Multivariable_Regression_and_Valuation_Model

September 25, 2023

1 Predict House Prices

1.0.1 Introduction

Welcome to Boston Massachusetts in the 1970s!in this project we woll analyse data for a real estate development company. the company wants to value any residential project before they start. we will building a model that can provide a price estimate based on a home's characteristics like:

- The number of rooms
- The distance to employment centres
- How rich or poor the area is
- How many students there are per teacher in local schools etc

We will:

1. Analyse and explore the Boston house price data
2. Split our data for training and testing
3. Run a Multivariable Regression
4. Evaluate how our model's coefficients and residuals
5. Use data transformation to improve our model performance
6. Use our model to estimate a property price

1.0.2 Import Statements

[46]:
```
import pandas as pd
import numpy as np

import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

1.0.3 Notebook Presentation

[47]:
```
pd.options.display.float_format = '{:,.2f}'.format
```

2 Load the Data

The first column in the .csv file just has the row numbers, so it will be used as the index.

[48]:
```python
data = pd.read_csv('boston.csv', index_col=0)
```

2.0.1 Understand the Boston House Price Dataset

Characteristics:

```
:Number of Instances: 506

:Number of Attributes: 13 numeric/categorical predictive. The Median Value (attribute 14) is t

:Attribute Information (in order):
    1. CRIM     per capita crime rate by town
    2. ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
    3. INDUS    proportion of non-retail business acres per town
    4. CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
    5. NOX      nitric oxides concentration (parts per 10 million)
    6. RM       average number of rooms per dwelling
    7. AGE      proportion of owner-occupied units built prior to 1940
    8. DIS      weighted distances to five Boston employment centres
    9. RAD      index of accessibility to radial highways
    10. TAX      full-value property-tax rate per $10,000
    11. PTRATIO  pupil-teacher ratio by town
    12. B        1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
    13. LSTAT    % lower status of the population
    14. PRICE     Median value of owner-occupied homes in $1000's
```

3 Preliminary Data Exploration

- What is the shape of `data`?
- How many rows and columns does it have?
- What are the column names?
- Are there any NaN values or duplicates?

[49]:
```python
data.shape # 506 data points
```

[49]:
```
(506, 14)
```

[50]:
```python
data.columns # column names
```

[50]:
```
Index(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD', 'TAX',
       'PTRATIO', 'B', 'LSTAT', 'PRICE'],
      dtype='object')
```

[51]:
```
data.head()
```

[51]:
```
   CRIM    ZN  INDUS  CHAS  NOX   RM    AGE  DIS  RAD    TAX  PTRATIO      B  \
0  0.01 18.00   2.31  0.00 0.54 6.58 65.20 4.09 1.00 296.00    15.30 396.90
1  0.03  0.00   7.07  0.00 0.47 6.42 78.90 4.97 2.00 242.00    17.80 396.90
2  0.03  0.00   7.07  0.00 0.47 7.18 61.10 4.97 2.00 242.00    17.80 392.83
3  0.03  0.00   2.18  0.00 0.46 7.00 45.80 6.06 3.00 222.00    18.70 394.63
4  0.07  0.00   2.18  0.00 0.46 7.15 54.20 6.06 3.00 222.00    18.70 396.90

   LSTAT  PRICE
0   4.98  24.00
1   9.14  21.60
2   4.03  34.70
3   2.94  33.40
4   5.33  36.20
```

[52]:
```
data.tail()
```

[52]:
```
     CRIM   ZN  INDUS  CHAS  NOX   RM    AGE  DIS  RAD    TAX  PTRATIO      B  \
501  0.06 0.00  11.93  0.00 0.57 6.59 69.10 2.48 1.00 273.00    21.00 391.99
502  0.05 0.00  11.93  0.00 0.57 6.12 76.70 2.29 1.00 273.00    21.00 396.90
503  0.06 0.00  11.93  0.00 0.57 6.98 91.00 2.17 1.00 273.00    21.00 396.90
504  0.11 0.00  11.93  0.00 0.57 6.79 89.30 2.39 1.00 273.00    21.00 393.45
505  0.05 0.00  11.93  0.00 0.57 6.03 80.80 2.50 1.00 273.00    21.00 396.90

     LSTAT  PRICE
501   9.67  22.40
502   9.08  20.60
503   5.64  23.90
504   6.48  22.00
505   7.88  11.90
```

[53]:
```
data.count() # number of rows
```

[53]:
```
CRIM       506
ZN         506
INDUS      506
CHAS       506
NOX        506
RM         506
AGE        506
DIS        506
RAD        506
TAX        506
PTRATIO    506
B          506
LSTAT      506
```

```
PRICE      506
dtype: int64
```

3.1 Data Cleaning - Check for Missing Values and Duplicates

[54]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   CRIM     506 non-null    float64
 1   ZN       506 non-null    float64
 2   INDUS    506 non-null    float64
 3   CHAS     506 non-null    float64
 4   NOX      506 non-null    float64
 5   RM       506 non-null    float64
 6   AGE      506 non-null    float64
 7   DIS      506 non-null    float64
 8   RAD      506 non-null    float64
 9   TAX      506 non-null    float64
 10  PTRATIO  506 non-null    float64
 11  B        506 non-null    float64
 12  LSTAT    506 non-null    float64
 13  PRICE    506 non-null    float64
dtypes: float64(14)
memory usage: 59.3 KB
```

[55]:
```
print(f'Any NaN values? {data.isna().values.any()}')
```

```
Any NaN values? False
```

[56]:
```
print(f'Any duplicates? {data.duplicated().values.any()}')
```

```
Any duplicates? False
```

There are no null (i.e., NaN) values. Fantastic!

3.2 Descriptive Statistics

- How many students are there per teacher on average?
- What is the average price of a home in the dataset?
- What is the `CHAS` feature?
- What are the minimum and the maximum value of the `CHAS` and why?
- What is the maximum and the minimum number of rooms per dwelling in the dataset?

[57]:
```
data.describe()
```

[57]:
```
         CRIM     ZN  INDUS   CHAS    NOX     RM    AGE    DIS    RAD    TAX  \
count  506.00 506.00 506.00 506.00 506.00 506.00 506.00 506.00 506.00 506.00
mean     3.61  11.36  11.14   0.07   0.55   6.28  68.57   3.80   9.55 408.24
std      8.60  23.32   6.86   0.25   0.12   0.70  28.15   2.11   8.71 168.54
min      0.01   0.00   0.46   0.00   0.39   3.56   2.90   1.13   1.00 187.00
25%      0.08   0.00   5.19   0.00   0.45   5.89  45.02   2.10   4.00 279.00
50%      0.26   0.00   9.69   0.00   0.54   6.21  77.50   3.21   5.00 330.00
75%      3.68  12.50  18.10   0.00   0.62   6.62  94.07   5.19  24.00 666.00
max     88.98 100.00  27.74   1.00   0.87   8.78 100.00  12.13  24.00 711.00

       PTRATIO      B  LSTAT  PRICE
count   506.00 506.00 506.00 506.00
mean     18.46 356.67  12.65  22.53
std       2.16  91.29   7.14   9.20
min      12.60   0.32   1.73   5.00
25%      17.40 375.38   6.95  17.02
50%      19.05 391.44  11.36  21.20
75%      20.20 396.23  16.96  25.00
max      22.00 396.90  37.97  50.00
```

`CHAS` shows whether the home is next to the Charles River or not. As such, it only has the value 0 or 1. This kind of feature is also known as a dummy variable.

The average price of a Boston home in the 1970s was 22.53 or $22,530. We've experienced a lot of inflation and house price appreciation since then!

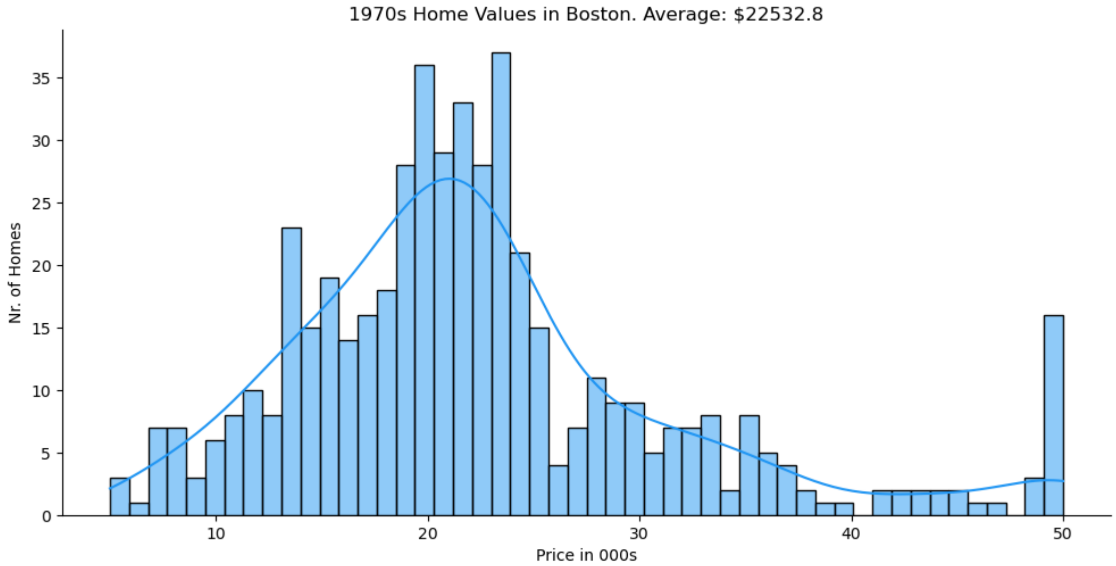
3.3 Visualise the Features

Having looked at some descriptive statistics,visualizing the data for our model. Using a bar chart and superimpose the Kernel Density Estimate (KDE) for the following variables: * PRICE: The home price in thousands. * RM: the average number of rooms per owner unit. * DIS: the weighted distance to the 5 Boston employment centres i.e., the estimated length of the commute. * RAD: the index of accessibility to highways.

What do you notice in the distributions of the data?

House Prices

[58]:
```python
sns.displot(data['PRICE'],
            bins=50,
            aspect=2,
            kde=True,
            color='#2196f3')

plt.title(f'1970s Home Values in Boston. Average: ${(1000*data.PRICE.mean()):.
 ↪6}')
plt.xlabel('Price in 000s')
plt.ylabel('Nr. of Homes')

plt.show()
```



Note there is a spike in the number homes at the very right tail at the $50,000 mark.

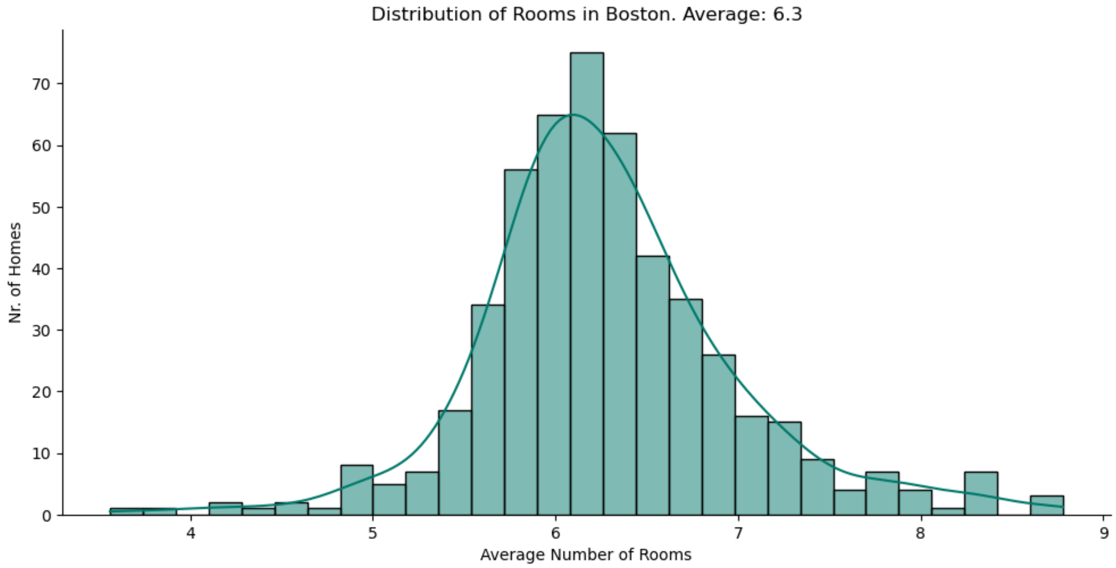
Distance to Employment - Length of Commute

[59]:
```
sns.displot(data.DIS,
            bins=50,
            aspect=2,
            kde=True,
            color='darkblue')

plt.title(f'Distance to Employment Centres. Average: {(data.DIS.mean()):.2}')
plt.xlabel('Weighted Distance to 5 Boston Employment Centres')
plt.ylabel('Nr. of Homes')

plt.show()
```

Most homes are about 3.8 miles away from work. There are fewer and fewer homes the further out we go.

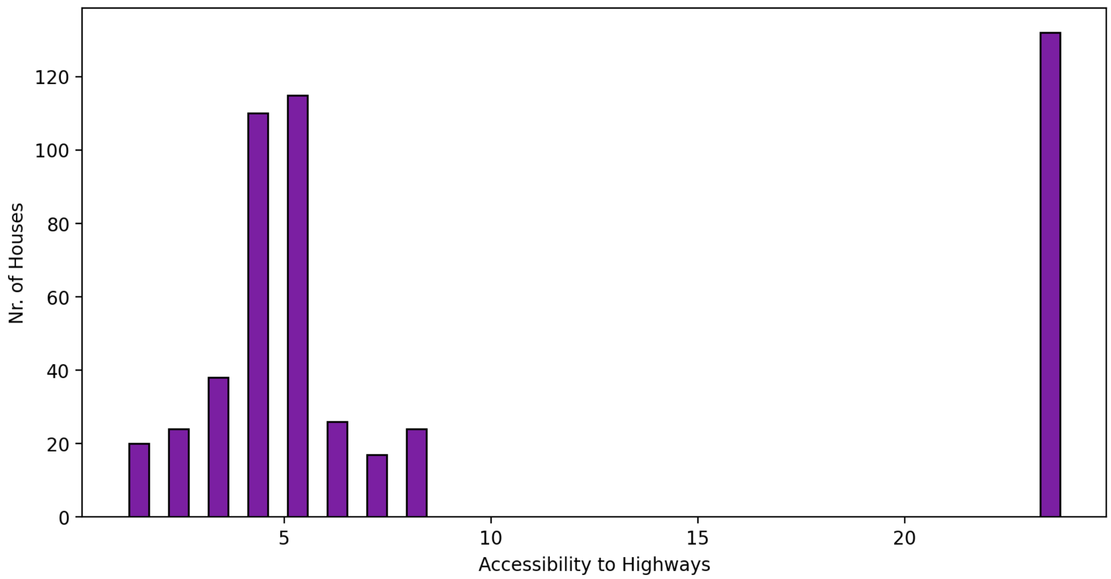
Number of Rooms

[60]:
```
sns.displot(data.RM,
            aspect=2,
            kde=True,
            color='#00796b')

plt.title(f'Distribution of Rooms in Boston. Average: {data.RM.mean():.2}')
plt.xlabel('Average Number of Rooms')
plt.ylabel('Nr. of Homes')

plt.show()
```



Access to Highways

[61]:
```
plt.figure(figsize=(10, 5), dpi=200)

plt.hist(data['RAD'],
         bins=24,
         ec='black',
         color='#7b1fa2',
         rwidth=0.5)

plt.xlabel('Accessibility to Highways')
plt.ylabel('Nr. of Houses')
plt.show()
```



RAD is an index of accessibility to roads. Better access to a highway is represented by a higher number. There's a big gap in the values of the index.

Next to the River?

[62]:
```
river_access = data['CHAS'].value_counts()

bar = px.bar(x=['No', 'Yes'],
             y=river_access.values,
             color=river_access.values,
             color_continuous_scale=px.colors.sequential.haline,
             title='Next to Charles River?')

bar.update_layout(xaxis_title='Property Located Next to the River?',
                  yaxis_title='Number of Homes',
                  coloraxis_showscale=False)
bar.show()
```

We see that out of the total number of 506 homes, only 35 are located next to the Charles River.

4 Understand the Relationships in the Data

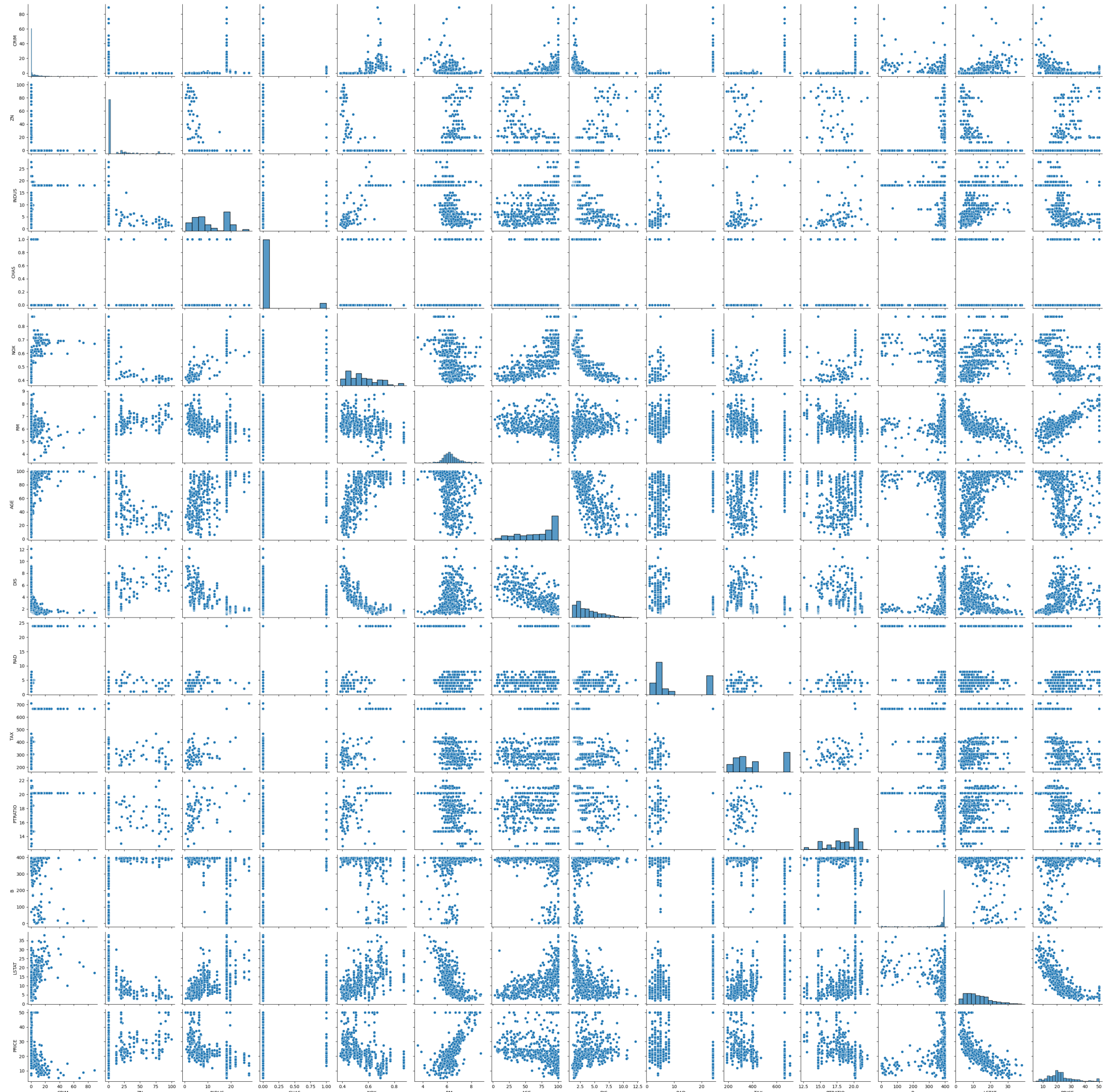
4.0.1 Run a Pair Plot

There might be some relationships in the data that we should know about. Before you run the code, make some predictions:

- What would you expect the relationship to be between pollution (NOX) and the distance to employment (DIS)?

- What kind of relationship do you expect between the number of rooms (RM) and the home value (PRICE)?
- What about the amount of poverty in an area (LSTAT) and home prices?

[63]:
```python
sns.pairplot(data)



# sns.pairplot(data, kind='reg', plot_kws={'line_kws':{'color': 'cyan'}})
plt.show()
```



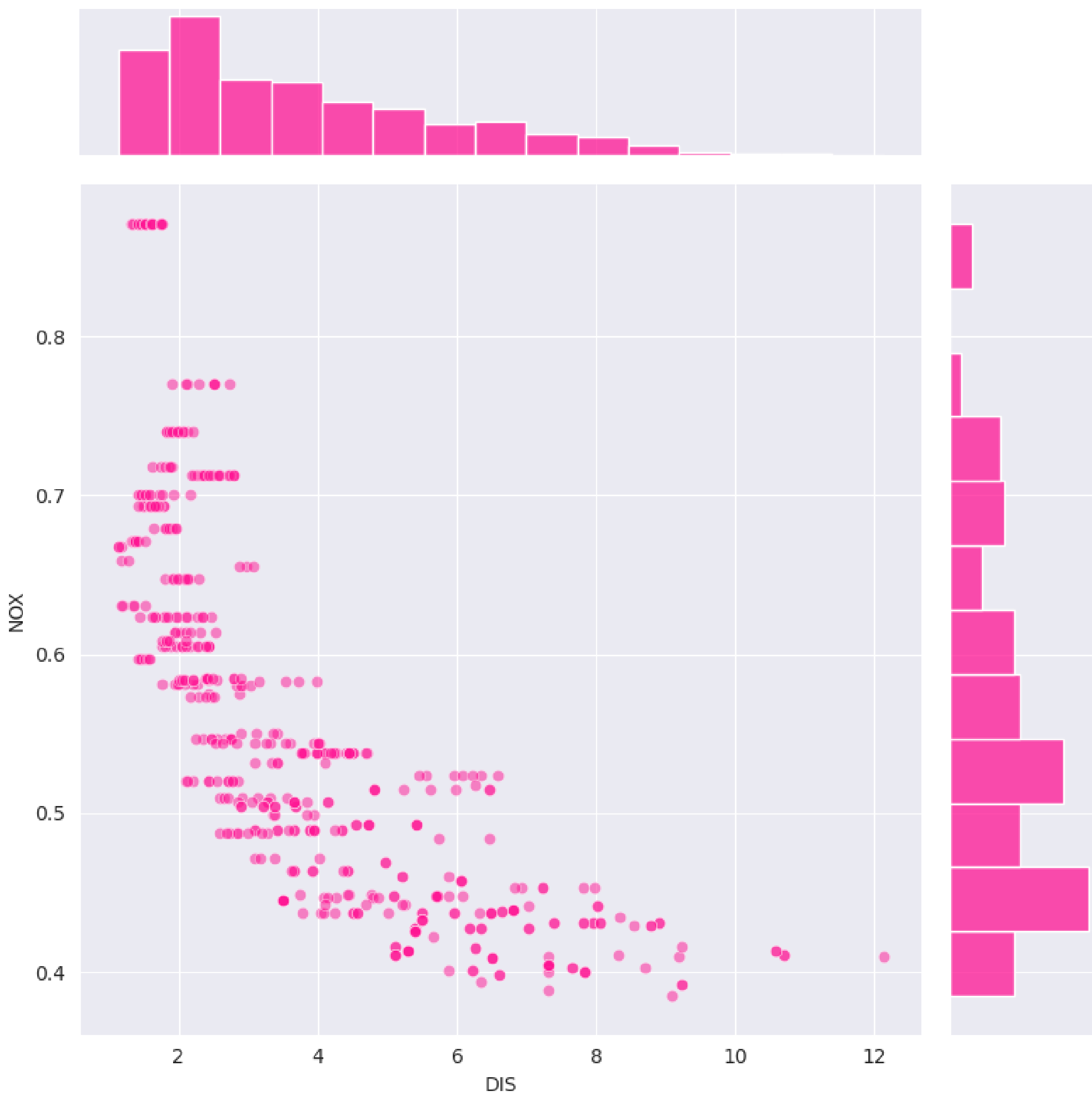
We see that we get back a grid. You might have to zoom in or squint a bit, but there are scatterplots between all the columns in our dataset. And down the diagonal in the middle, we get histograms for all our columns.

look at some of the relationships in more detail. Create a jointplot for:

- DIS and NOX
- INDUS vs NOX
- LSTAT vs RM
- LSTAT vs PRICE
- RM vs PRICE

Distance from Employment vs. Pollution Does pollution go up or down as the distance increases?

[64]:
```
with sns.axes_style('darkgrid'):
  sns.jointplot(x=data['DIS'],
                y=data['NOX'],
                height=8,
                kind='scatter',
                color='deeppink',
                joint_kws={'alpha':0.5})

plt.show()
```

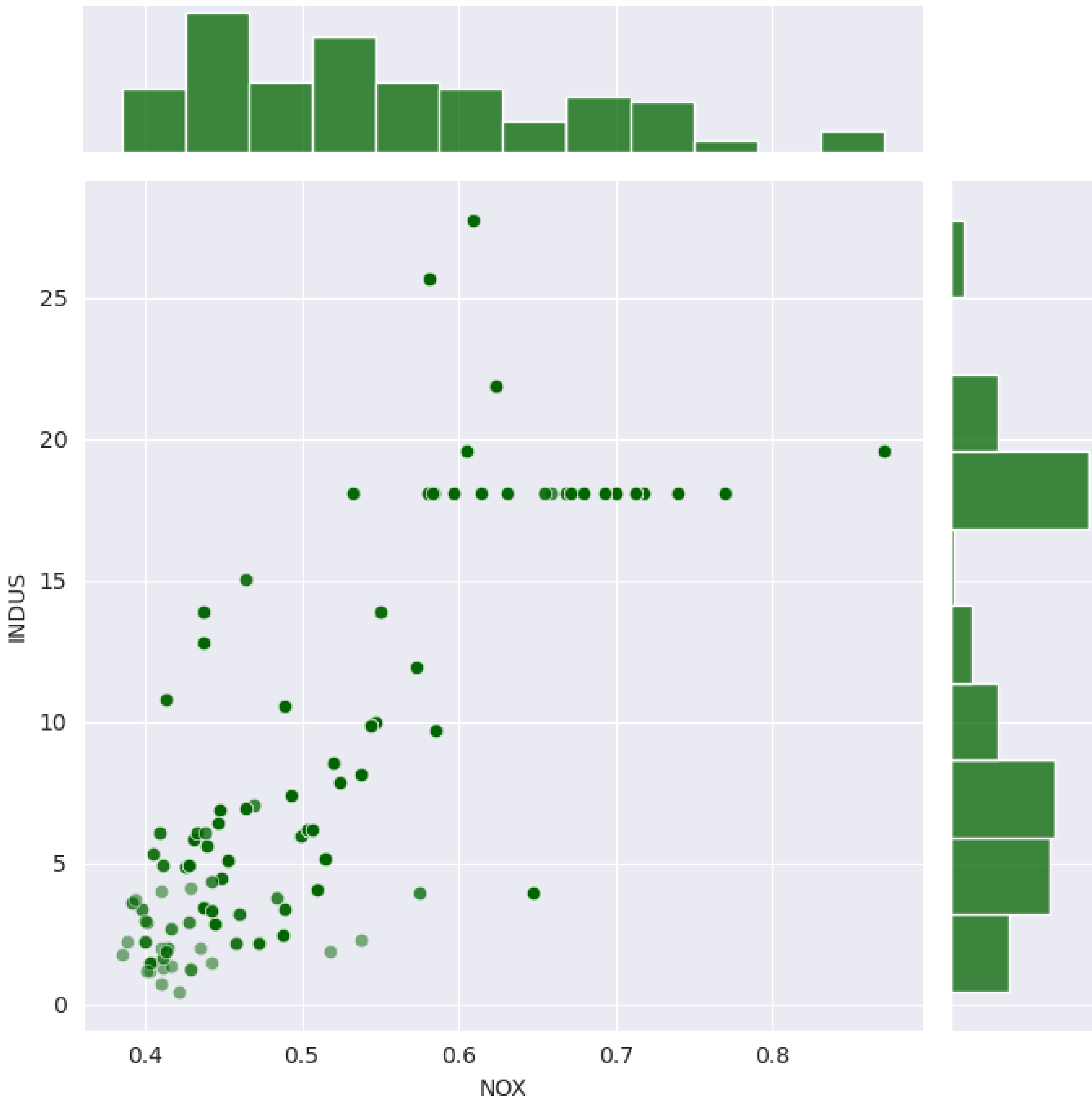


We see that pollution goes down as we go further and further out of town. This makes intuitive sense. However, even at the same distance of 2 miles to employment centres, we can get very different levels of pollution. By the same token, DIS of 9 miles and 12 miles have very similar levels of pollution.

Proportion of Non-Retail Industry versus Pollution Does pollution go up or down as there is a higher proportion of industry?

[65]:
```
with sns.axes_style('darkgrid'):
  sns.jointplot(x=data.NOX,
                y=data.INDUS,
                # kind='hex',
                height=7,
```

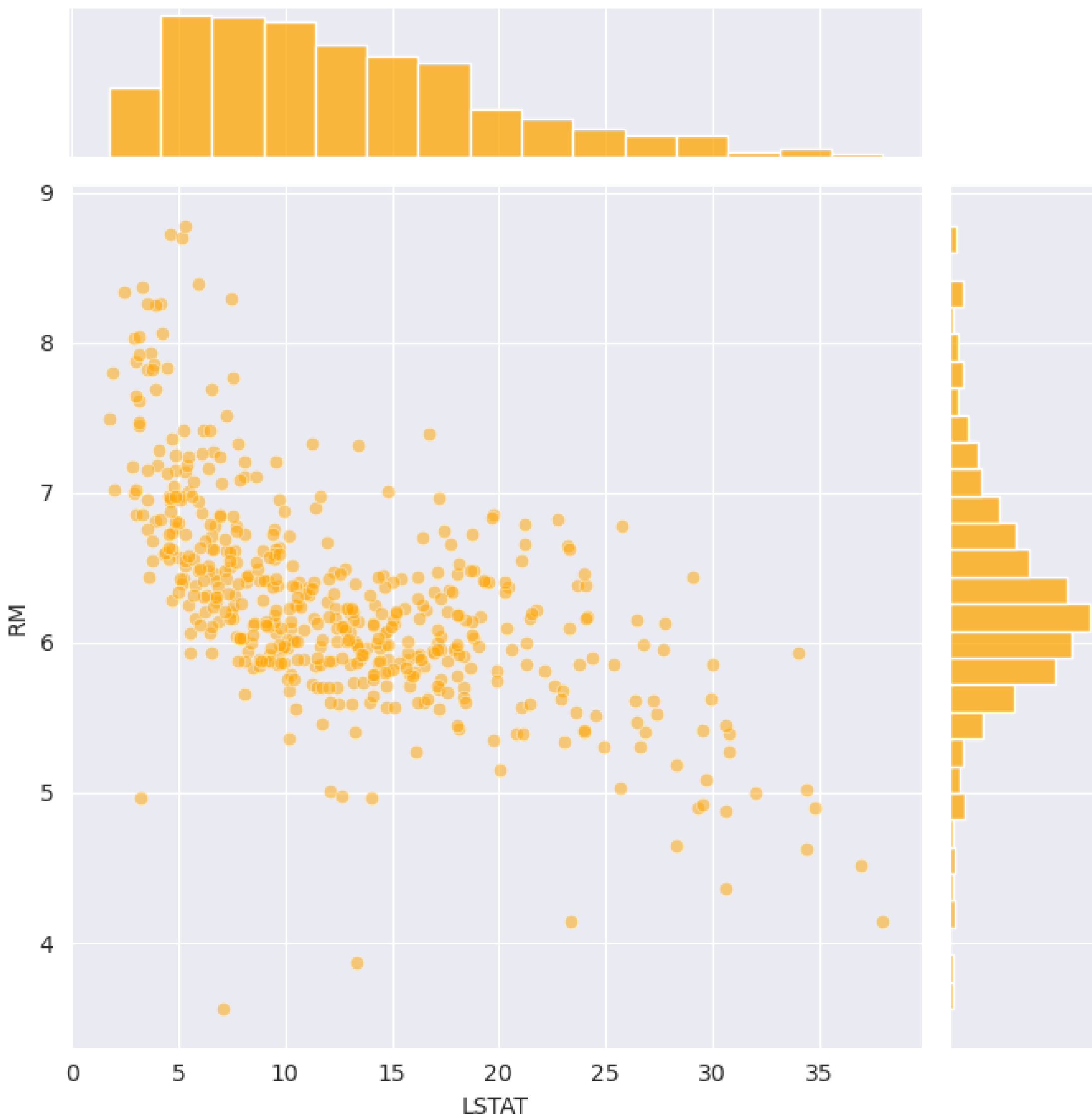
```
                color='darkgreen',
                joint_kws={'alpha':0.5})
plt.show()
```



% of Lower Income Population vs Average Number of Rooms How does the number of rooms per dwelling vary with the poverty of area? Do homes have more or fewer rooms when LSTAT is low?

[66]:
```
with sns.axes_style('darkgrid'):
    sns.jointplot(x=data['LSTAT'],
                  y=data['RM'],
                  # kind='hex',
```

```
                height=7,
                color='orange',
                joint_kws={'alpha':0.5})
plt.show()
```

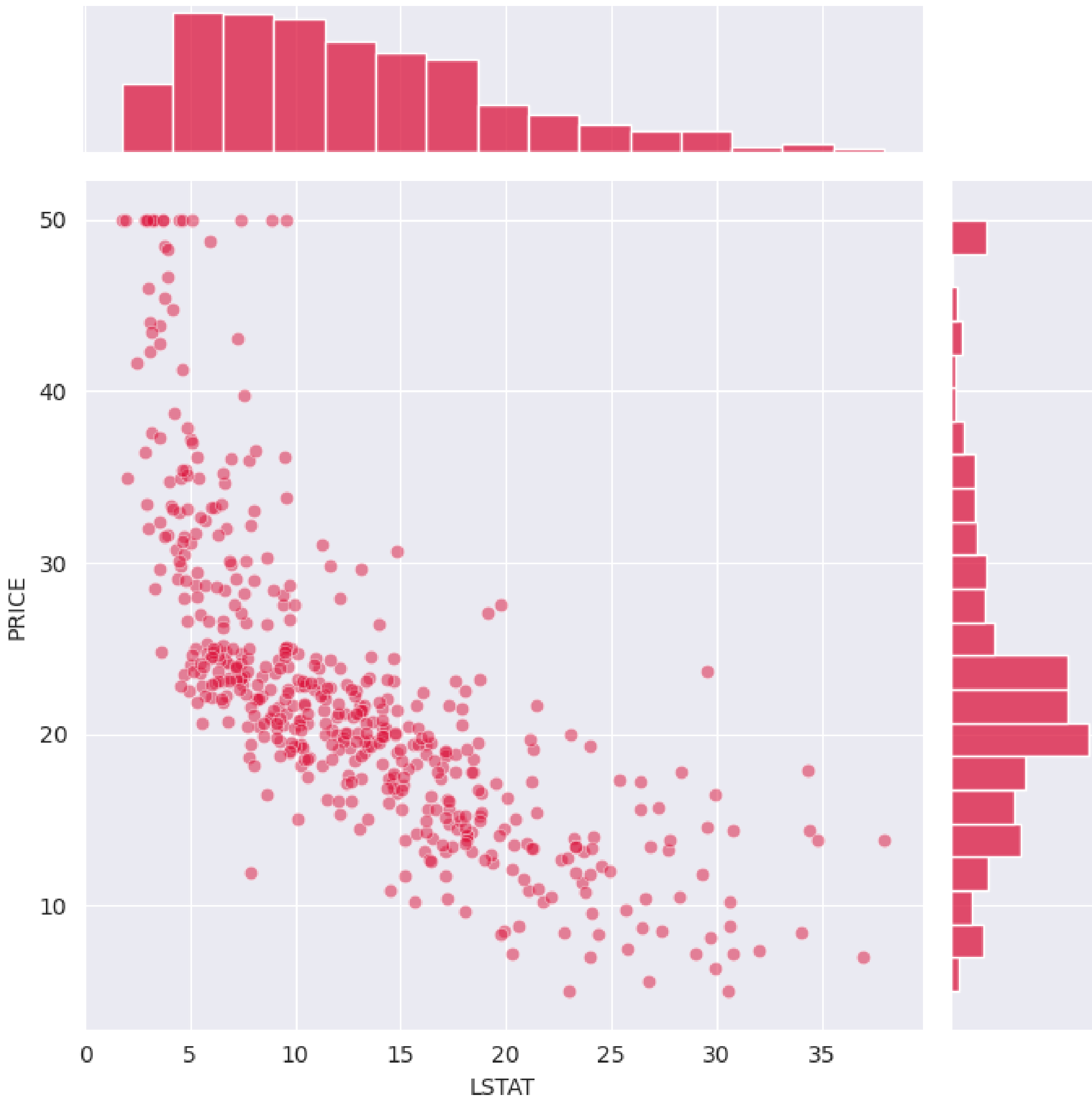


In the top left corner we see that all the homes with 8 or more rooms, LSTAT is well below 10%.

% of Lower Income Population versus Home Price How does the proportion of the lower-income population in an area affect home prices?

[67]:
```
with sns.axes_style('darkgrid'):
  sns.jointplot(x=data.LSTAT,
```

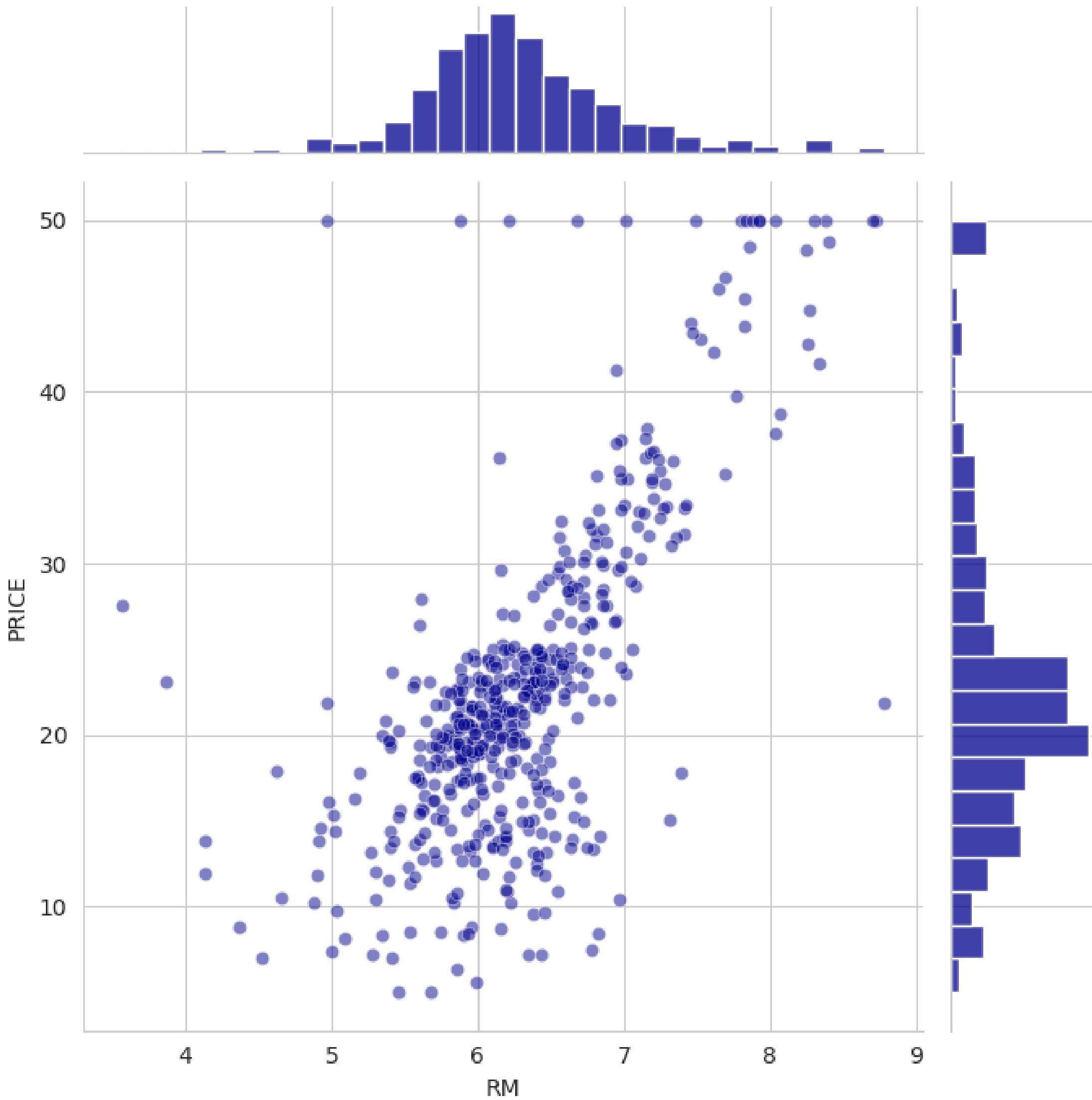
```
                y=data.PRICE,
                # kind='hex',
                height=7,
                color='crimson',
                joint_kws={'alpha':0.5})
plt.show()
```



Number of Rooms versus Home Value You can probably guess how the number of rooms affects home prices.

[68]:
```
with sns.axes_style('whitegrid'):
  sns.jointplot(x=data.RM,
```

```
                 y=data.PRICE,
                 height=7,
                 color='darkblue',
                 joint_kws={'alpha':0.5})
plt.show()
```



Again, we see those homes at the $50,000 mark all lined up at the top of the chart. Perhaps there was some sort of cap or maximum value imposed during data collection.

5 Split Training & Test Dataset

We *can't* use all 506 entries in our dataset to train our model. The reason is that we want to evaluate our model on data that it hasn't seen yet (i.e., out-of-sample data). That way we can get a better idea of its performance in the real world.

note, our **target** is our home PRICE, and our **features** are all the other columns we'll use to predict the price.

[69]:
```
target = data['PRICE']
features = data.drop('PRICE', axis=1)

X_train, X_test, y_train, y_test = train_test_split(features,
                                                    target,
                                                    test_size=0.2,
                                                    random_state=10)
```

[70]:
```
# % of training set
train_pct = 100*len(X_train)/len(features)
print(f'Training data is {train_pct:.3}% of the total data.')

# % of test data set
test_pct = 100*X_test.shape[0]/features.shape[0]
print(f'Test data makes up the remaining {test_pct:0.3}%.')
```

```
Training data is 79.8% of the total data.
Test data makes up the remaining 20.2%.
```

6 Multivariable Regression

$$P R \hat{I} C E = \theta_0 + \theta_1 RM + \theta_2 NOX + \theta_3 DIS + \theta_4 CHAS ... + \theta_{13} LSTAT$$

6.0.1 Running our Regression

How high is the r-squared for the regression on the training data?

[71]:
```
regr = LinearRegression()
regr.fit(X_train, y_train)
rsquared = regr.score(X_train, y_train)

print(f'Training data r-squared: {rsquared:.2}')
```

```
Training data r-squared: 0.75
```

0.75 is a very high r-squared!

6.0.2 Evaluate the Coefficients of the Model

Here we do a sense check on our regression coefficients. The first thing to look for is if the coefficients have the expected sign (positive or negative).

- We already saw that RM on its own had a positive relation to PRICE based on the scatter plot. Is RM's coefficient also positive?
- What is the sign on the LSAT coefficient? Does it match our intuition and the scatter plot above?

- Check the other coefficients. Do they have the expected sign?
- Based on the coefficients, how much more expensive is a room with 6 rooms compared to a room with 5 rooms? According to the model, what is the premium you would have to pay for an extra room?

[72]:
```
regr_coef = pd.DataFrame(data=regr.coef_, index=X_train.columns,␣
 ↪columns=['Coefficient'])
regr_coef
```

[72]:

	Coefficient
CRIM	-0.13
ZN	0.06
INDUS	-0.01
CHAS	1.97
NOX	-16.27
RM	3.11
AGE	0.02
DIS	-1.48
RAD	0.30
TAX	-0.01
PTRATIO	-0.82
B	0.01
LSTAT	-0.58

[73]:
```
# Premium for having an extra room
premium = regr_coef.loc['RM'].values[0] * 1000  # i.e., ~3.11 * 1000
print(f'The price premium for having an extra room is ${premium:.5}')
```

```
The price premium for having an extra room is $3108.5
```

6.0.3 Analyzing the Estimated Values & Regression Residuals

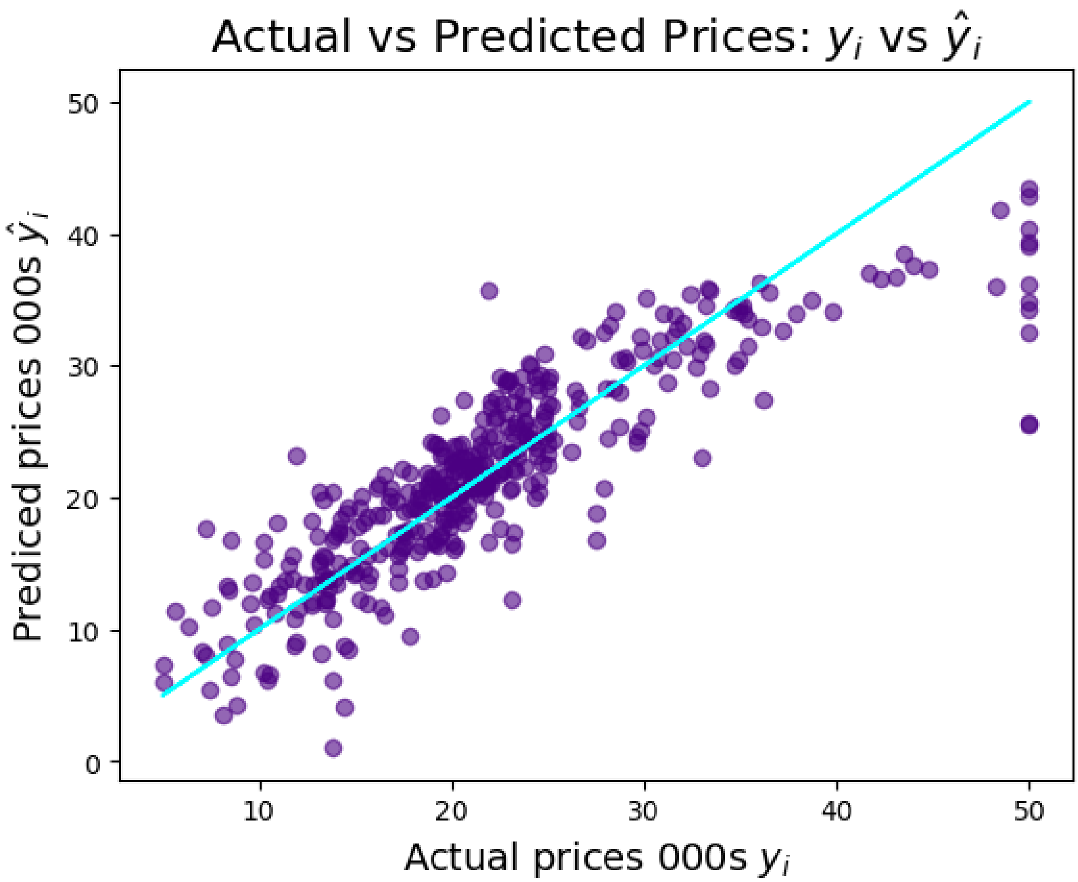
The next step is to evaluate our regression. How good our regression is depends not only on the r-squared. It also depends on the **residuals** - the difference between the model's predictions ($\hat{y}_i$) and the true values (y_i) inside `y_train`.

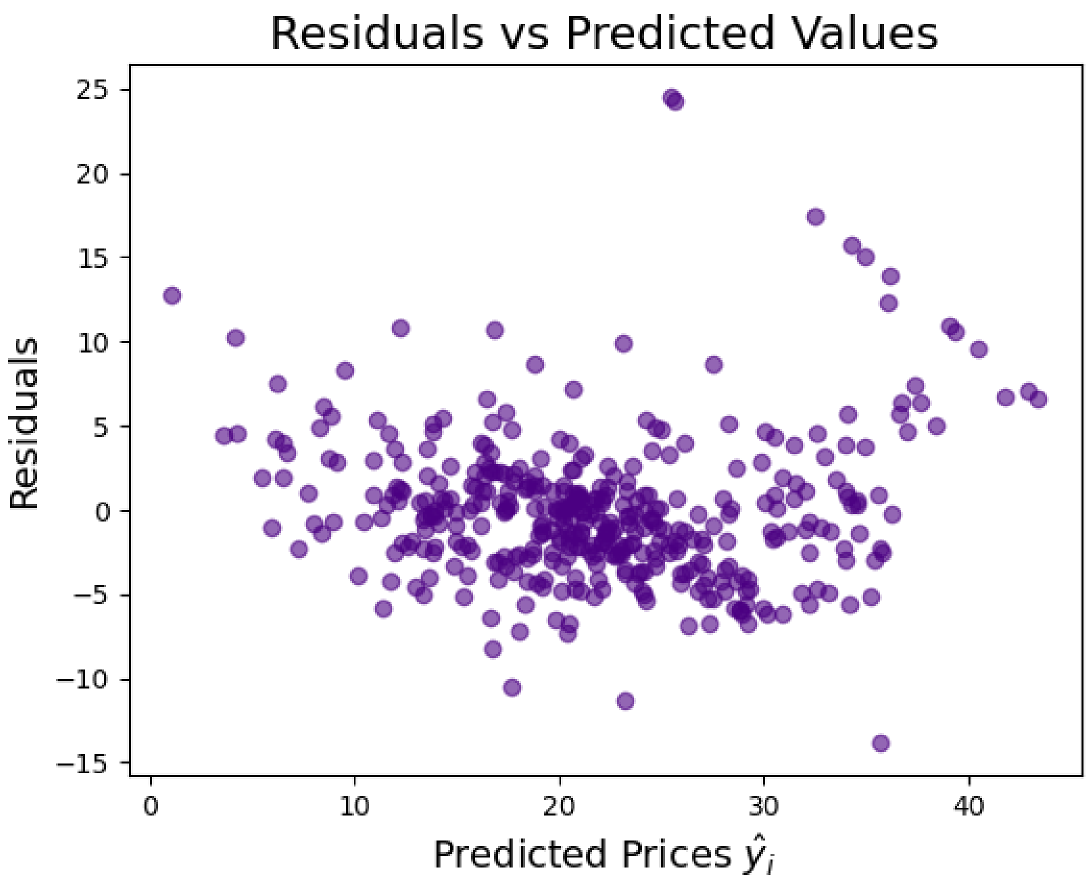
```
predicted_values = regr.predict(X_train)
residuals = (y_train - predicted_values)
```

[74]:
```
predicted_vals = regr.predict(X_train)
residuals = (y_train - predicted_vals)
```

[75]:
```
# Original Regression of Actual vs. Predicted Prices
plt.figure(dpi=100)
plt.scatter(x=y_train, y=predicted_vals, c='indigo', alpha=0.6)
plt.plot(y_train, y_train, color='cyan')
plt.title(f'Actual vs Predicted Prices: $y _i$ vs $\hat y_i$', fontsize=17)
plt.xlabel('Actual prices 000s $y _i$', fontsize=14)
plt.ylabel('Prediced prices 000s $\hat y _i$', fontsize=14)
```

```python
plt.show()

# Residuals vs Predicted values
plt.figure(dpi=100)
plt.scatter(x=predicted_vals, y=residuals, c='indigo', alpha=0.6)
plt.title('Residuals vs Predicted Values', fontsize=17)
plt.xlabel('Predicted Prices $\hat y _i$', fontsize=14)
plt.ylabel('Residuals', fontsize=14)
plt.show()
```



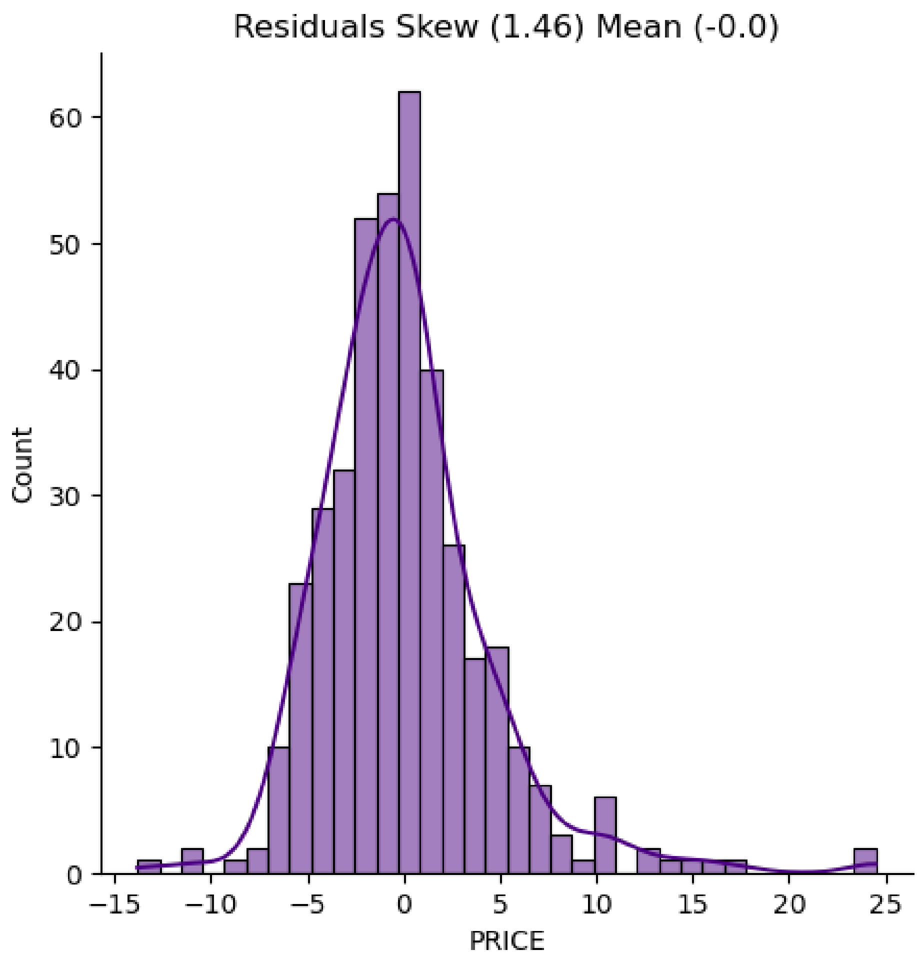


Why do we want to look at the residuals? We want to check that they look random. Why? The residuals represent the errors of our model. If there's a pattern in our errors, then our model has a systematic bias.

We can analyse the distribution of the residuals. In particular, we're interested in the **skew** and the **mean**.

- the mean and the skewness of the residuals.
- Is the skewness different from zero? If so, by how much?
- Is the mean different from zero?

[76]:
```
# Residual Distribution Chart
resid_mean = round(residuals.mean(), 2)
resid_skew = round(residuals.skew(), 2)

sns.displot(residuals, kde=True, color='indigo')
plt.title(f'Residuals Skew ({resid_skew}) Mean ({resid_mean})')
plt.show()
```

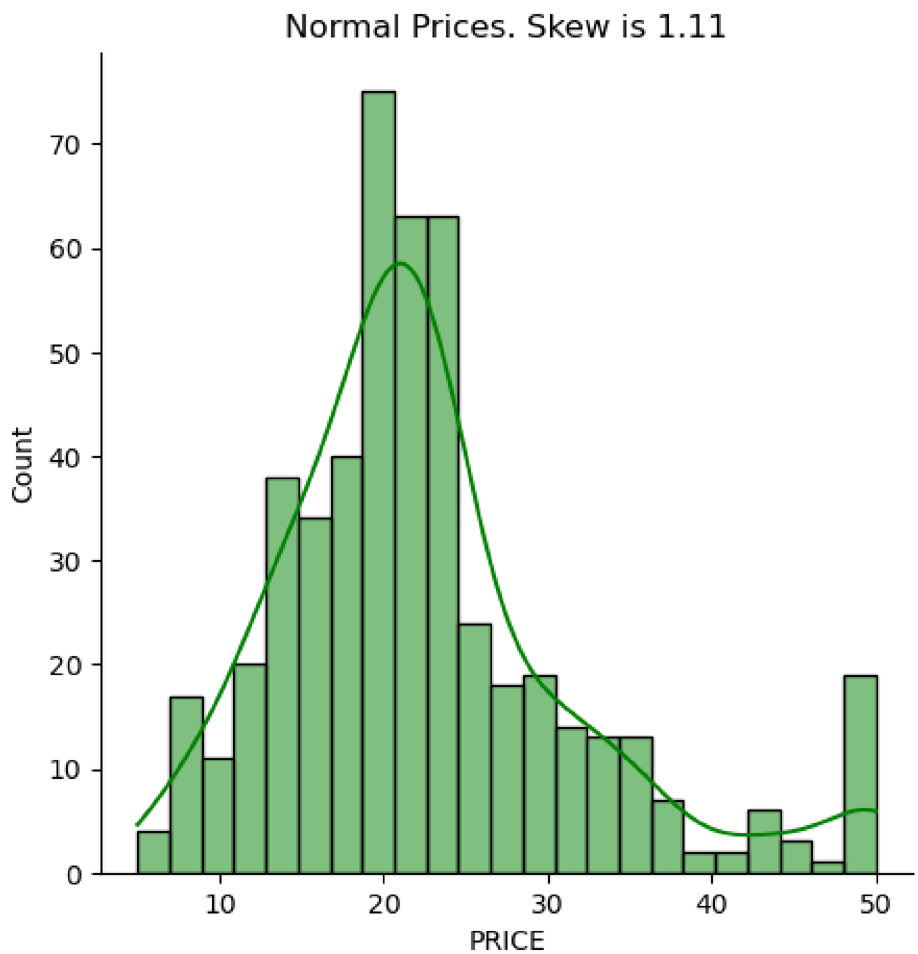


We see that the residuals have a skewness of 1.46. There could be some room for improvement here.

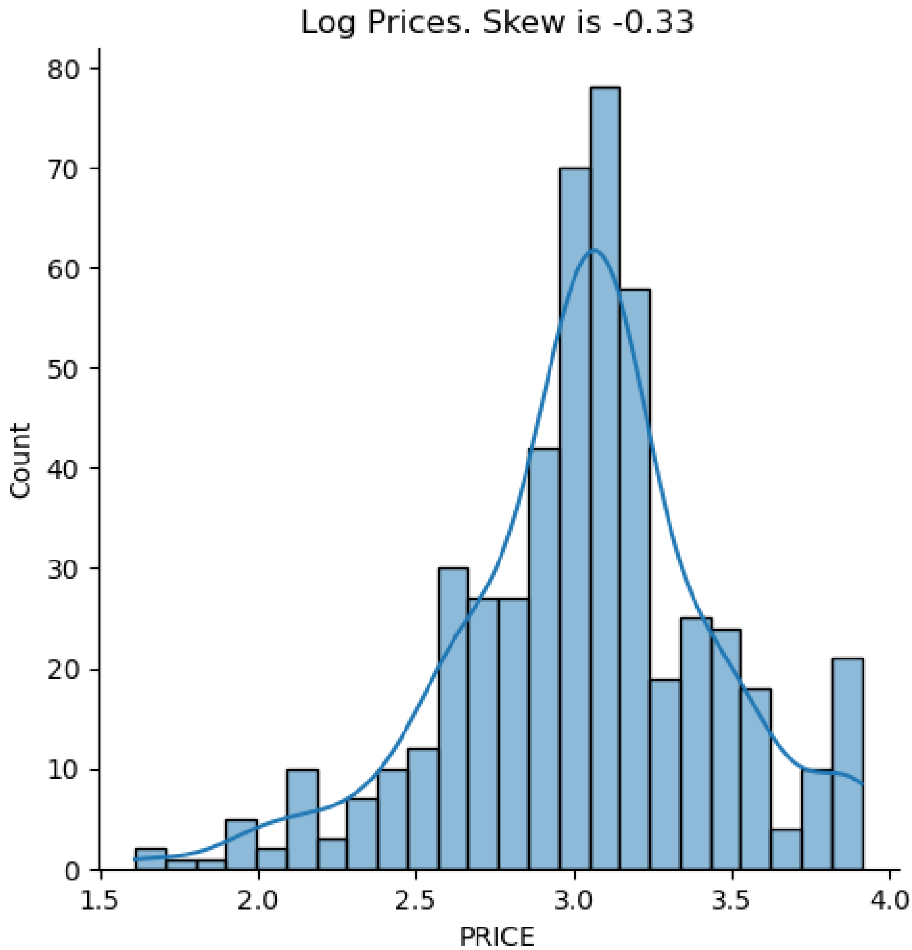
6.0.4 Data Transformations for a Better Fit

Transform our data to make it fit better with our linear model.

[77]:
```
tgt_skew = data['PRICE'].skew()
sns.displot(data['PRICE'], kde='kde', color='green')
plt.title(f'Normal Prices. Skew is {tgt_skew:.3}')
plt.show()
```

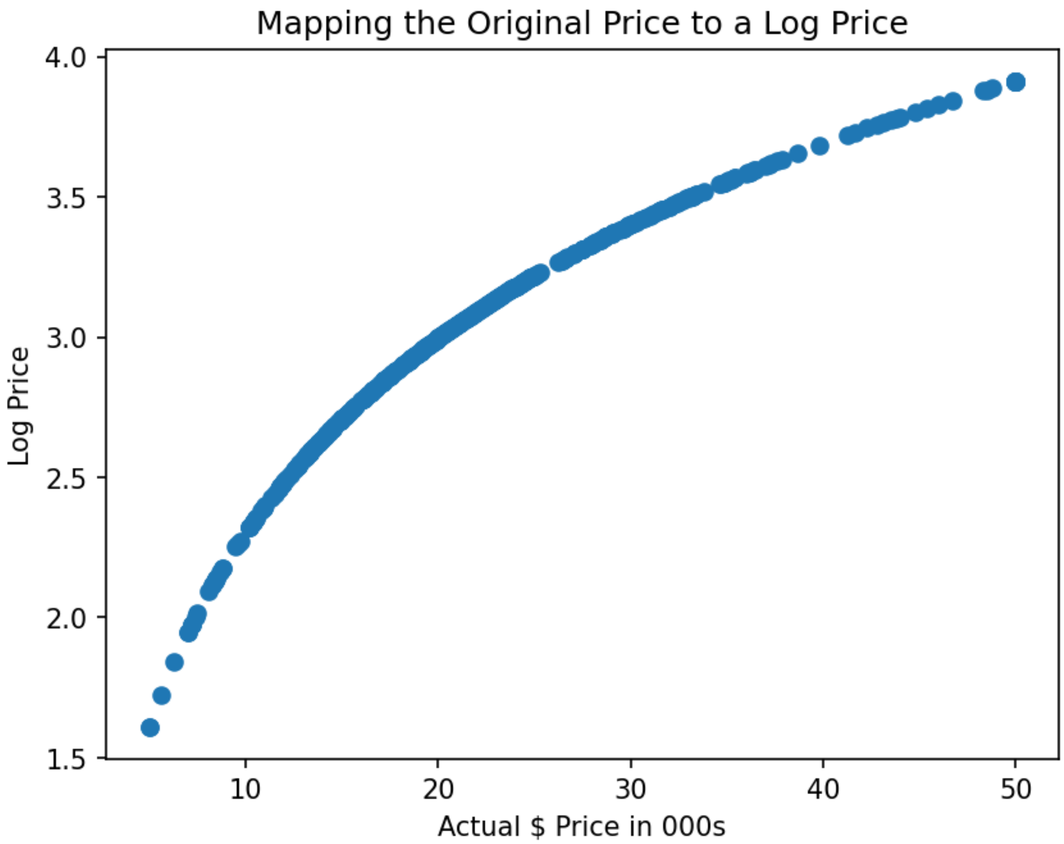


[78]:
```
y_log = np.log(data['PRICE'])
sns.displot(y_log, kde=True)
plt.title(f'Log Prices. Skew is {y_log.skew():.3}')
plt.show()
```



The log prices have a skew that's closer to zero. This makes them a good candidate for use in our linear model. Perhaps using log prices will improve our regression's r-squared and our model's residuals.

[79]:
```
plt.figure(dpi=150)
plt.scatter(data.PRICE, np.log(data.PRICE))

plt.title('Mapping the Original Price to a Log Price')
plt.ylabel('Log Price')
plt.xlabel('Actual $ Price in 000s')
plt.show()
```



6.1 Regression using Log Prices

Using log prices instead, our model has changed to:

$$\log(PR\hat{I}CE) = \theta_0 + \theta_1 RM + \theta_2 NOX + \theta_3 DIS + \theta_4 CHAS + ... + \theta_{13} LSTAT$$

- What is the r-squared of the regression on the training data?
- Have we improved the fit of our model compared to before based on this measure?

[80]:
```
new_target = np.log(data['PRICE']) # Use log prices
features = data.drop('PRICE', axis=1)

X_train, X_test, log_y_train, log_y_test = train_test_split(features,
                                                    new_target,
                                                    test_size=0.2,
                                                    random_state=10)

log_regr = LinearRegression()
log_regr.fit(X_train, log_y_train)
log_rsquared = log_regr.score(X_train, log_y_train)
```

```
log_predictions = log_regr.predict(X_train)
log_residuals = (log_y_train - log_predictions)

print(f'Training data r-squared: {log_rsquared:.2}')
```

```
Training data r-squared: 0.79
```

This time we got an r-squared of 0.79 compared to 0.75. This looks like a promising improvement.

6.2 Evaluating Coefficients with Log Prices

the coefficients of the new regression model.

- Do the coefficients still have the expected sign?
- Is being next to the river a positive based on the data?
- How does the quality of the schools affect property prices? What happens to prices as there are more students per teacher?

[81]:
```
df_coef = pd.DataFrame(data=log_regr.coef_, index=X_train.columns,␣
 ↪columns=['coef'])
df_coef
```

[81]:

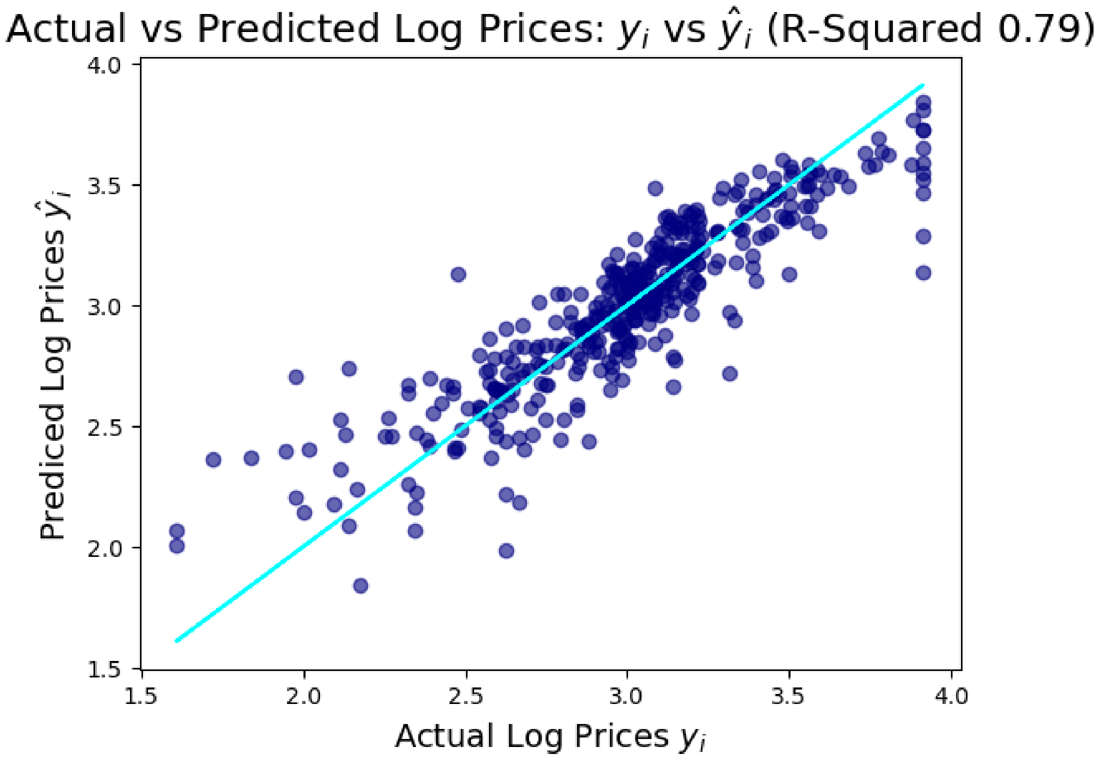
	coef
CRIM	-0.01
ZN	0.00
INDUS	0.00
CHAS	0.08
NOX	-0.70
RM	0.07
AGE	0.00
DIS	-0.05
RAD	0.01
TAX	-0.00
PTRATIO	-0.03
B	0.00
LSTAT	-0.03

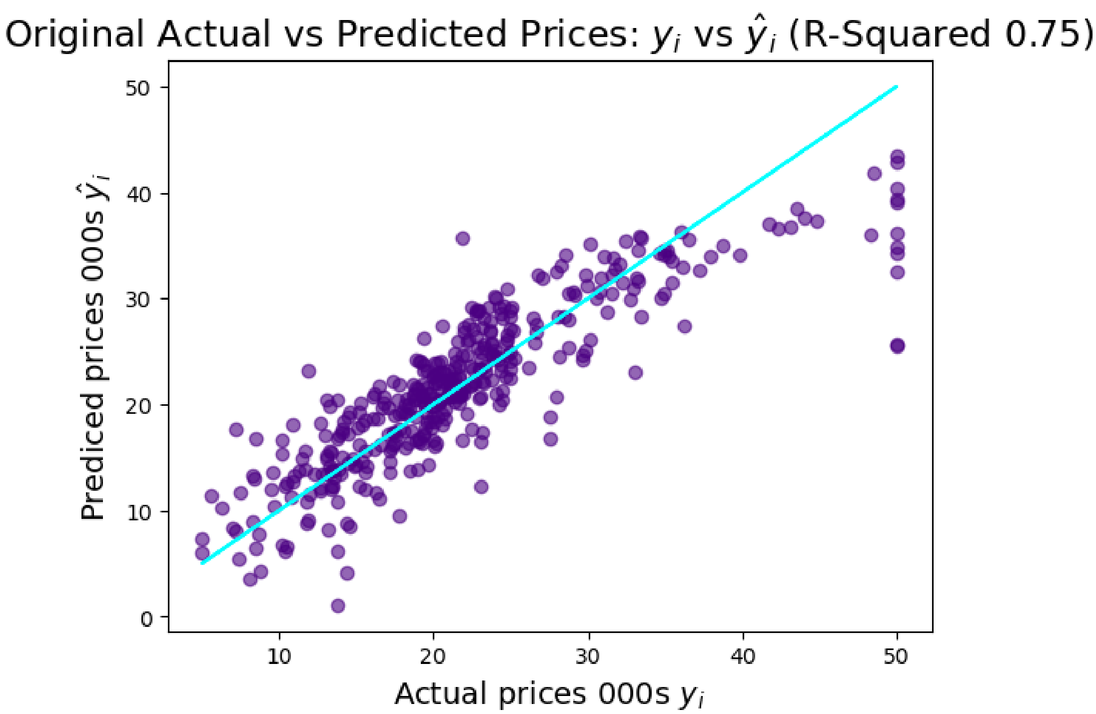
So how can we interpret the coefficients? The key thing we look for is still the sign - being close to the river results in higher property prices because CHAS has a coefficient greater than zero. Therefore property prices are higher next to the river.

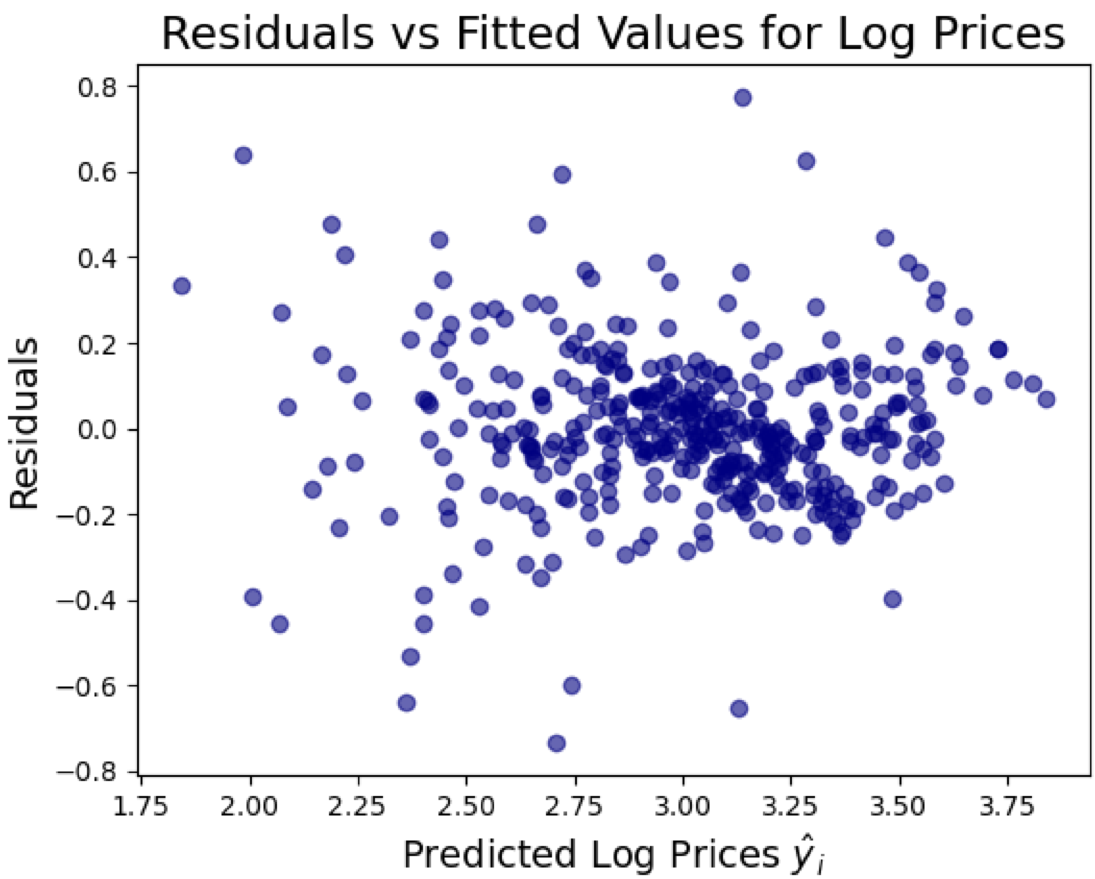
More students per teacher - a higher PTRATIO - is a clear negative. Smaller classroom sizes are indicative of higher quality education, so have a negative coefficient for PTRATIO.

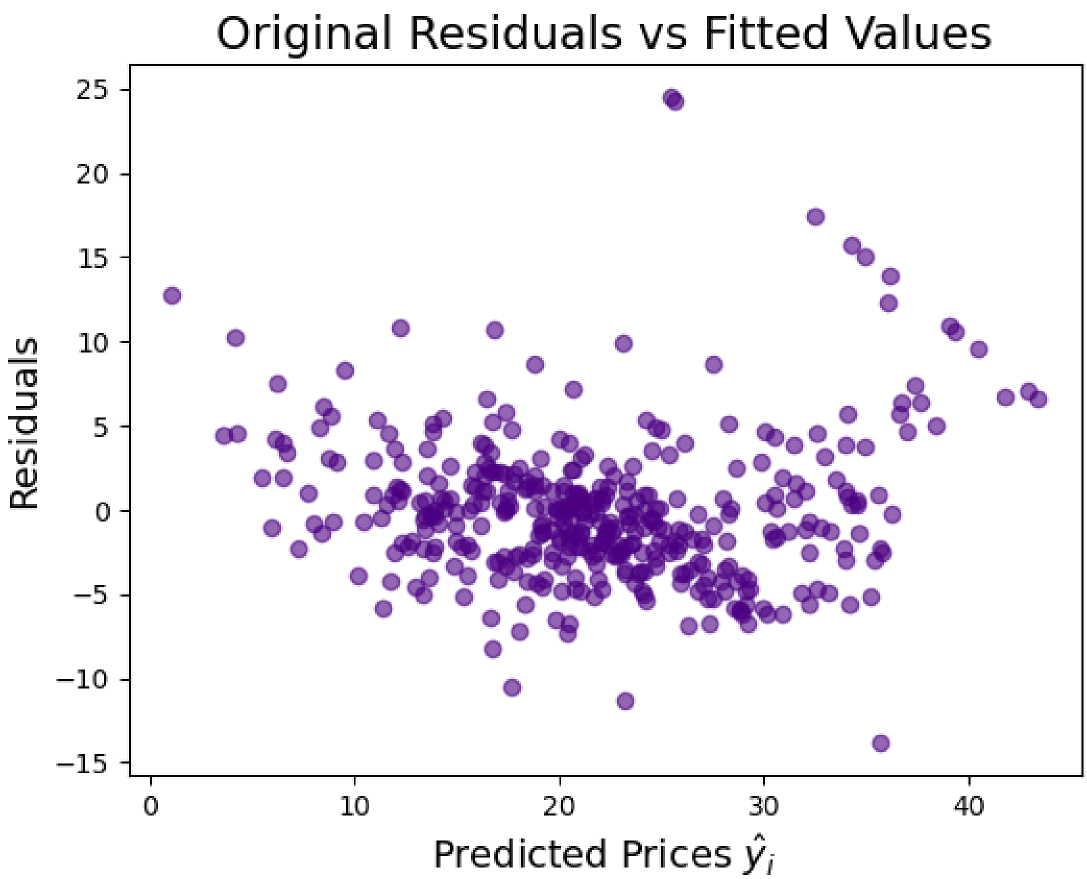
6.3 Regression with Log Prices & Residual Plots

[82]:
```
# Graph of Actual vs. Predicted Log Prices
plt.scatter(x=log_y_train, y=log_predictions, c='navy', alpha=0.6)
plt.plot(log_y_train, log_y_train, color='cyan')
plt.title(f'Actual vs Predicted Log Prices: $y _i$ vs $\hat y_i$ (R-Squared␣
 ↪{log_rsquared:.2})', fontsize=17)
plt.xlabel('Actual Log Prices $y _i$', fontsize=14)
plt.ylabel('Prediced Log Prices $\hat y _i$', fontsize=14)
plt.show()

# Original Regression of Actual vs. Predicted Prices
plt.scatter(x=y_train, y=predicted_vals, c='indigo', alpha=0.6)
plt.plot(y_train, y_train, color='cyan')
plt.title(f'Original Actual vs Predicted Prices: $y _i$ vs $\hat y_i$␣
 ↪(R-Squared {rsquared:.3})', fontsize=17)
plt.xlabel('Actual prices 000s $y _i$', fontsize=14)
plt.ylabel('Prediced prices 000s $\hat y _i$', fontsize=14)
plt.show()

# Residuals vs Predicted values (Log prices)
plt.scatter(x=log_predictions, y=log_residuals, c='navy', alpha=0.6)
plt.title('Residuals vs Fitted Values for Log Prices', fontsize=17)
plt.xlabel('Predicted Log Prices $\hat y _i$', fontsize=14)
plt.ylabel('Residuals', fontsize=14)
plt.show()

# Residuals vs Predicted values
plt.scatter(x=predicted_vals, y=residuals, c='indigo', alpha=0.6)
plt.title('Original Residuals vs Fitted Values', fontsize=17)
plt.xlabel('Predicted Prices $\hat y _i$', fontsize=14)
plt.ylabel('Residuals', fontsize=14)
plt.show()
```


Actual vs Predicted Log Prices: y_i vs $\hat{y}_i$ (R-Squared 0.79)
Prediced Log Prices $\hat{y}_i$
Actual Log Prices y_i
1.5
2.0
2.5
3.0
3.5
4.0


Original Actual vs Predicted Prices: y_i vs $\hat{y}_i$ (R-Squared 0.75)
Prediced prices 000s $\hat{y}_i$
Actual prices 000s y_i
0
10
20
30
40
50

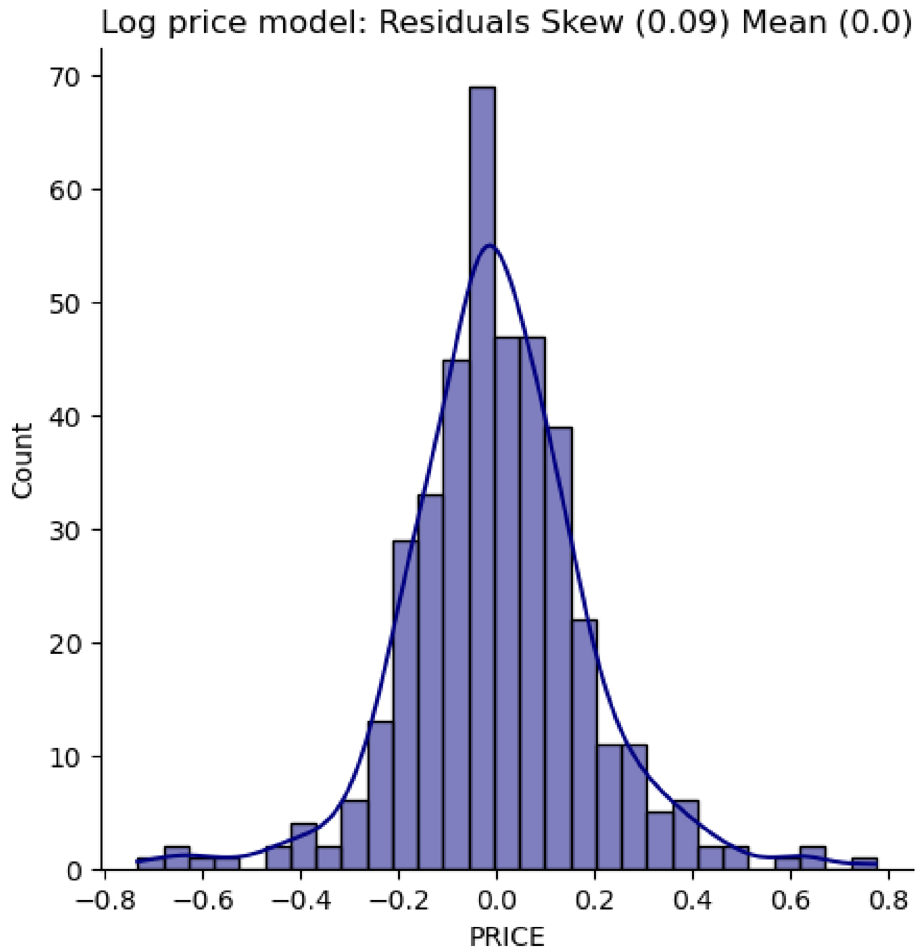

Residuals vs Fitted Values for Log Prices
Residuals
Predicted Log Prices $\hat{y}_i$
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
1.75
2.00
2.25
2.50
2.75
3.00
3.25
3.50
3.75

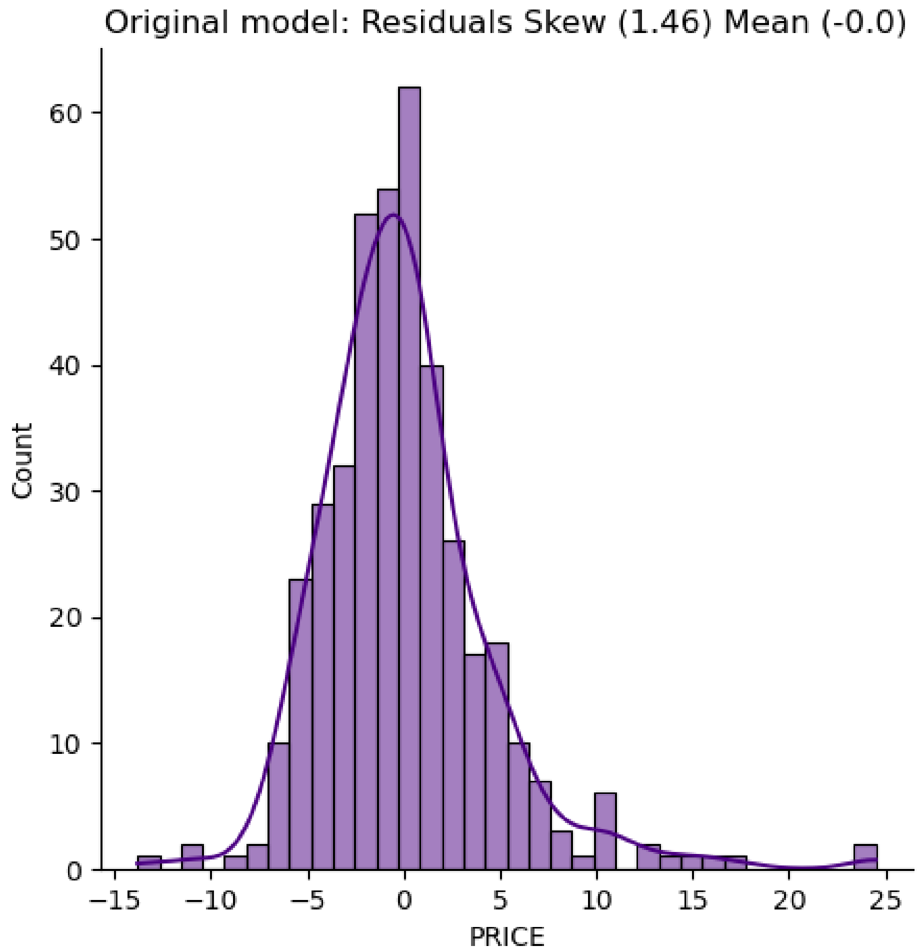


It's hard to see a difference here just by eye. The predicted values seems slightly closer to the cyan line, but eyeballing the charts is not terribly helpful in this case.

Calculate the mean and the skew for the residuals using log prices. Are the mean and skew closer to 0 for the regression using log prices?

[83]:
```
# Distribution of Residuals (log prices) - checking for normality
log_resid_mean = round(log_residuals.mean(), 2)
log_resid_skew = round(log_residuals.skew(), 2)

sns.displot(log_residuals, kde=True, color='navy')
plt.title(f'Log price model: Residuals Skew ({log_resid_skew}) Mean␣
  ↪({log_resid_mean})')
plt.show()

sns.displot(residuals, kde=True, color='indigo')
plt.title(f'Original model: Residuals Skew ({resid_skew}) Mean ({resid_mean})')
plt.show()
```


Log price model: Residuals Skew (0.09) Mean (0.0)
Count
PRICE
0
10
20
30
40
50
60
70
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8



Our new regression residuals have a skew of 0.09 compared to a skew of 1.46. The mean is still around 0. From both a residuals perspective and an r-squared perspective we have improved our model with the data transformation.

7 Compare Out of Sample Performance

Comparing the r-squared of the two models on the test dataset. Which model does better? Is the r-squared higher or lower than for the training dataset? Why?

[84]:
```
print(f'Original Model Test Data r-squared: {regr.score(X_test, y_test):.2}')
print(f'Log Model Test Data r-squared: {log_regr.score(X_test, log_y_test):.2}')
```

```
Original Model Test Data r-squared: 0.67
Log Model Test Data r-squared: 0.74
```

By definition, the model has not been optimised for the testing data. Therefore performance will be worse than on the training data. However, our r-squared still remains high, so we have built a useful model.

8 Predict a Property's Value using the Regression Coefficients

Our preferred model now has an equation that looks like this:

$$\log(P\hat{RIC}E) = \theta_0 + \theta_1 RM + \theta_2 NOX + \theta_3 DIS + \theta_4 CHAS + ... + \theta_{13} LSTAT$$

The average property has the mean value for all its charactistics:

[85]:
```
# Starting Point: Average Values in the Dataset
features = data.drop(['PRICE'], axis=1)
average_vals = features.mean().values
property_stats = pd.DataFrame(data=average_vals.reshape(1, len(features.
  ↪columns)),
                              columns=features.columns)
property_stats
```

[85]:
```
   CRIM    ZN  INDUS  CHAS  NOX   RM   AGE  DIS  RAD    TAX  PTRATIO      B  \
0  3.61 11.36  11.14  0.07 0.55 6.28 68.57 3.80 9.55 408.24    18.46 356.67

   LSTAT
0  12.65
```

Predict how much the average property is worth using the stats above. What is the log price estimate and what is the dollar estimate?

[86]:
```
# Make prediction
log_estimate = log_regr.predict(property_stats)[0]
print(f'The log price estimate is ${log_estimate:.3}')

# Convert Log Prices to Acutal Dollar Values
dollar_est = np.e**log_estimate * 1000
# or use
dollar_est = np.exp(log_estimate) * 1000
print(f'The property is estimated to be worth ${dollar_est:.6}')
```

```
The log price estimate is $3.03
The property is estimated to be worth $20703.2
```

A property with an average value for all the features has a value of $20,700.

Keeping the average values for CRIM, RAD, INDUS and others, value a property with the following characteristics:

[87]:
```
# Define Property Characteristics
next_to_river = True
nr_rooms = 8
students_per_classroom = 20
distance_to_town = 5
pollution = data.NOX.quantile(q=0.75) # high
```

```
amount_of_poverty =  data.LSTAT.quantile(q=0.25) # low
```

[88]:
```
# Solution
# Set Property Characteristics
property_stats['RM'] = nr_rooms
property_stats['PTRATIO'] = students_per_classroom
property_stats['DIS'] = distance_to_town

if next_to_river:
    property_stats['CHAS'] = 1
else:
    property_stats['CHAS'] = 0

property_stats['NOX'] = pollution
property_stats['LSTAT'] = amount_of_poverty
```

[89]:
```
# Make prediction
log_estimate = log_regr.predict(property_stats)[0]
print(f'The log price estimate is ${log_estimate:.3}')

# Convert Log Prices to Acutal Dollar Values
dollar_est = np.e**log_estimate * 1000
print(f'The property is estimated to be worth ${dollar_est:.6}')
```

```
The log price estimate is $3.25
The property is estimated to be worth $25792.0
```

[ ]:
```

```