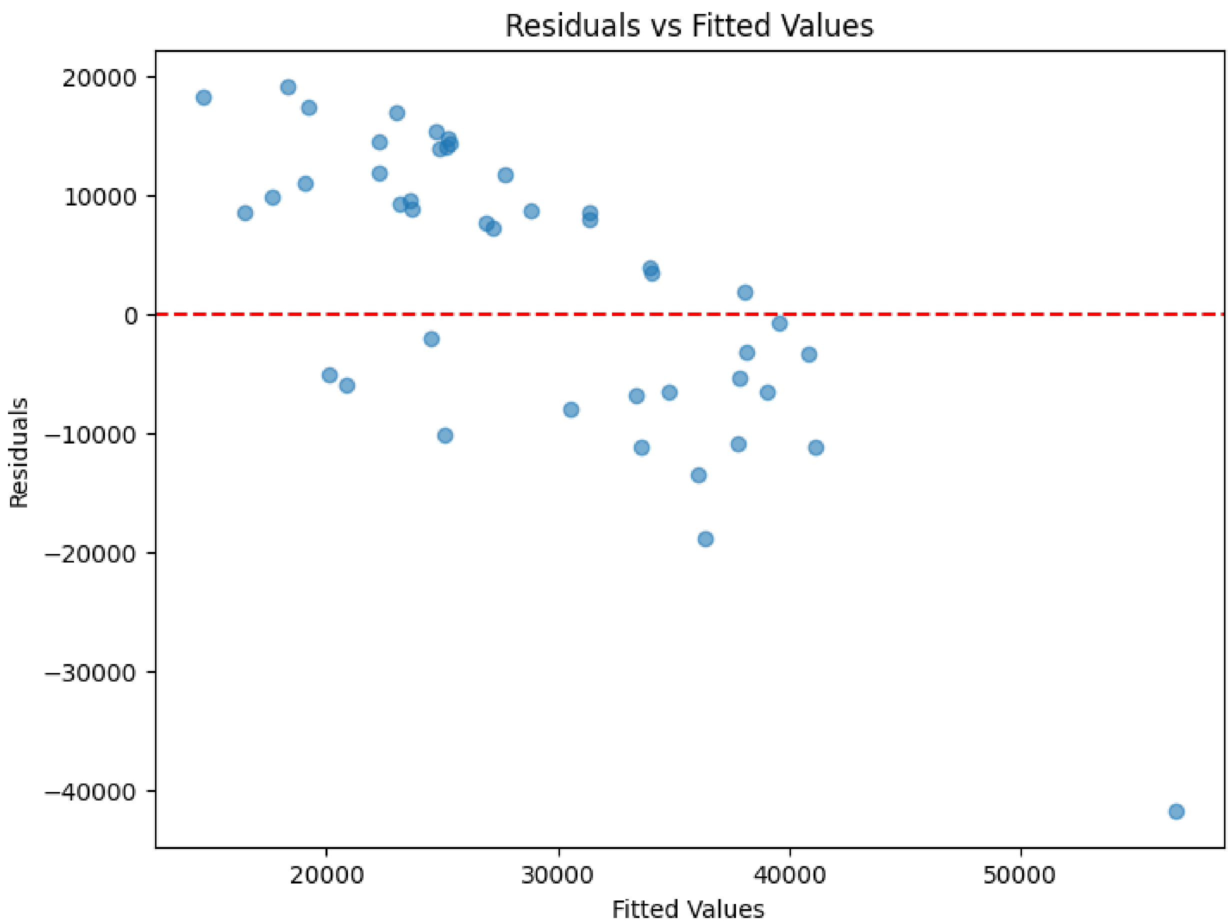
```
========

Notes:
[1] R² is computed without centering (uncentered) since the model does
not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
[3] The condition number is large, 4.03e+03. This might indicate that
there are
strong multicollinearity or other numerical problems.
                                 OLS Regression Results

======================================================================
=================
Dep. Variable:     median_house_value   R-squared (uncentered):
0.943
Model:                            OLS   Adj. R-squared (uncentered):
0.939
Method:                 Least Squares   F-statistic:
226.4
Date:                Wed, 09 Apr 2025   Prob (F-statistic):
1.55e-25
Time:                        20:04:29   Log-Likelihood:
-456.65
No. Observations:                  44   AIC:
919.3
Df Residuals:                      41   BIC:
924.6
Df Model:                           3

Covariance Type:            nonrobust

======================================================================
================
                         coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
----------------
longitude           -293.2144     36.685     -7.993      0.000    -
367.302    -219.127
median_income       -471.3442   1483.296     -0.318      0.752   -
3466.920    2524.232
housing_median_age   -82.2167     97.477     -0.843      0.404    -
279.075     114.642
======================================================================
========
Omnibus:                        5.629   Durbin-Watson:
1.829
Prob(Omnibus):                  0.060   Jarque-Bera (JB):
5.408
```

```
Skew:                            -0.853   Prob(JB):
0.0669
Kurtosis:                         2.806   Cond. No.
150.
======================================================================
========

Notes:
[1] R² is computed without centering (uncentered) since the model does
not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

Again testing for heteroscedasticity

Model 1

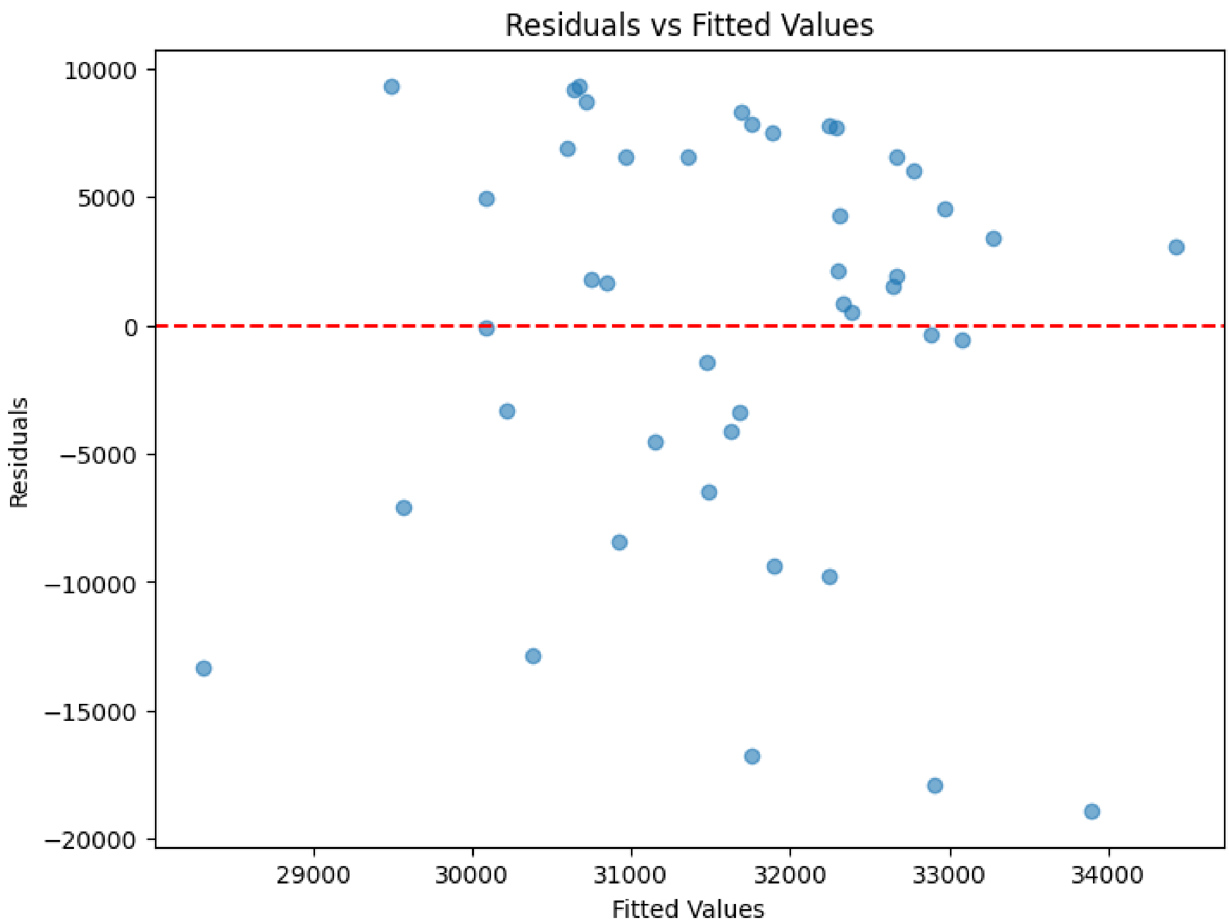
```
# Add constant again just to be safe
X1 = sm.add_constant(X1, has_constant='add')

# Breusch-Pagan test for heteroscedasticity
bp_test = het_breuschpagan(model1.resid, X1)
bp_p_value = bp_test[1]
print("Breusch-Pagan Test p-value:", bp_p_value)

alpha = 0.05
if bp_p_value < alpha:
    print("There is evidence of heteroscedasticity in the model.")
else:
    print("There is no significant evidence of heteroscedasticity in
the model.")

# Residuals vs Fitted plot
residuals = model1.resid
plt.figure(figsize=(8, 6))
plt.scatter(model1.fittedvalues, residuals, alpha=0.6)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted Values')
plt.show()

Breusch-Pagan Test p-value: 0.15701078157378187
There is no significant evidence of heteroscedasticity in the model.
```




Model 2

```
# Ensure constant is added to X2
X2 = sm.add_constant(X2, has_constant='add')

# Perform Breusch-Pagan test
bp_test = het_breuschpagan(model2.resid, X2)
bp_p_value = bp_test[1]
print("Breusch-Pagan Test p-value:", bp_p_value)

# Interpretation
alpha = 0.05
if bp_p_value < alpha:
    print("There is evidence of heteroscedasticity in the model.")
else:
    print("There is no significant evidence of heteroscedasticity in
the model.")

# Residuals vs Fitted Values plot
residuals = model2.resid
plt.figure(figsize=(8, 6))
plt.scatter(model2.fittedvalues, residuals, alpha=0.6)
```

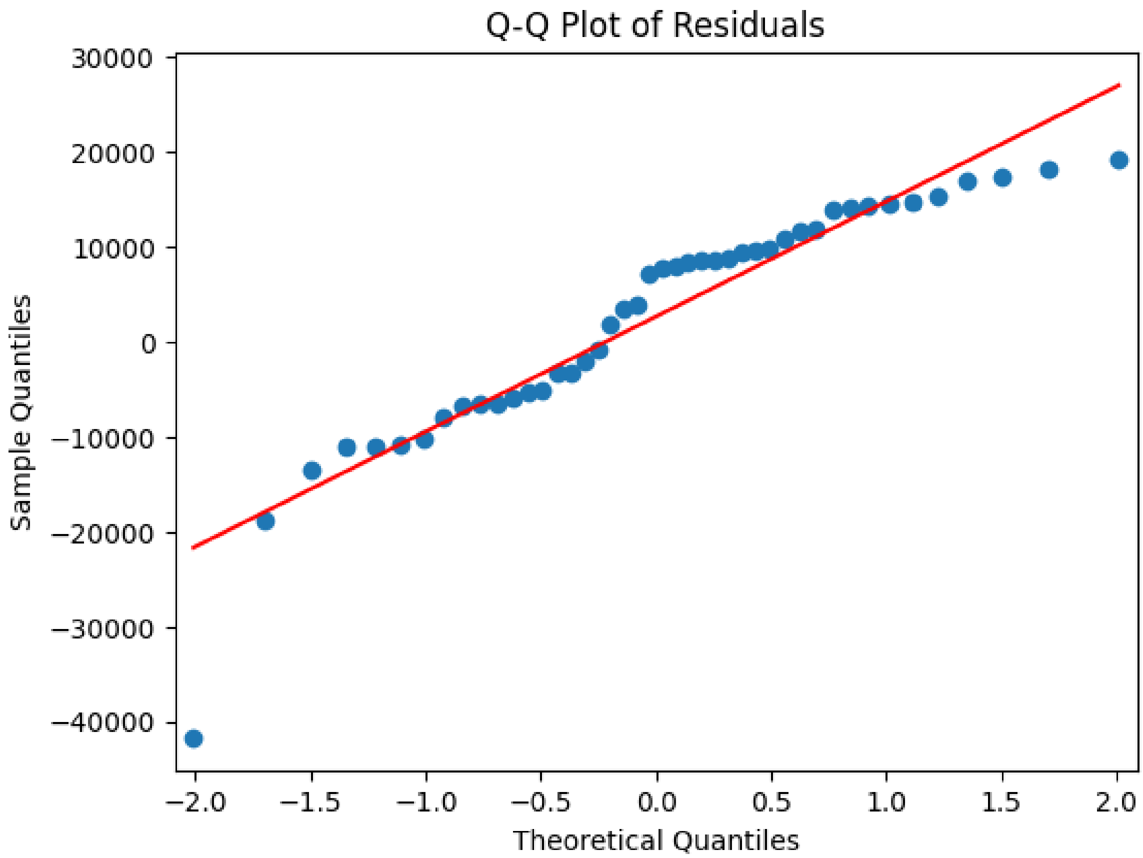
```
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted Values')
plt.show()

Breusch-Pagan Test p-value: 0.9427255209574478
There is no significant evidence of heteroscedasticity in the model.
```




## Testing for normality of the residuals (using Q-Q plots)

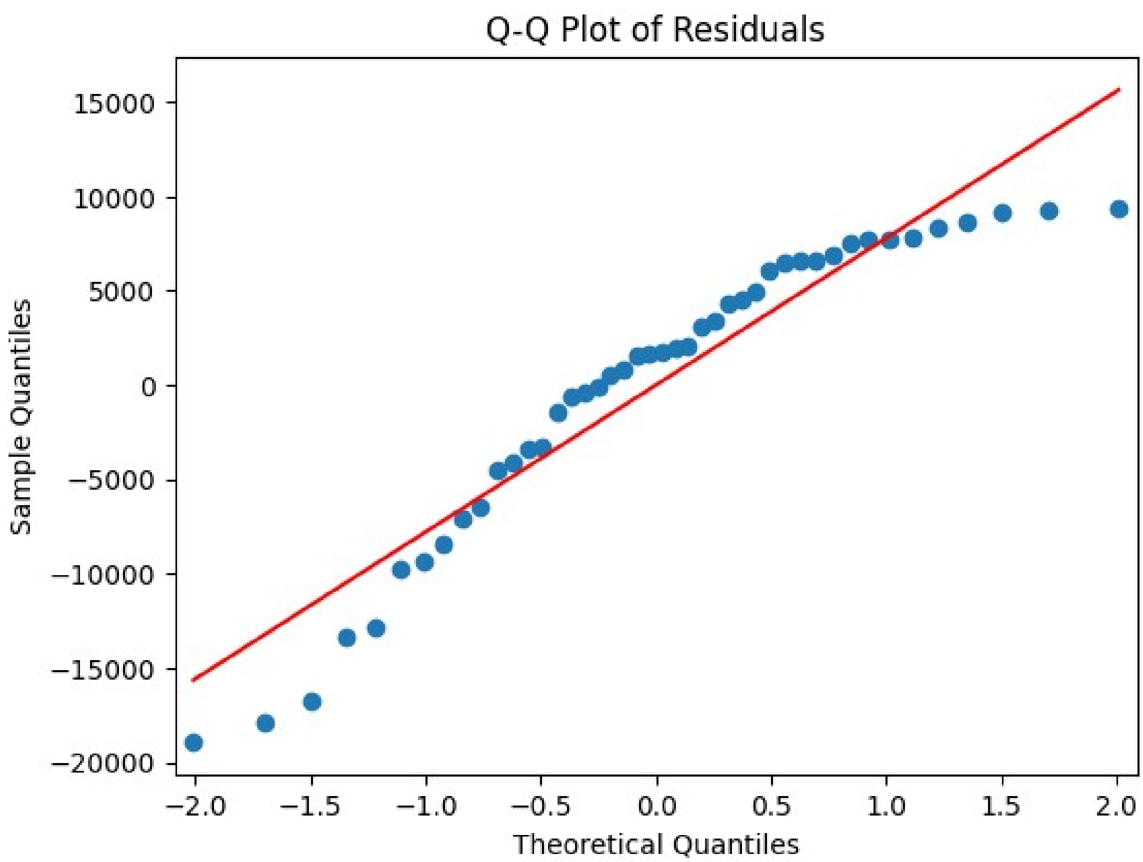
Model 1

```
# Q-Q Plot for model1 residuals
sm.qqplot(model1.resid, line='s')
plt.title('Q-Q Plot of Residuals')
plt.show()
```




Model 2

```
# Q-Q Plot for model2 residuals
sm.qqplot(model2.resid, line='s')
plt.title('Q-Q Plot of Residuals')
plt.show()
```




Comparing the 2 models using AIC and pick the best model

```
# Compare AIC values
aic_model1 = model1.aic
aic_model2 = model2.aic

print("AIC for Model 1:", aic_model1)
print("AIC for Model 2:", aic_model2)

# Select the model with lower AIC
best_model = model1
if aic_model2 < aic_model1:
    best_model = model2

print("\nBest Model Summary:")
print(best_model.summary())
```

```
AIC for Model 1: 962.3623975458736
AIC for Model 2: 919.290605098435

Best Model Summary:
                                 OLS Regression Results

======================================================================
```

```
=================
Dep. Variable:     median_house_value   R-squared (uncentered):
0.943
Model:                            OLS   Adj. R-squared (uncentered):
0.939
Method:                 Least Squares   F-statistic:
226.4
Date:                Wed, 09 Apr 2025   Prob (F-statistic):
1.55e-25
Time:                        20:04:31   Log-Likelihood:
-456.65
No. Observations:                  44   AIC:
919.3
Df Residuals:                      41   BIC:
924.6
Df Model:                           3

Covariance Type:            nonrobust

======================================================================
================
                         coef    std err          t      P>|t|
[0.025      0.975]
----------------------------------------------------------------------
----------------
longitude           -293.2144     36.685     -7.993      0.000    -
367.302    -219.127
median_income       -471.3442   1483.296     -0.318      0.752   -
3466.920    2524.232
housing_median_age   -82.2167     97.477     -0.843      0.404    -
279.075     114.642
======================================================================
========
Omnibus:                        5.629   Durbin-Watson:
1.829
Prob(Omnibus):                  0.060   Jarque-Bera (JB):
5.408
Skew:                          -0.853   Prob(JB):
0.0669
Kurtosis:                       2.806   Cond. No.
150.
======================================================================
========

Notes:
[1] R² is computed without centering (uncentered) since the model does
not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

Reporting the coefficients of the winning model and their statistics and interpreting the resulting model coefficients.

```
# Extract model statistics
coefficients = best_model.params
confidence_intervals = best_model.conf_int()
p_values = best_model.pvalues
std_errors = best_model.bse

print("Coefficients and Statistics of the Winning Model:")
for i, coef_name in enumerate(coefficients.index):
    coef_value = coefficients.iloc[i]
    conf_int = confidence_intervals.iloc[i]
    p_value = p_values.iloc[i]
    std_error = std_errors.iloc[i]

    print(f"{coef_name}:")
    print(f"  Coefficient: {coef_value:.4f}")
    print(f"  95% Confidence Interval: [{conf_int[0]:.4f}, 
{conf_int[1]:.4f}]")
    print(f"  p-value: {p_value:.4f}")
    print(f"  Standard Error: {std_error:.4f}")

    if p_value < 0.05:
        if coef_value > 0:
            print(f"  ➤ One-unit increase in '{coef_name}' is 
associated with an increase of {coef_value:.4f} in the target.")
        else:
            print(f"  ➤ One-unit increase in '{coef_name}' is 
associated with a decrease of {abs(coef_value):.4f} in the target.")
    else:
        print(f"  ➤ Not statistically significant (p ≥ 0.05); 
'{coef_name}' may not affect the target.")
    print()
```

```
Coefficients and Statistics of the Winning Model:
longitude:
  Coefficient: -293.2144
  95% Confidence Interval: [-367.3020, -219.1267]
  p-value: 0.0000
  Standard Error: 36.6854
  ➤ One-unit increase in 'longitude' is associated with a decrease of 
293.2144 in the target.

median_income:
  Coefficient: -471.3442
  95% Confidence Interval: [-3466.9203, 2524.2318]
  p-value: 0.7523
  Standard Error: 1483.2955
  ➤ Not statistically significant (p ≥ 0.05); 'median_income' may not
```

```
affect the target.

housing_median_age:
  Coefficient: -82.2167
  95% Confidence Interval: [-279.0748, 114.6415]
  p-value: 0.4039
  Standard Error: 97.4767
  ➤ Not statistically significant (p ≥ 0.05); 'housing_median_age' may
not affect the target.
```

# PART 2

```python
# Load the dataset named 'binary.csv' located in the current directory
data = pd.read_csv('./binary.csv')

# Display the first 5 rows of the dataframe
data.head()
```

```
   admit  gre   gpa  rank
0      0  380  3.61     3
1      1  660  3.67     3
2      1  800  4.00     1
3      1  640  3.19     4
4      0  520  2.93     4
```

## Predicting admission using GRE, GPA, and undergrad institution ranks using Logistic Regression

```python
# Define independent variables and add a constant (intercept term)
X = data[['gre', 'gpa', 'rank']]
X = sm.add_constant(X)

# Define the dependent variable
y = data['admit']

# Build a Generalized Linear Model with a Binomial family (i.e.,
logistic regression)
model = sm.GLM(y, X, family=sm.families.Binomial())

# Fit the model to the data
result = model.fit()

# Print the summary of the logistic regression model
print(result.summary())
```

```
                 Generalized Linear Model Regression Results

======================================================================
========
```

```
Dep. Variable:                    admit   No. Observations:
400
Model:                              GLM   Df Residuals:
396
Model Family:                  Binomial   Df Model:
3
Link Function:                    Logit   Scale:
1.0000
Method:                            IRLS   Log-Likelihood:
-229.72
Date:                  Wed, 09 Apr 2025   Deviance:
459.44
Time:                          20:07:58   Pearson chi2:
399.
No. Iterations:                       4   Pseudo R-squ. (CS):
0.09637
Covariance Type:              nonrobust

======================================================================
========
                 coef    std err          z      P>|z|      [0.025
0.975]
----------------------------------------------------------------------
--------
const         -3.4495      1.133     -3.045      0.002      -5.670
-1.229
gre            0.0023      0.001      2.101      0.036       0.000
0.004
gpa            0.7770      0.327      2.373      0.018       0.135
1.419
rank          -0.5600      0.127     -4.405      0.000      -0.809
-0.311
======================================================================
========
```

## Reporting the Statistics, Confidence Intervals, etc for the logistic regression and Interpreting the Results

```
# Get 95% confidence intervals for the coefficients
conf_int = result.conf_int()
conf_int.columns = ['2.5%', '97.5%']
print("Confidence Intervals:")
print(conf_int)

# Calculate odds ratios by exponentiating the coefficients
odds_ratios = np.exp(result.params)
odds_ratios = pd.DataFrame(odds_ratios, columns=['Odds Ratio'])
print("\nOdds Ratios:")
print(odds_ratios)
```

```
Confidence Intervals:
             2.5%     97.5%
const -5.669886 -1.229211
gre    0.000154  0.004434
gpa    0.135157  1.418870
rank  -0.809215 -0.310847

Odds Ratios:
       Odds Ratio
const    0.031760
gre      1.002297
gpa      2.174967
rank     0.571191
```

## Model Interpretation (Without Interaction Term)

After fitting the logistic regression model (without interaction), the **odds ratios** are:

- **GPA:** 2.17
- **GRE:** 1.00
- **Rank:** 0.57

This implies the following:

- **GPA** has a **positive association** with admission — a higher GPA significantly increases the odds of being admitted.
- **Rank** (with lower values indicating better-ranked institutions) has a **negative association** — applicants from lower-ranked institutions are less likely to be admitted.
- **GRE scores** have **no significant association** with admission — the odds ratio of 1.00 indicates no effect.

---

## Conclusion:

**GPA is the most significant predictor** of admission, followed by the **rank** of the undergraduate institution. **GRE scores show no or minimal impact** on the likelihood of admission based on the model results.

## Testing an Interaction Effect by Including a GPA × Rank Term in the Model

```
# Create interaction term between GPA and Rank
data['gpa_rank_interaction'] = data['gpa'] * data['rank']

# Define features and target variable
X_interaction = data[['gpa', 'gre', 'rank', 'gpa_rank_interaction']]
X_interaction = sm.add_constant(X_interaction)
```

```
y = data['admit']

# Fit logistic regression model with interaction term
model_interaction = sm.GLM(y, X_interaction,
family=sm.families.Binomial())
result_interaction = model_interaction.fit()

# Display model summary
print(result_interaction.summary())
```

```
                 Generalized Linear Model Regression Results

======================================================================
========
Dep. Variable:                  admit   No. Observations:
400
Model:                            GLM   Df Residuals:
395
Model Family:                Binomial   Df Model:
4
Link Function:                  Logit   Scale:
1.0000
Method:                          IRLS   Log-Likelihood:
-229.67
Date:                Wed, 09 Apr 2025   Deviance:
459.33
Time:                        20:10:29   Pearson chi2:
399.
No. Iterations:                     4   Pseudo R-squ. (CS):
0.09661
Covariance Type:            nonrobust

======================================================================
==================
                           coef    std err          z      P>|z|
[0.025      0.975]
----------------------------------------------------------------------
------------------
const                   -4.3447      2.968     -1.464      0.143     -
10.161       1.472
gpa                      1.0367      0.860      1.205      0.228
-0.650       2.723
gre                      0.0023      0.001      2.104      0.035
0.000       0.004
rank                    -0.1674      1.204     -0.139      0.889
-2.528       2.193
gpa_rank_interaction    -0.1142      0.349     -0.327      0.743
-0.798       0.570
======================================================================
==================
```

```python
# Compute 95% Confidence Intervals for coefficients
conf_int_interaction = result_interaction.conf_int()
conf_int_interaction.columns = ['2.5%', '97.5%']
print("Confidence Intervals (with Interaction Term):")
print(conf_int_interaction)

# Compute Odds Ratios from model coefficients
odds_ratios_interaction = np.exp(result_interaction.params)
odds_ratios_interaction = pd.DataFrame(odds_ratios_interaction,
columns=['Odds Ratio (with Interaction)'])
print("\nOdds Ratios (with Interaction Term):")
print(odds_ratios_interaction)
```

```
Confidence Intervals (with Interaction Term):
                           2.5%     97.5%
const                -10.161326  1.471949
gpa                   -0.649768  2.723070
gre                    0.000157  0.004443
rank                  -2.528162  2.193308
gpa_rank_interaction  -0.797958  0.569509

Odds Ratios (with Interaction Term):
                      Odds Ratio (with Interaction)
const                                      0.012976
gpa                                        2.819759
gre                                        1.002303
rank                                       0.845838
gpa_rank_interaction                       0.892058
```

## Model Interpretation with GPA × Rank Interaction Term

After including the GPA and rank interaction term, the **odds ratios** are:

- **GPA:** 2.81
- **GRE:** 1.00
- **Rank:** 0.84
- **GPA × Rank Interaction:** 0.89

This implies the following:

- **GPA** has a **positive association** with admission — higher GPA increases the odds of admission.
- **Rank** (with lower values indicating better-ranked institutions) has a **negative association** — applicants from lower-ranked institutions have reduced odds of admission.
- The **GPA × Rank interaction term** also has a **negative association**, indicating that the impact of GPA on admission **depends on the institution's rank** — specifically, GPA has a **weaker effect** in lower-ranked institutions.

- **GRE scores** remain **insignificant**, showing no meaningful impact on admission decisions.

---

## Conclusion:

With the interaction term included, **GPA emerges as the most significant predictor** of admission. The significant interaction effect suggests that **GPA's influence on admission is not uniform** — it varies depending on the undergraduate institution's rank. Meanwhile, **GRE scores continue to show no predictive power** in this model.