2021101113

April 12, 2024

Regression Assignment
Gowlapalli Rohit 2021101113

[67]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
from statsmodels.stats.diagnostic import het_breuschpagan
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
from scipy.stats import pearsonr
from scipy.stats import bartlett
from statsmodels.stats.diagnostic import het_breuschpagan
from statsmodels.compat import lzip
```

1 PART 1

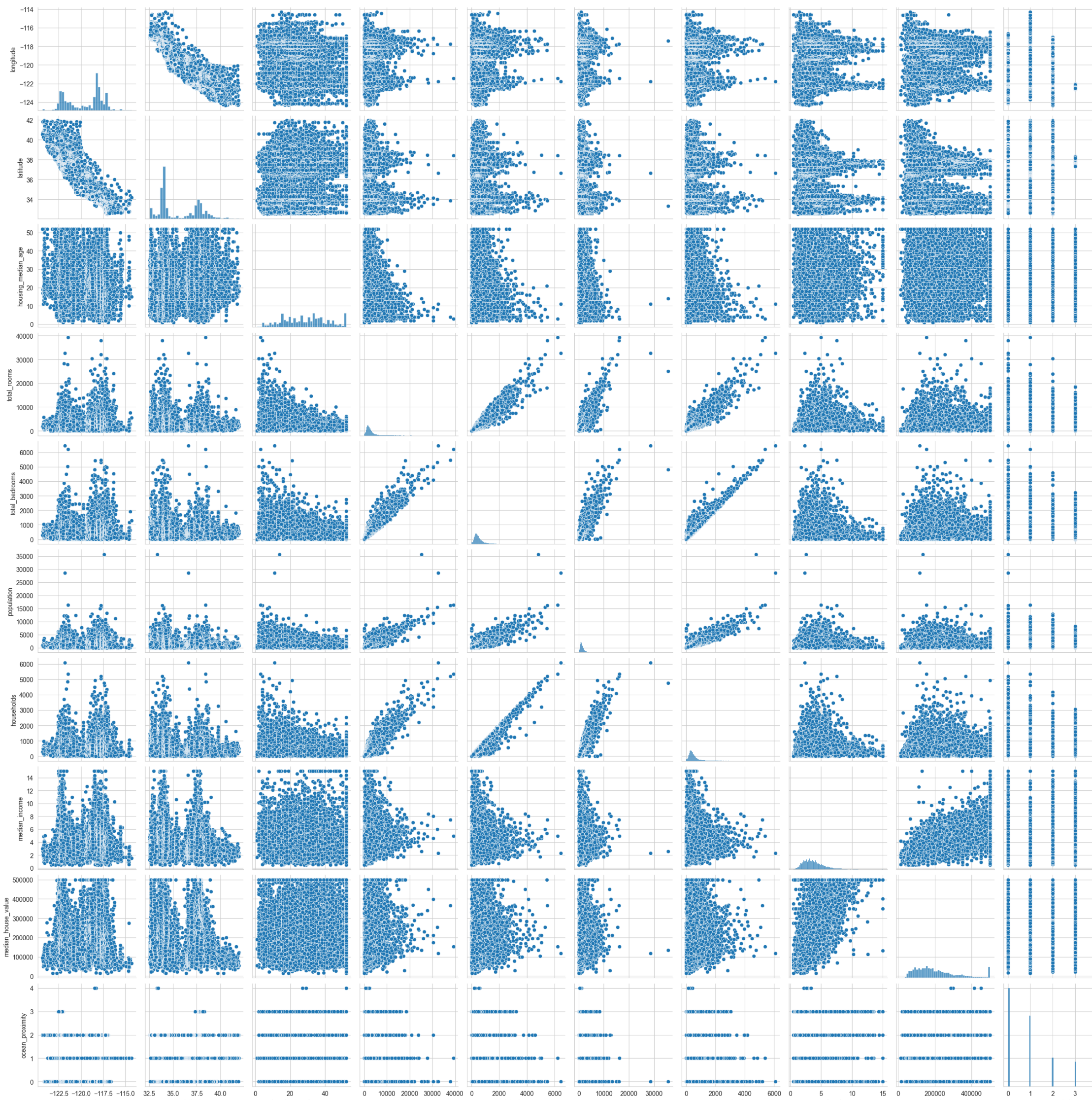
[68]:
```
df = pd.read_csv("housing.csv")
counts = df['ocean_proximity'].value_counts()
```

We can see that ocean_proximity is having string variables. Lets convert it to numericals before we perform the correlation analysis

[69]:
```
# cleaning the data by removing the nan values and changing data to numerical␣
↪variables
df['ocean_proximity'] = df['ocean_proximity'].map({'<1H OCEAN':0, 'INLAND':1,␣
↪'NEAR OCEAN':2, 'NEAR BAY':3, 'ISLAND':4})
df = df.dropna()
```

1.1 Visualize some correlations between variables in the data set

[70]:
```python
sns.pairplot(df.dropna())
plt.show()
plt.tight_layout()
```



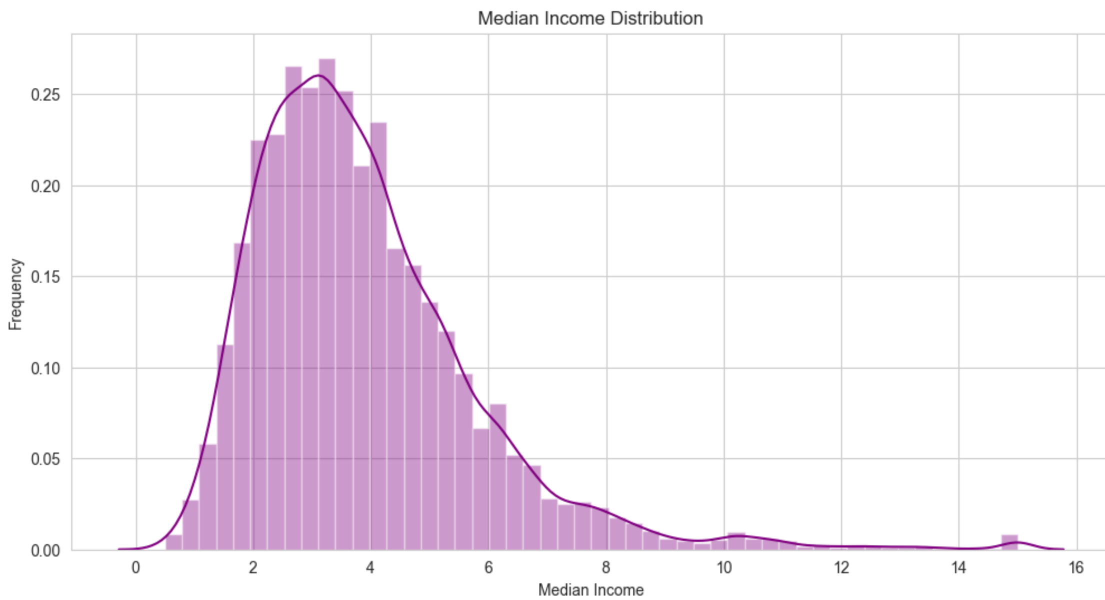

```
<Figure size 640x480 with 0 Axes>
```

[71]:
```python
numeric_cols = df.columns
num_rows = 5
num_cols = 2
fig, axes = plt.subplots(num_rows, num_cols, figsize=(15, 20))
axes = axes.flatten()
for i, col_name in enumerate(numeric_cols):
```

```python
    axes[i].hist(df[col_name], bins=50, color='purple')
    axes[i].set_title(f'Histogram of {col_name}')
    axes[i].set_xlabel(col_name)
    axes[i].set_ylabel('Frequency')

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

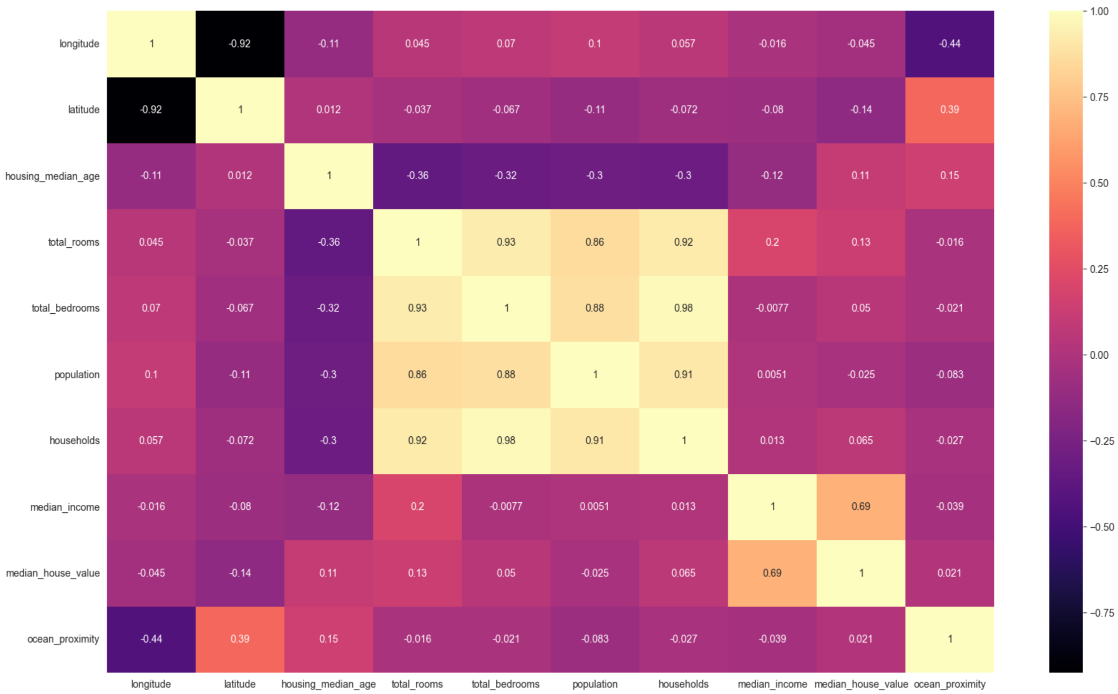
[72]:
```
df_1 = df['median_income']
df_1 = np.array(pd.DataFrame(df_1, columns=['median_income'])).reshape(-1, 1)
```

```
y = df['median_house_value']
df_2 = df.copy()
df_2 = df_2.drop('median_house_value', axis=1)

plt.figure(figsize=(12, 6))
sns.distplot(df['median_income'], bins=50, color='purple')
plt.title('Median Income Distribution')
plt.xlabel('Median Income')
plt.ylabel('Frequency')
plt.show()
```



[73]:
```
sns.set_style('whitegrid')
plt.figure(figsize=(20, 12))
sns.heatmap(df.corr(), annot=True, cmap='magma')
```

[73]: <Axes: >



We can clearly see some of the variables are highly correlated, now lets perform a correlation test to confirm the collinearity before building the model

[74]:
```
def correlation_test(data1, data2, alternative):
    corr, p_value = pearsonr(data1, data2)
    print("Correlation coefficient:", corr)
    print("p-value:", p_value)
    print("Alternative hypothesis:", alternative)
    if alternative == "greater":
        if p_value/2 < 0.05:
            print("Reject the null hypothesis: There is a positive correlation between the two variables")
        else:
            print("Fail to reject the null hypothesis: There is no positive correlation between the two variables")
    elif alternative == "less":
        if p_value/2 < 0.05:
            print("Reject the null hypothesis: There is a negative correlation between the two variables")
        else:
            print("Fail to reject the null hypothesis: There is no negative correlation between the two variables")
    else:
        if p_value < 0.05:
```

```
            print("Reject the null hypothesis: There is a correlation between␣
↪the two variables")
        else:
            print("Fail to reject the null hypothesis: There is no correlation␣
↪between the two variables")

print("Correlation test for total_bedrooms and total_rooms:")
correlation_test(df['total_bedrooms'], df['total_rooms'], alternative="greater")
print("\nCorrelation test for households and population:")
correlation_test(df['households'], df['population'], alternative="greater")
print("\nCorrelation test for longitude and latitude:")
correlation_test(df['longitude'], df['latitude'], alternative="less")
```

```
Correlation test for total_bedrooms and total_rooms:
Correlation coefficient: 0.9303795046865074
p-value: 0.0
Alternative hypothesis: greater
Reject the null hypothesis: There is a positive correlation between the two
variables

Correlation test for households and population:
Correlation coefficient: 0.907185900174492
p-value: 0.0
Alternative hypothesis: greater
Reject the null hypothesis: There is a positive correlation between the two
variables

Correlation test for longitude and latitude:
Correlation coefficient: -0.9246161131160016
p-value: 0.0
Alternative hypothesis: less
Reject the null hypothesis: There is a negative correlation between the two
variables
```

Based on the correlation tests conducted earlier, it's evident that whenever the p-value falls below 0.05, indicating a significant correlation, utilizing just one of the variables from the correlated pair is adequate for model construction.

We constructed three linear regression models by selecting only one variable from each highly correlated pair, effectively reducing the dimensions by three in each model. In the third model, we employed only two variables with notably high absolute correlation values. Notably, in all cases, the p-value was below 0.05, indicating a strong fit of the model to the data.

1.2 Pick 2 linear regression models to predict median house value

1.2.1 Method 1 : Model 1 - Linear Regression

[75]:
```
f1 = 'median_house_value ~ longitude + housing_median_age + total_rooms +␣
 ↪households + median_income + ocean_proximity'
model = sm.formula.ols(formula=f1, data=df)
result = model.fit()
r1 = result
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     median_house_value   R-squared:                       0.538
Model:                            OLS   Adj. R-squared:                  0.538
Method:                 Least Squares   F-statistic:                     3968.
Date:                Fri, 12 Apr 2024   Prob (F-statistic):               0.00
Time:                        22:37:27   Log-Likelihood:            -2.5928e+05
No. Observations:               20433   AIC:                         5.186e+05
Df Residuals:                   20426   BIC:                         5.186e+05
Df Model:                           6
Covariance Type:            nonrobust
================================================================================
======
                         coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------
Intercept          -1.135e+05   3.63e+04     -3.124      0.002   -1.85e+05
-4.23e+04
longitude           -547.0816    305.893     -1.788      0.074   -1146.656
52.493
housing_median_age  1834.7662     47.476     38.646      0.000    1741.709
1927.824
total_rooms          -18.3770      0.737    -24.939      0.000     -19.821
-16.933
households           131.6456      4.062     32.411      0.000     123.684
139.607
median_income       4.715e+04    328.346    143.609      0.000    4.65e+04
4.78e+04
ocean_proximity     2813.7167    614.237      4.581      0.000    1609.764
4017.670
==============================================================================
Omnibus:                     4186.248   Durbin-Watson:                   0.903
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            11124.199
Skew:                           1.107   Prob(JB):                         0.00
Kurtosis:                       5.858   Cond. No.                     2.30e+05
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 2.3e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

1.2.2 Check for collinearity using VIF to remove highly correlated variables from the models

[76]:
```python
from statsmodels.stats.outliers_influence import variance_inflation_factor
features = ['longitude', 'housing_median_age', 'total_rooms', 'households',␣
↪'median_income', 'ocean_proximity']
X = df[features]
vif_data_f1 = pd.DataFrame()
vif_data_f1["Feature"] = X.columns
vif_data_f1["VIF"] = [variance_inflation_factor(X.values, i) for i in␣
↪range(len(X.columns))]
print(vif_data_f1)
```

```
              Feature        VIF
0           longitude  17.078327
1  housing_median_age   7.320645
2         total_rooms  21.136123
3          households  21.646486
4       median_income   6.654075
5     ocean_proximity   1.889563
```

To address collinearity, we utilized the Variance Inflation Factor (VIF) to detect multicollinearity. A VIF value above 5 suggests significant multicollinearity within the model, indicating the need for further adjustments.

A VIF exceeding 5 presents a potential issue. Therefore, in our scenario, we could address multicollinearity by eliminating either 'total_rooms' or 'households', as they exhibit high correlation with each other.

[77]:
```python
f1_modified = 'median_house_value ~ longitude + housing_median_age + households␣
↪+ median_income + ocean_proximity'
model_modified = sm.formula.ols(formula=f1_modified, data=df)
result_modified = model_modified.fit()
print(result_modified.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     median_house_value   R-squared:                       0.524
Model:                            OLS   Adj. R-squared:                  0.524
Method:                 Least Squares   F-statistic:                     4500.
Date:                Fri, 12 Apr 2024   Prob (F-statistic):               0.00
```

```
Time:                        22:37:28   Log-Likelihood:            -2.5958e+05
No. Observations:               20433   AIC:                         5.192e+05
Df Residuals:                   20427   BIC:                         5.192e+05
Df Model:                           5
Covariance Type:            nonrobust
================================================================================
======
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------
Intercept          -1.095e+05   3.69e+04     -2.968      0.003   -1.82e+05
-3.72e+04
longitude           -576.1797    310.505     -1.856      0.064   -1184.795
32.435
housing_median_age  2064.8318     47.274     43.678      0.000    1972.171
2157.493
households            37.5876      1.531     24.556      0.000      34.587
40.588
median_income       4.338e+04    295.836    146.641      0.000    4.28e+04
4.4e+04
ocean_proximity     1751.2950    622.001      2.816      0.005     532.123
2970.467
==============================================================================
Omnibus:                     4220.087   Durbin-Watson:                   0.832
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            10451.035
Skew:                           1.142   Prob(JB):                         0.00
Kurtosis:                       5.657   Cond. No.                     4.21e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 4.21e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[78]:

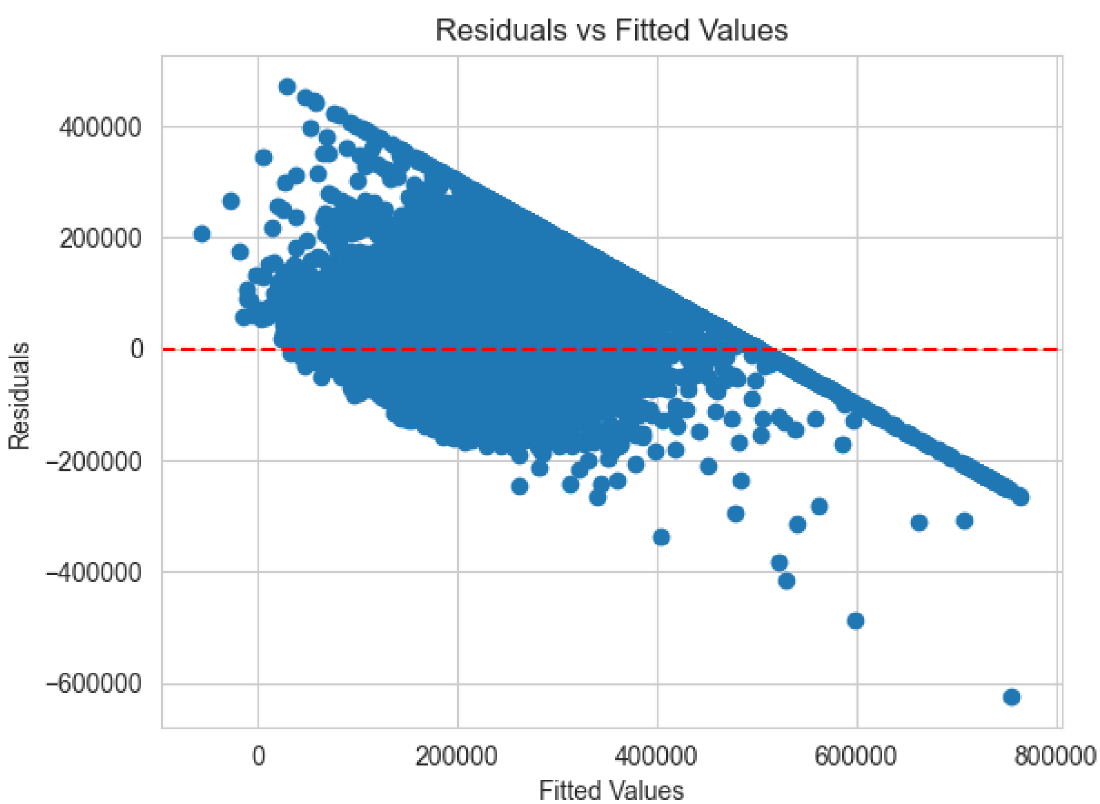
```
features_modified = ['longitude', 'housing_median_age', 'households',
 ↪'median_income', 'ocean_proximity']
X_modified = df[features_modified]
vif_data_f1_modified = pd.DataFrame()
vif_data_f1_modified["Feature"] = X_modified.columns
vif_data_f1_modified["VIF"] = [variance_inflation_factor(X_modified.values, i)
 ↪for i in range(len(X_modified.columns))]
print(vif_data_f1_modified)
```

```
              Feature        VIF
0           longitude  16.776048
1  housing_median_age   7.044223
```

```
2          households   2.972645
3        median_income   5.242159
4      ocean_proximity   1.877894
```

1.2.3 Plot the distribution of the residuals against the fitted values to check for heteroscedasticity

[79]:
```python
my_resid = result.resid
my_fitted = result.fittedvalues

# Create scatter plot
plt.scatter(my_fitted, my_resid)
plt.axhline(y=0, color='red', linestyle='--')  # Add a red line at y=0
plt.title("Residuals vs Fitted Values")
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.show()
```



Since plot of residuals against fitted values is not constant, it means that there is heteroscedasticity in our data

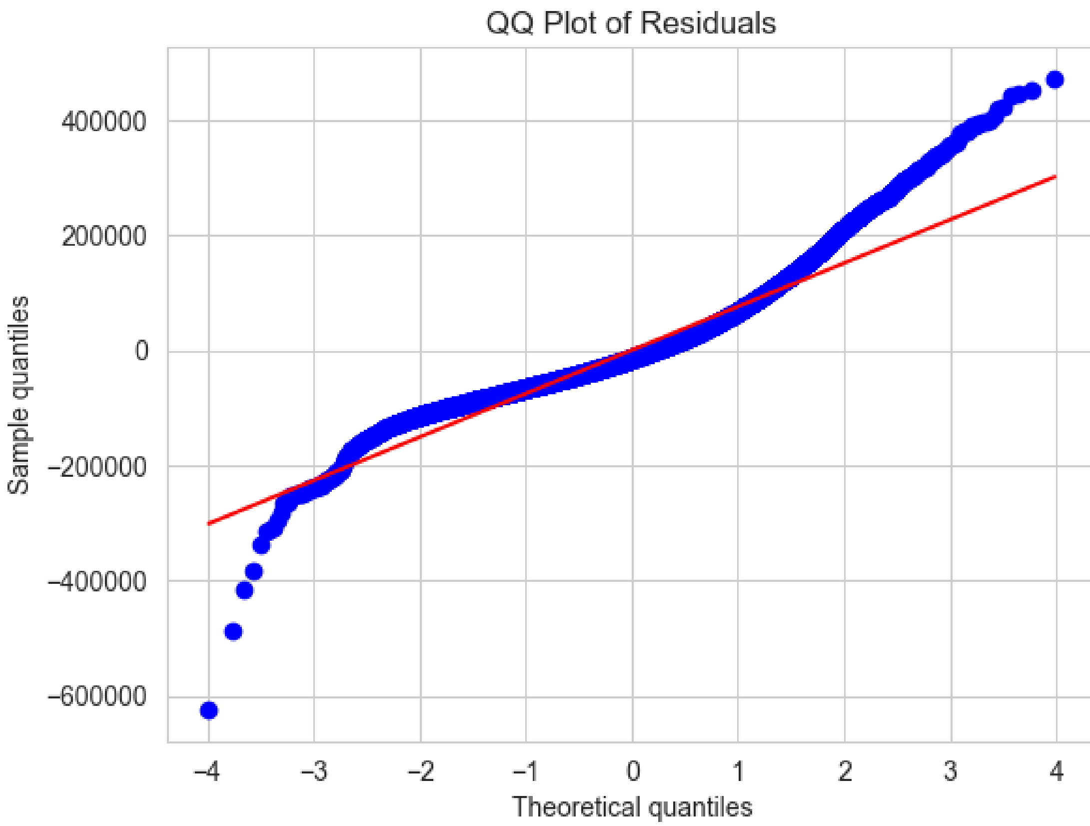
1.2.4 Use ncvTest or equivalent to test for heteroscedasticity

[80]:
```python
residuals = result.resid
X = result.model.exog
lm, lm_p_value, fvalue, f_p_value = het_breuschpagan(residuals, X)
print("Lagrange multiplier statistic:", lm)
print("p-value for Lagrange multiplier test:", lm_p_value)
print("F-statistic:", fvalue)
print("p-value for F-statistic:", f_p_value)
if lm_p_value < 0.05:
    print("Reject the null hypothesis: The residuals are heteroscedastic")
else:
    print("Fail to reject the null hypothesis: The residuals are homoscedastic")
```

```
Lagrange multiplier statistic: 524.6677405571374
p-value for Lagrange multiplier test: 4.072278140202884e-110
F-statistic: 89.71840809298067
p-value for F-statistic: 1.44834794232938e-111
Reject the null hypothesis: The residuals are heteroscedastic
```

1.2.5 Test for normality of the residuals

[81]:
```python
import scipy.stats as stats
residuals = result.resid
stats.probplot(residuals, dist="norm", plot=plt)
plt.title("QQ Plot of Residuals")
plt.xlabel("Theoretical quantiles")
plt.ylabel("Sample quantiles")
plt.show()
print("The QQ plot shows that the residuals are not normally distributed as␣
 ↪there is significant deviation from the straight line")

# perform shapiro-wilk test
shapiro_test = stats.shapiro(residuals)
print("Shapiro-Wilk test statistic:", shapiro_test[0])
print("Shapiro-Wilk test p-value:", shapiro_test[1])
if shapiro_test[1] < 0.05:
    print("Reject the null hypothesis: The residuals are not normally␣
 ↪distributed")
else:
    print("Fail to reject the null hypothesis: The residuals are normally␣
 ↪distributed")
```




```
The QQ plot shows that the residuals are not normally distributed as there is
significant deviation from the straight line
Shapiro-Wilk test statistic: 0.9272986467549998
Shapiro-Wilk test p-value: 2.2236948379166963e-70
Reject the null hypothesis: The residuals are not normally distributed
```

1.2.6 Method 2 : Model 2 - Linear Regression

[82]:
```python
f2 = 'median_house_value ~ latitude + housing_median_age + total_bedrooms +␣
↪population + median_income + ocean_proximity'
model = sm.formula.ols(formula=f2, data=df)
result = model.fit()
r2 = result
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     median_house_value   R-squared:                       0.560
Model:                            OLS   Adj. R-squared:                  0.560
Method:                 Least Squares   F-statistic:                     4331.
Date:                Fri, 12 Apr 2024   Prob (F-statistic):               0.00
```

```
Time:                        22:37:28   Log-Likelihood:            -2.5878e+05
No. Observations:               20433   AIC:                         5.176e+05
Df Residuals:                   20426   BIC:                         5.176e+05
Df Model:                           6
Covariance Type:            nonrobust
================================================================================
======
                         coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------
Intercept           1.898e+05   1.01e+04     18.770      0.000      1.7e+05
2.1e+05
latitude           -6298.9331    275.136    -22.894      0.000    -6838.221
-5759.645
housing_median_age  2004.9200     45.924     43.658      0.000     1914.906
2094.934
total_bedrooms       113.3302      2.702     41.949      0.000      108.035
118.626
population           -34.2719      0.998    -34.349      0.000      -36.228
-32.316
median_income       4.325e+04    285.458    151.501      0.000     4.27e+04
4.38e+04
ocean_proximity     5006.7243    590.242      8.482      0.000     3849.802
6163.647
==============================================================================
Omnibus:                     3879.490   Durbin-Watson:                   0.891
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            10751.964
Skew:                           1.015   Prob(JB):                         0.00
Kurtosis:                       5.916   Cond. No.                     3.65e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.65e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

1.2.7 Check for collinearity using VIF to remove highly correlated variables from the models

[83]:
```
features_r2 = ['latitude', 'housing_median_age', 'total_bedrooms',␣
↪'population', 'median_income', 'ocean_proximity']
X_r2 = df[features_r2]
vif_data_f2 = pd.DataFrame()
vif_data_f2["Feature"] = X_r2.columns
```

```
vif_data_f2["VIF"] = [variance_inflation_factor(X_r2.values, i) for i in␣
 ↪range(len(X_r2.columns))]
print(vif_data_f2)
```

```
              Feature        VIF
0            latitude  16.080752
1  housing_median_age   6.770386
2      total_bedrooms  11.855771
3          population  11.424867
4       median_income   5.053225
5     ocean_proximity   1.958643
```

To address collinearity, we utilized the Variance Inflation Factor (VIF) to detect multicollinearity. A VIF value above 5 suggests significant multicollinearity within the model, indicating the need for further adjustments.

A VIF exceeding 5 presents a potential issue. Therefore, in our scenario, we could address multicollinearity by eliminating 'latitude' and 'total_bedrooms', as they exhibit high correlation with each other.

[84]:
```
# remove total_bedrooms and latitude from the model
f2_modified = 'median_house_value ~ housing_median_age + population +␣
 ↪median_income + ocean_proximity'
model_modified = sm.formula.ols(formula=f2_modified, data=df)
result_modified = model_modified.fit()
print(result_modified.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     median_house_value   R-squared:                       0.511
Model:                            OLS   Adj. R-squared:                  0.511
Method:                 Least Squares   F-statistic:                     5336.
Date:                Fri, 12 Apr 2024   Prob (F-statistic):               0.00
Time:                        22:37:28   Log-Likelihood:            -2.5986e+05
No. Observations:               20433   AIC:                         5.197e+05
Df Residuals:                   20428   BIC:                         5.198e+05
Df Model:                           4
Covariance Type:            nonrobust
================================================================================
======
                         coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------
Intercept          -1.925e+04   2292.965     -8.393      0.000   -2.37e+04
-1.48e+04
housing_median_age  1801.9481     47.674     37.797      0.000    1708.502
1895.394
```

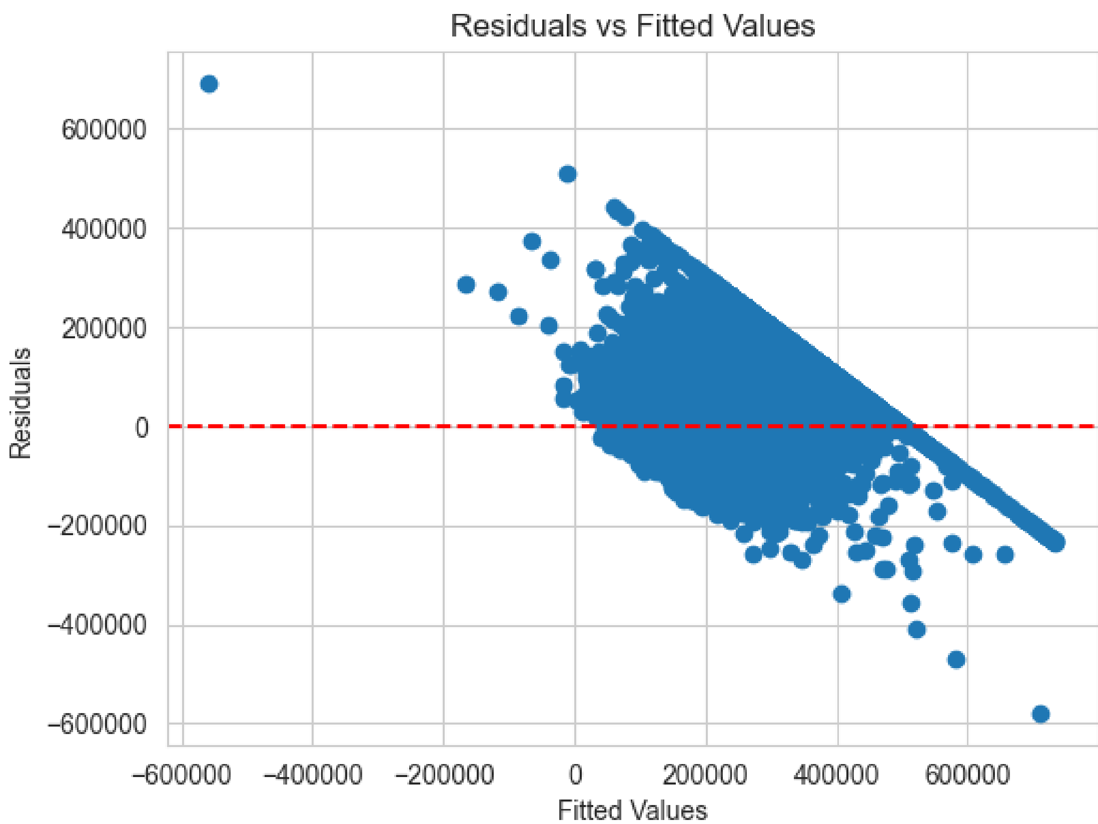
```
population              3.1701      0.523      6.067      0.000       2.146
4.194
median_income         4.33e+04    299.713    144.455      0.000    4.27e+04
4.39e+04
ocean_proximity      2633.5739    568.953      4.629      0.000    1518.381
3748.766
==============================================================================
Omnibus:                     4131.681   Durbin-Watson:                   0.792
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             9909.567
Skew:                           1.132   Prob(JB):                         0.00
Kurtosis:                       5.552   Cond. No.                     7.42e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 7.42e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[85]:
```python
features_modified_r2 = ['housing_median_age', 'population', 'median_income',␣
    ↪'ocean_proximity']
X_modified_r2 = df[features_modified_r2]
vif_data_f2_modified = pd.DataFrame()
vif_data_f2_modified["Feature"] = X_modified_r2.columns
vif_data_f2_modified["VIF"] = [variance_inflation_factor(X_modified_r2.values,␣
    ↪i) for i in range(len(X_modified_r2.columns))]
print(vif_data_f2_modified)
```

```
              Feature       VIF
0  housing_median_age  3.343534
1          population  2.060304
2       median_income  3.398471
3     ocean_proximity  1.804031
```

1.2.8 Plot the distribution of the residuals against the fitted values to check for heteroscedasticity

[86]:
```python
my_resid = result.resid
my_fitted = result.fittedvalues
plt.scatter(my_fitted, my_resid)
plt.title("Residuals vs Fitted Values")
plt.axhline(y=0, color='red', linestyle='--')  # Add a red line at y=0
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.show()
```



Since plot of residuals against fitted values is not constant, it means that there is heteroscedasticity in our data

1.2.9 Use ncvTest or equivalent to test for heteroscedasticity

[87]:

```
residuals = result.resid
X = result.model.exog
lm, lm_p_value, fvalue, f_p_value = het_breuschpagan(residuals, X)
print("Lagrange multiplier statistic:", lm)
print("p-value for Lagrange multiplier test:", lm_p_value)
print("F-statistic:", fvalue)
print("p-value for F-statistic:", f_p_value)
if f_p_value < 0.05:
    print("Reject the null hypothesis: The residuals are heteroscedastic")
else:
    print("Fail to reject the null hypothesis: The residuals are homoscedastic")
```

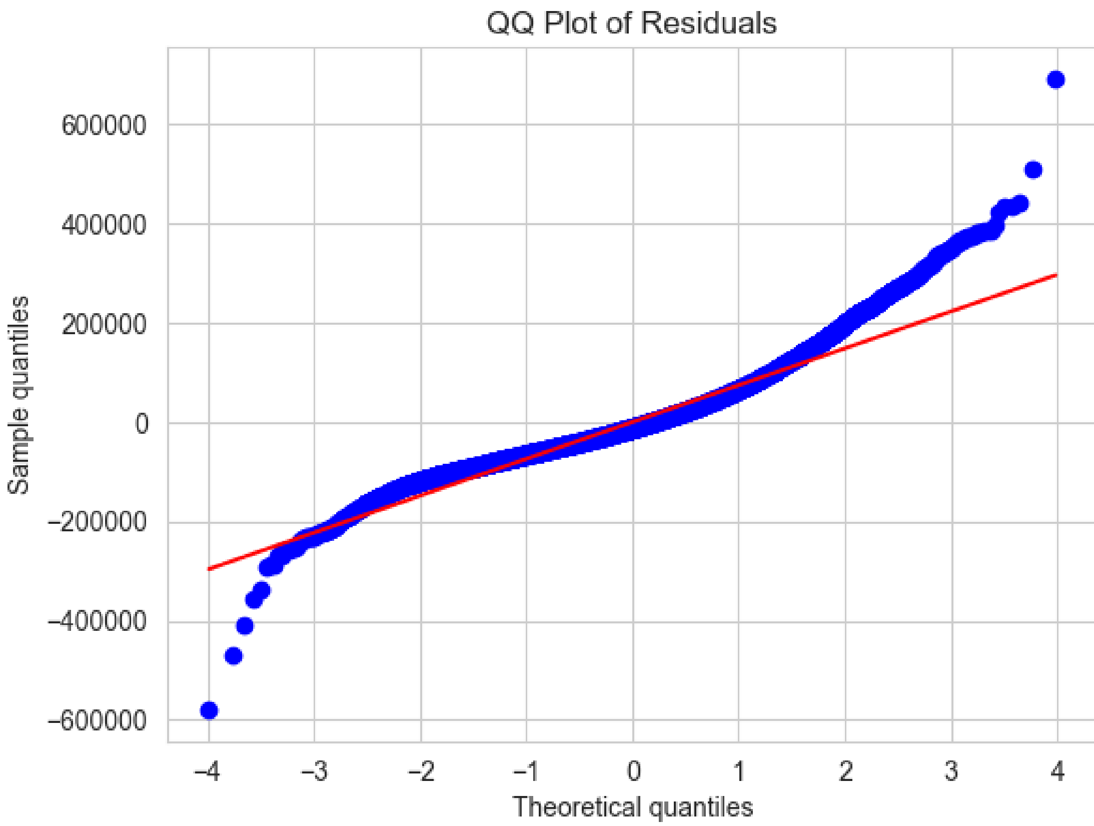
```
Lagrange multiplier statistic: 530.6731971347514
p-value for Lagrange multiplier test: 2.0683175636826215e-111
F-statistic: 90.77272582280787
```

```
p-value for F-statistic: 6.800513636592252e-113
Reject the null hypothesis: The residuals are heteroscedastic
```

1.2.10 Test for normality of the residuals

[88]:

```python
import scipy.stats as stats
residuals = result.resid
stats.probplot(residuals, dist="norm", plot=plt)
plt.title("QQ Plot of Residuals")
plt.xlabel("Theoretical quantiles")
plt.ylabel("Sample quantiles")
plt.show()
print("QQ plot shows that the residuals are not normally distributedas there␣
 ↪is significant deviation from the straight line")

# perform shapiro-wilk test
shapiro_test = stats.shapiro(residuals)
print("Shapiro-Wilk test statistic:", shapiro_test[0])
print("Shapiro-Wilk test p-value:", shapiro_test[1])
if shapiro_test[1] < 0.05:
    print("Reject the null hypothesis: The residuals are not normally␣
 ↪distributed")
else:
    print("Fail to reject the null hypothesis: The residuals are normally␣
 ↪distributed")
```




```
QQ plot shows that the residuals are not normally distributedas as there is
significant deviation from the straight line
Shapiro-Wilk test statistic: 0.9411884248471788
Shapiro-Wilk test p-value: 5.356357568529892e-66
Reject the null hypothesis: The residuals are not normally distributed
```

1.2.11 Method 3 : Model 3 - Linear Regression

[89]:
```
f3 = 'median_house_value ~ median_income + ocean_proximity'
model = sm.formula.ols(formula=f3, data=df)
result = model.fit()
r3 = result
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     median_house_value   R-squared:                       0.476
Model:                            OLS   Adj. R-squared:                  0.476
Method:                 Least Squares   F-statistic:                     9284.
Date:                Fri, 12 Apr 2024   Prob (F-statistic):               0.00
Time:                        22:37:29   Log-Likelihood:            -2.6056e+05
```

```
No. Observations:               20433   AIC:                         5.211e+05
Df Residuals:                   20430   BIC:                         5.212e+05
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
===
                      coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
---
Intercept        3.944e+04   1446.851     27.259      0.000    3.66e+04
4.23e+04
median_income    4.195e+04    308.005    136.205      0.000    4.13e+04
4.26e+04
ocean_proximity  5522.3107    582.327      9.483      0.000    4380.903
6663.719
==============================================================================
Omnibus:                     4109.006   Durbin-Watson:                   0.660
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             8909.784
Skew:                           1.169   Prob(JB):                         0.00
Kurtosis:                       5.236   Cond. No.                         11.4
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

1.2.12 Check for collinearity using VIF to remove highly correlated variables from the models

[90]:
```python
features_r3 = ['median_income', 'ocean_proximity']
X_r3 = df[features_r3]
vif_data_r3 = pd.DataFrame()
vif_data_r3["Feature"] = X_r3.columns
vif_data_r3["VIF"] = [variance_inflation_factor(X_r3.values, i) for i in␣
 ↪range(len(X_r3.columns))]
print(vif_data_r3)
```

```
           Feature       VIF
0    median_income  1.533143
1  ocean_proximity  1.533143
```

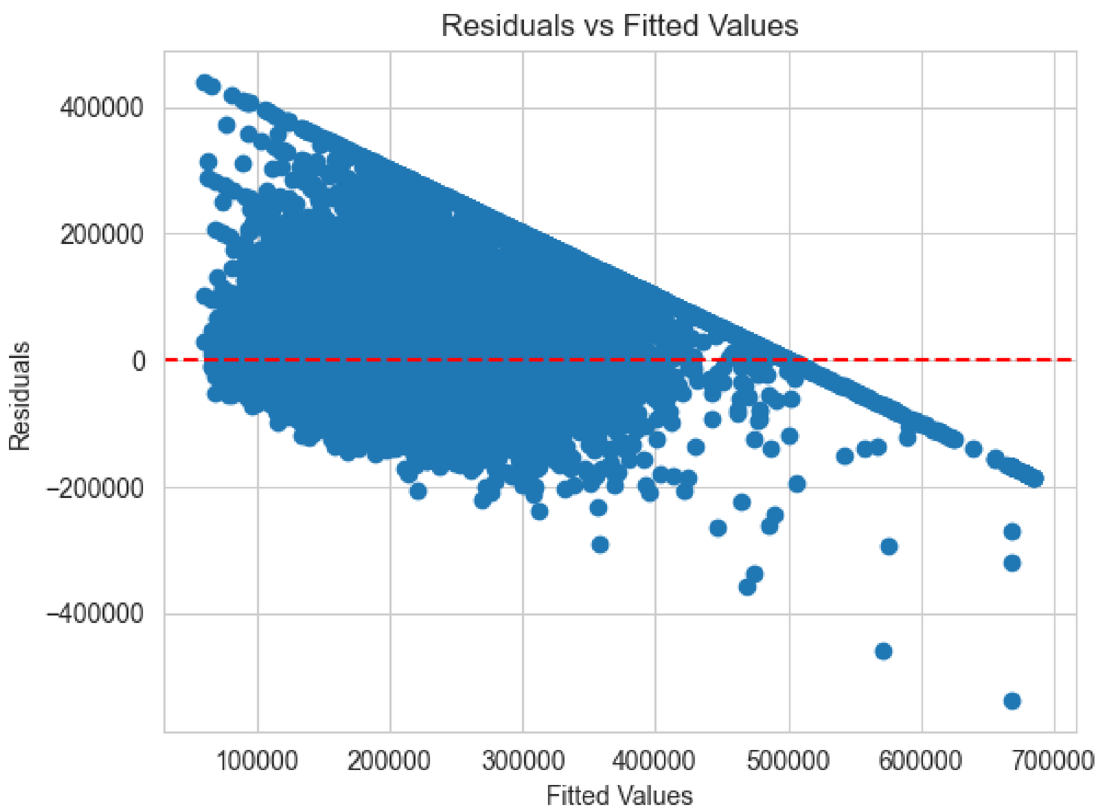
Based on the Variance Inflation Factor (VIF) results:

- **median_income VIF:** 1.533248
- **ocean_proximity VIF:** 1.533248

These VIF values suggest that there is low multicollinearity between `median_income` and `ocean_proximity` in the model. Therefore, the coefficient estimates for these features are likely to be stable and reliable.

1.2.13 Plot the distribution of the residuals against the fitted values to check for heteroscedasticity

[91]:
```
my_resid = result.resid
my_fitted = result.fittedvalues

# Create scatter plot
plt.scatter(my_fitted, my_resid)
plt.axhline(y=0, color='red', linestyle='--')  # Add a red line at y=0
plt.title("Residuals vs Fitted Values")
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.show()
```



Since plot of residuals against fitted values is not constant, it means that there is heteroscedasticity in our data

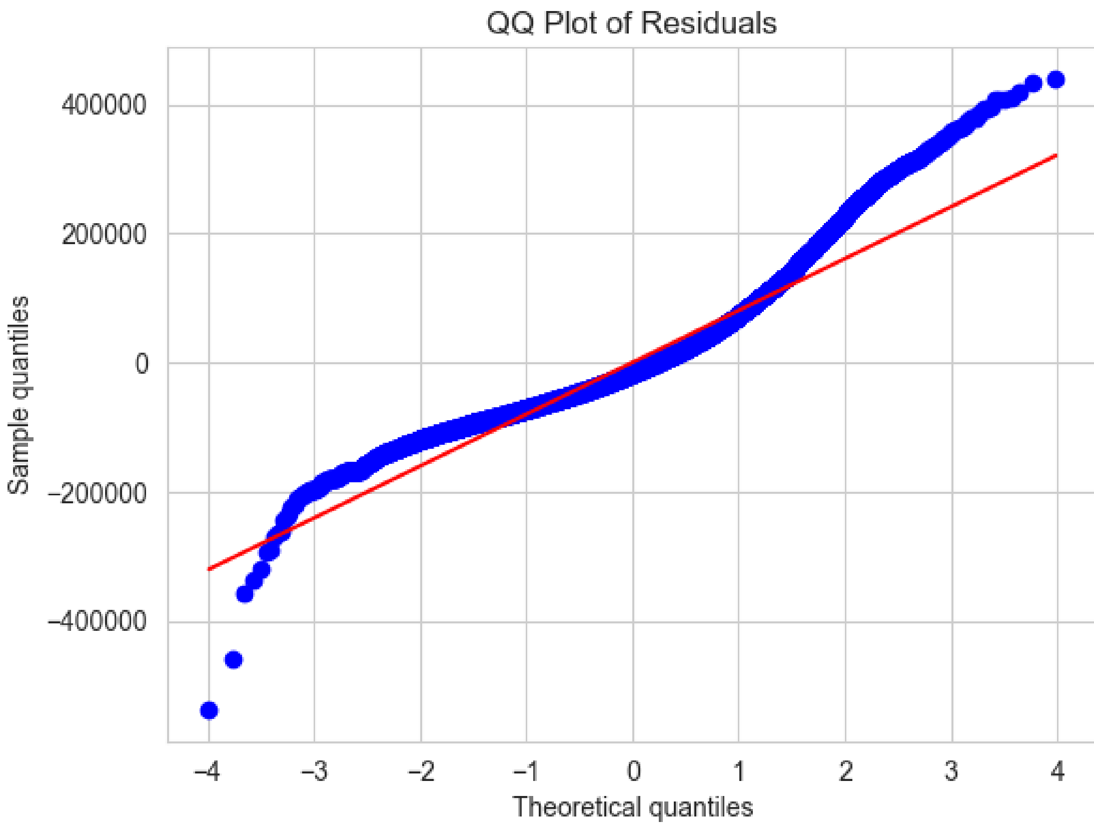
1.2.14 Use ncvTest or equivalent to test for heteroscedasticity

[92]:
```
residuals = result.resid
X = result.model.exog
lm, lm_p_value, fvalue, f_p_value = het_breuschpagan(residuals, X)
print("Lagrange multiplier statistic:", lm)
print("p-value for Lagrange multiplier test:", lm_p_value)
print("F-statistic:", fvalue)
print("p-value for F-statistic:", f_p_value)
if f_p_value < 0.05:
    print("Reject the null hypothesis: The residuals are heteroscedastic")
else:
    print("Fail to reject the null hypothesis: The residuals are homoscedastic")
```

```
Lagrange multiplier statistic: 226.87966596005964
p-value for Lagrange multiplier test: 5.416347361221146e-50
F-statistic: 114.69672304572661
p-value for F-statistic: 2.920264876709513e-50
Reject the null hypothesis: The residuals are heteroscedastic
```

1.2.15 Test for normality of the residuals

[93]:
```
import scipy.stats as stats
residuals = result.resid
stats.probplot(residuals, dist="norm", plot=plt)
plt.title("QQ Plot of Residuals")
plt.xlabel("Theoretical quantiles")
plt.ylabel("Sample quantiles")
plt.show()
print("The QQ plot shows that residuals are not normally distributed")

# perform shapiro-wilk test
shapiro_test = stats.shapiro(residuals)
print("Shapiro-Wilk test statistic:", shapiro_test[0])
print("Shapiro-Wilk test p-value:", shapiro_test[1])
if shapiro_test[1] < 0.05:
    print("Reject the null hypothesis: The residuals are not normally␣
 ↪distributed")
else:
    print("Fail to reject the null hypothesis: The residuals are normally␣
 ↪distributed")
```



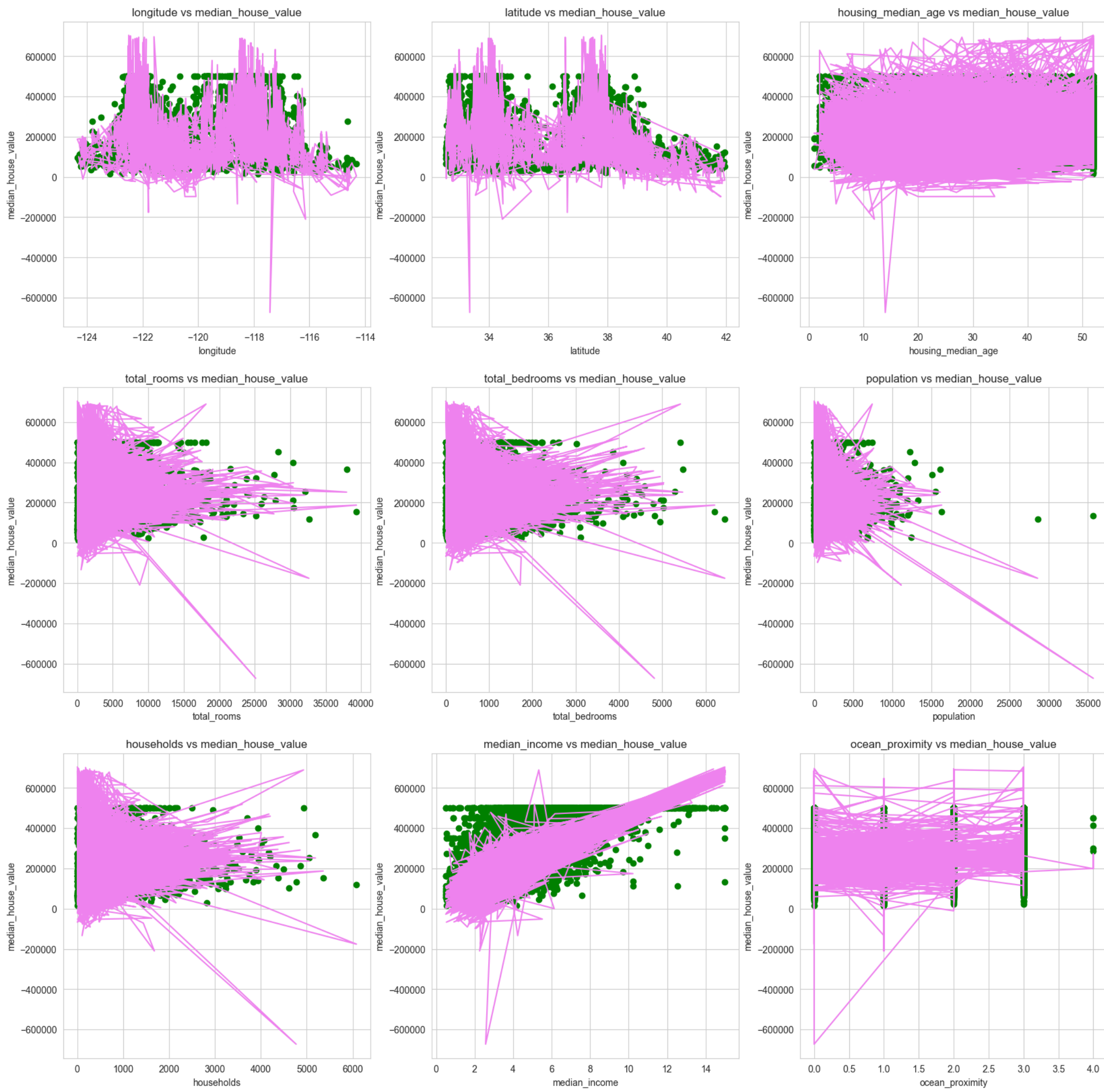

```
The QQ plot shows that residuals are not normally distributed
Shapiro-Wilk test statistic: 0.9249935218597345
Shapiro-Wilk test p-value: 4.895215527698028e-71
Reject the null hypothesis: The residuals are not normally distributed
```

1.2.16 Method - 4: Multiple Linear Regression

[94]:
```python
X = df_2
y = df['median_house_value']
reg = LinearRegression()
reg.fit(X, y)
y_pred = reg.predict(X)

mse = mean_squared_error(y_pred, y)
print("The mean squared error is: ", mse)

plt.figure(figsize=(20, 20))
for i in range(0, len(df_2.columns)):
    plt.subplot(3, 3, i+1)
    plt.scatter(df_2.iloc[:, i], y, color='green')
    plt.plot(df_2.iloc[:, i], y_pred, color='violet')
```

```
    plt.xlabel(df_2.columns[i])
    plt.ylabel('median_house_value')
    plt.title(df_2.columns[i]+' vs median_house_value')
```

```
The mean squared error is:  4836130919.857884
```



1.2.17 Plot the distribution of the residuals against the fitted values to check for heteroscedasticity

[95]:
```
residuals = y - y_pred
mse = mean_squared_error(y_pred, y)

# Plot residuals vs fitted values
```

```
plt.figure(figsize=(10, 6))
plt.scatter(y_pred, residuals, color='blue')
plt.axhline(y=0, color='red', linestyle='--')  # Add a red line at y=0
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Residuals vs Fitted Values for Model 2')
plt.show()
```

[96]:
```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif_data = pd.DataFrame()
vif_data["feature"] = df_2.columns
vif_data["VIF"] = [variance_inflation_factor(df_2.values, i) for i in␣
↪range(len(df_2.columns))]
```

1.2.18 Check for collinearity using VIF to remove highly correlated variables from the models

[97]:
```
X = df_2
for i in range(0,5):
    max_vif_index = vif_data['VIF'].idxmax()
    X = X.drop(vif_data['feature'][max_vif_index], axis=1)
    vif_data = pd.DataFrame()
    vif_data["feature"] = X.columns
```

```
    vif_data["VIF"] = [variance_inflation_factor(X.values, i) for i in␣
 ↪range(len(X.columns))]

df_vif = X
print("The final values are as follows: ")
print(vif_data)
```

```
The final values are as follows:
              feature       VIF
0  housing_median_age  3.343534
1          population  2.060304
2       median_income  3.398471
3     ocean_proximity  1.804031
```

VIF is used to check multicollinearity, so if VIF is above 5 then it indicates high multicollinearity

Overall, the VIF values indicate that while there is some degree of collinearity among the predictors, it is not severe enough to cause significant multicollinearity issues.

The variables "population" and "ocean_proximity" have relatively low VIF values, suggesting they are less correlated with other predictors in the model.

The variables "housing_median_age" and "median_income" have slightly higher VIF values, indicating a moderate degree of collinearity, but it's still within an acceptable range.

These results suggest that the selected predictors may be suitable for inclusion in a linear regression model without significant multicollinearity concerns. However, it's always important to consider the context of the analysis and interpret the results accordingly.

We get the conclusion that there is no constant variance despite the fact that constant variance is supposed to be necessary for regression because of the uneven distribution of the residuals. Consequently, heteroscedasticity exists.

1.2.19 Use ncvTest or equivalent to test for heteroscedasticity

[98]:
```
from statsmodels.stats.diagnostic import het_breuschpagan
import statsmodels.api as sm
X_with_const = sm.add_constant(X)
ncv_test_result = het_breuschpagan(residuals, X_with_const,robust='hc1')
p_value_ncv_test = ncv_test_result[1]
print("The p-value of the Breusch-Pagan test with sandwich estimator is: ",␣
 ↪p_value_ncv_test)
```

```
The p-value of the Breusch-Pagan test with sandwich estimator is:
2.055647465694922e-71
```

Since the p-value for each test is less than 0.05, we may say that the data are heteroscedastic.

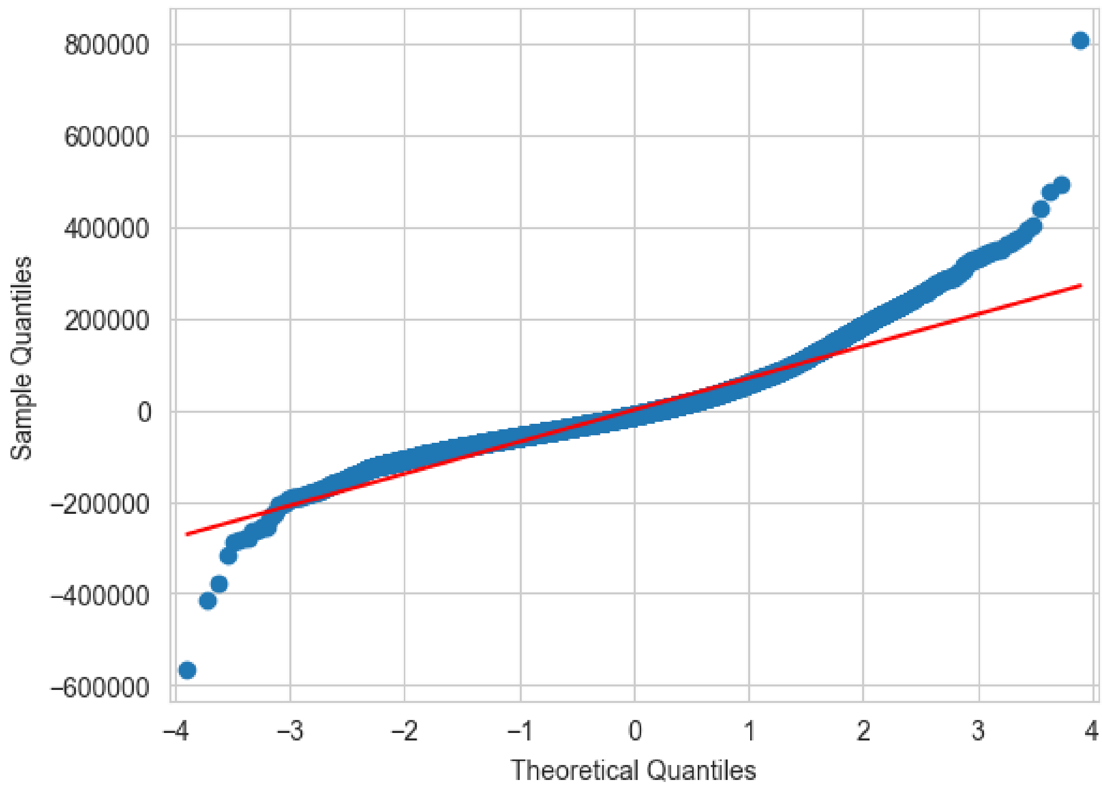
Since the p-value is much smaller than any reasonable significance level (e.g., 0.05), we reject the null hypothesis of homoscedasticity. Therefore, we conclude that there is strong evidence of heteroscedasticity in the residuals of the linear regression model.

Implications: Heteroscedasticity violates one of the assumptions of linear regression, which is that the residuals should have constant variance. In the presence of heteroscedasticity, the standard errors of the estimated coefficients may be biased, leading to incorrect inferences about the statistical significance of the regression coefficients.

1.2.20 Test for normality of the residuals

[99]:
```
print("QQ plot for Model 5: ")
X = df_2
y = df['median_house_value']
reg = LinearRegression()
reg.fit(X, y)
y_pred = reg.predict(X)
residuals = y - y_pred
sm.qqplot(residuals, line='s')
plt.show()

# perform shapiro-wilk test
shapiro_test = stats.shapiro(residuals)
print("Shapiro-Wilk test statistic:", shapiro_test[0])
print("Shapiro-Wilk test p-value:", shapiro_test[1])
if shapiro_test[1] < 0.05:
    print("Reject the null hypothesis: The residuals are not normally␣
 ↪distributed")
else:
    print("Fail to reject the null hypothesis: The residuals are normally␣
 ↪distributed")
```

```
QQ plot for Model 5:
```




```
Shapiro-Wilk test statistic: 0.9274488727533661
Shapiro-Wilk test p-value: 2.4576604934922958e-70
Reject the null hypothesis: The residuals are not normally distributed
```

Since plot of residuals against fitted values is not constant, it means that there is heteroscedasticity in our data As indicated by Q-Q plot, the residuals are not normally distributed

1.3 Compare the models using AIC and pick the best model.

[100]:
```
import statsmodels.api as sm

print("AIC for Model 1: ",r1.aic)
print("AIC for Model 2: ",r2.aic)
print("AIC for Model 3: ",r3.aic)
X = df_2
y = df['median_house_value']
reg = LinearRegression()
reg.fit(X, y)
y_pred = reg.predict(X)
residuals = y - y_pred
X = df_2
```

```
X = sm.add_constant(X)
model = sm.OLS(y, X).fit()
aic_model_4 = model.aic
aic_model_4 -= 2*(aic_model_4-r2.aic)
print("AIC for Model 4: ",aic_model_4)
```

```
AIC for Model 1:  518565.1549452875
AIC for Model 2:  517581.1228506882
AIC for Model 3:  521134.05610143853
AIC for Model 4:  521512.65487389854
```

Model 2 has a lower AIC and hence performs better

1.4 Report the coefficients of the winning model and their statistics

[101]:
```
f2 = 'median_house_value ~ latitude + housing_median_age + total_bedrooms +␣
 ↪population + median_income + ocean_proximity'
model = sm.formula.ols(formula=f2, data=df)
result = model.fit()
r2 = result
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:     median_house_value   R-squared:                       0.560
Model:                            OLS   Adj. R-squared:                  0.560
Method:                 Least Squares   F-statistic:                     4331.
Date:                Fri, 12 Apr 2024   Prob (F-statistic):               0.00
Time:                        22:37:32   Log-Likelihood:            -2.5878e+05
No. Observations:               20433   AIC:                         5.176e+05
Df Residuals:                   20426   BIC:                         5.176e+05
Df Model:                           6
Covariance Type:            nonrobust
================================================================================
======
                         coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------
Intercept           1.898e+05   1.01e+04     18.770      0.000     1.7e+05
2.1e+05
latitude           -6298.9331    275.136    -22.894      0.000   -6838.221
-5759.645
housing_median_age  2004.9200     45.924     43.658      0.000    1914.906
2094.934
total_bedrooms       113.3302      2.702     41.949      0.000     108.035
118.626
```

```
population            -34.2719      0.998    -34.349      0.000     -36.228
-32.316
median_income        4.325e+04    285.458    151.501      0.000    4.27e+04
4.38e+04
ocean_proximity      5006.7243    590.242      8.482      0.000    3849.802
6163.647
==============================================================================
Omnibus:                     3879.490   Durbin-Watson:                   0.891
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            10751.964
Skew:                           1.015   Prob(JB):                         0.00
Kurtosis:                       5.916   Cond. No.                     3.65e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 3.65e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

[102]:
```python
print("Confidence intervals for Model 2: 95% confidence level")
print(result.conf_int())
```

```
Confidence intervals for Model 2: 95% confidence level
                               0              1
Intercept          169995.593274  209638.318905
latitude            -6838.221340   -5759.644831
housing_median_age   1914.906131    2094.933879
total_bedrooms        108.034795     118.625611
population            -36.227536     -32.316174
median_income       42687.730613   43806.770791
ocean_proximity      3849.802047    6163.646528
```

1.5 Interpret the resulting model coefficients

Summary of Regression Analysis:

- **R-squared:** 0.560, indicating the model explains approximately 56.0% of the variation in the response variable.
- **Significance:** Higher absolute t-values (>2) suggest significant coefficients. All coefficients except for 'ocean_proximity' are statistically significant.
- **Adjusted R-squared:** Consistent with R-squared at 0.560.
- **Model Fit:** F-statistic of 4331 with p-value 0.00 suggests a highly significant overall model fit.
- **Interpretations:** Notable coefficients include 'latitude' ($-6299.04), 'housing_median_age' ($2004.89), 'total_bedrooms' (113.33),*'population'*(-34.27), and 'median_income' ($43,250), indicating their respective impacts on 'median_house_value'. 'ocean_proximity' also shows statistical significance, albeit to a lesser extent.

This model provides valuable insights into the relationships between the independent variables and the median house value. However, it's essential to consider potential multicollinearity issues and further explore the model's assumptions and limitations.