Introduction

Perhaps the most common goal in statistics is to answer the question “Is the variable X (or more likely, X1, ..., X p) associated with a variable Y, and if so, what is the relationship and can we use it to predict Y?”

Nowhere is the nexus between statistics and data science stronger than in the realm of Prediction, specifically, the prediction of an outcome (target) variable based on the values of other “predictor” variables.

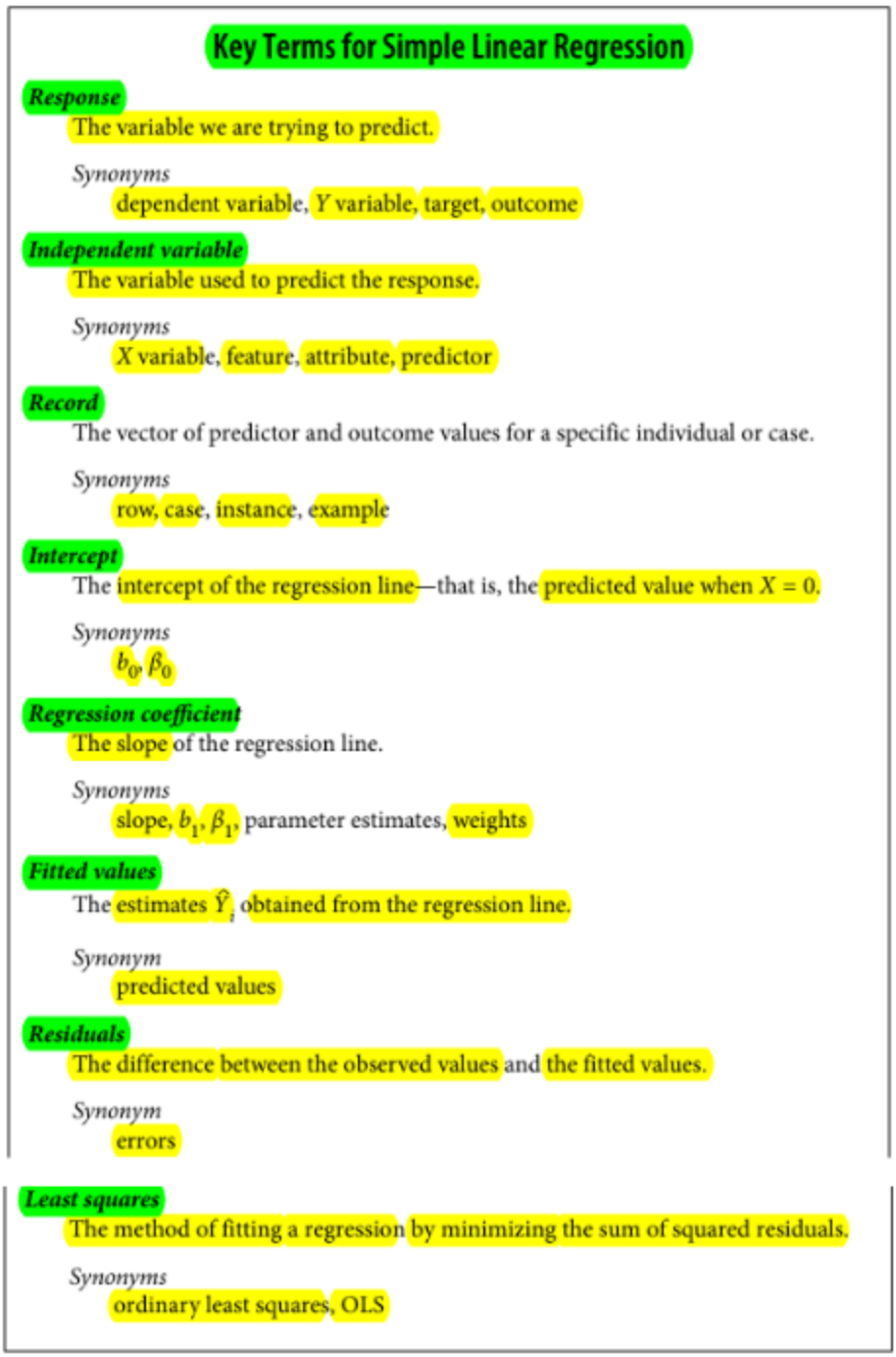
This process of ***training a model*** on **data** where the outcome is known, **for subsequent application to data where the outcome is not known**, is termed ***”supervised learning”***.

Another important connection between data science and statistics is in the area of ***anomaly detection***, where regression diagnostics originally intended for data analysis and improving the regression model can be used to detect unusual records.

I. Simple Linear RRegression

Simple linear regression provides a model of the relationship between the magnitude of one variable and that of a second—for example, as X increases, Y also increases. Or as X increases, Y decreases. Correlation is another way to measure how two variables are related.

The difference is that while **correlation measures the strength of an association between two variables (strength and direction of association)**, **regression quantifies the nature of the relationship (form, magnitude, and predictive nature of the relationship)**.



I.1. The regression Equation

Simple linear regression estimates how much Y will change when X changes by a certain amount. With the correlation coefficient, the variables X and Y are inter- changeable. With regression, we are trying to predict the Y variable from X using a linear relationship :
Y = b1x + b0

We read this as “Y equals b1 times X, plus a constant b0.”

The machine learning community tends to use other terms, calling Y the target and X a feature vector. Throughout this book, we will use the terms predictor and feature interchangeably.

Consider the scatterplot in Figure 4-1 displaying the number of years a worker was exposed to cotton dust (Exposure) versus a measure of lung capacity (PEFR or “peak expiratory flow rate”). How is PEFR related to Exposure? It’s hard to tell based just on the picture.

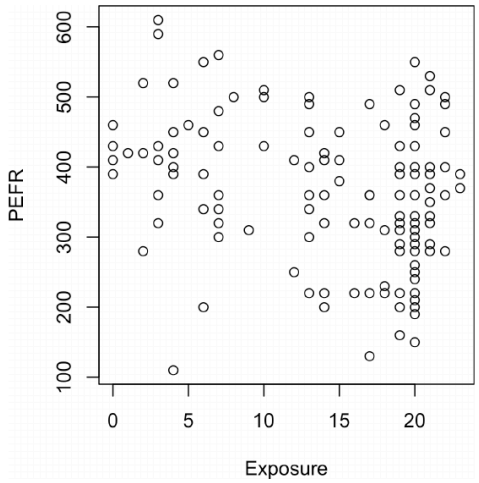


Figure 4-1. Cotton exposure versus lung capacity

Simple linear regression tries to find the “best” line to predict the response PEFR as a function of the predictor variable Exposure: PEFR = b1.Exposure + b0

Intercept: 424.583
Coefficient Exposure: -4.185

The intercept, or b0, is 424.583 and can be interpreted as the predicted PEFR for a worker with zero years exposure. The regression coefficient, or b1, can be interpreted as follows: for each additional year that a worker is exposed to cotton dust, the worker’s PEFR measurement is reduced by –4.185.

The regression line from this model is displayed in Figure 4-2:

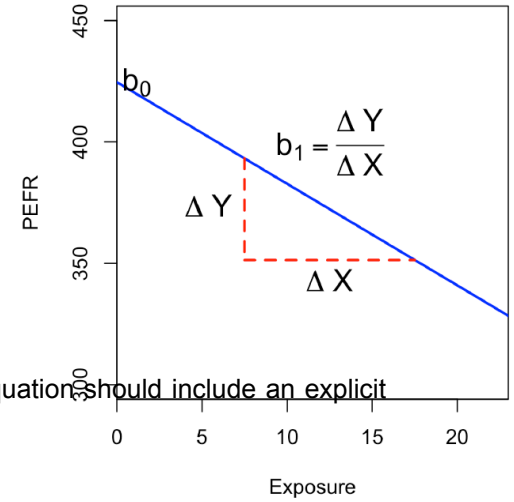


Figure 4-2. Slope and intercept for the regression fit to the lung data

I.2. Fitted values and Residuals

In general, the data doesn’t fall exactly on a line, so the regression equation should include an explicit error term ei: Yi = b1.Xi + b0 + ei

The fitted values, also referred to as the predicted values, are typically denoted by Y i (Y-hat). These are given by:

$$\hat{Y}_i = \hat{b}_0 + \hat{b}_1 X_i$$

The notation b 0 and b 1 indicates that the coefficients are estimated versus known.

We compute the residuals e i by subtracting the predicted values from the original Data:

$$\hat{e}_i = Y_i - \hat{Y}_i$$

With scikit-learn's LinearRegression model, we use the predict method on the training data to get the fitted values and subsequently the residuals. As we will see, this is a general pattern that all models in scikit-learn follow:

```
fitted = model.predict(lung[predictors])
residuals = lung[outcome] - fitted
```

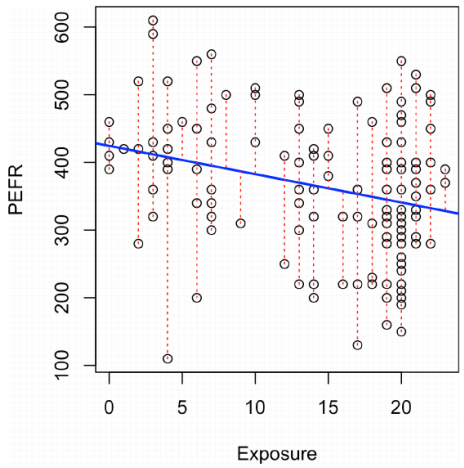


Figure 4-3. Residuals from a regression line (to accommodate all the data, the y-axis scale differs from Figure 4-2, hence the apparently different slope)

Figure 4-3 illustrates the residuals from the regression line fit to the lung data.

The residuals are the length of the vertical dashed lines from the data to the line.

I.3. Least Squares

How is the model fit to the data? When there is a clear relationship, you could imagine fitting the line by hand. In practice, the regression line is the estimate that minimizes the sum of squared residual values, also called the residual sum of squares or RSS:
The estimates b0 and b1 are the values that minimize RSS.

$$RSS = \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2$$

$$= \sum_{i=1}^{n} \left(Y_i - \hat{b}_0 - \hat{b}_1 X_i\right)^2$$

The method of minimizing the sum of the squared residuals is termed **least squares regression**, or ordinary least squares (OLS) regression. Least squares regression can be computed quickly and easily with any standard statistical software.

Least squares regression became popular partly because it is **computationally simple and fast** to calculate. Even today, with big data, speed still matters, and least squares remain efficient.

However, least squares regression is sensitive **to outliers**, just like the mean. A few extreme values can strongly influence the regression line. This sensitivity is usually a **serious issue only in small or medium-sized datasets**. In very large datasets, the effect of a few outliers is often diluted.

For cases where outliers matter, **robust regression methods** can be used instead.

*In machine learning, the term regression is used more loosely to mean **any model that predicts a numeric value**, regardless of whether the relationship is linear or nonlinear—unlike **classification**, which predicts a binary or categorical outcome.*

II. Prediction Versus Explanation (Profiling)

In regression, sometimes the main interest is in the **slope** bbb, which tells us how one variable affects another. For example, economists might study how **GDP growth affects consumer spending**, or public health officials might examine whether a **campaign increases safe sex practices**. Here, the goal isn't to predict individual cases but to **understand the overall relationship** between variables.

With **big data**, regression is often used to **predict individual outcomes**. For instance, marketers can use regression to estimate how **an ad campaign will change revenue**, or universities can predict a **student's GPA from their SAT score**. In these predictive settings, the focus shifts from the slope itself to the **fitted values YYY**—the predictions for new dat

A regression model can show that changes in **X** are associated with changes in **Y**, but it **doesn't prove causation** by itself. For example, a model might reveal a strong link between **web ad clicks** and **sales conversions**. However, knowing that clicks cause sales (and not the other way around) comes from our **understanding of the marketing process**, not just the regression numbers. Regression shows patterns; causation requires domain knowledge.

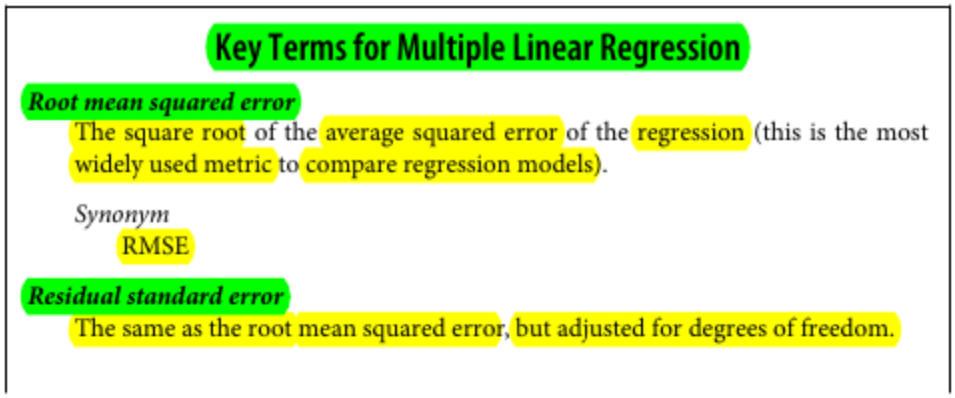
Key Ideas

- The regression equation models the relationship between a response variable Y and a predictor variable X as a line.
- A regression model yields fitted values and residuals—predictions of the response and the errors of the predictions.
- Regression models are typically fit by the method of least squares.
- Regression is used both for prediction and explanation.

III. Multiple Linear Regression

When there are multiple predictors, $$Y = b_0 + b_1X_1 + b_2X_2 + ... + b_pX_p + e$$

The relationship between each coefficient and its variable (feature) is linear.



Key Terms for Multiple Linear Regression

Root mean squared error

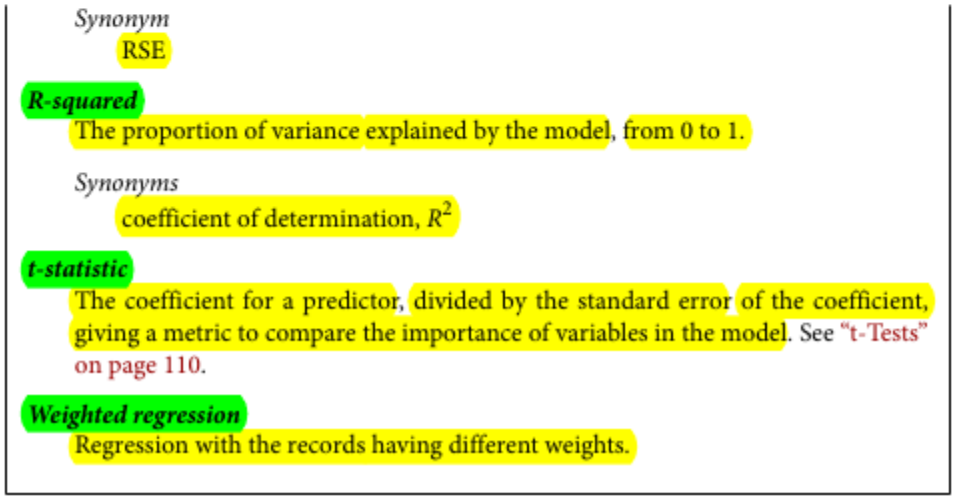
The square root of the average squared error of the regression (this is the most widely used metric to compare regression models).

Synonym

RMSE

Residual standard error

The same as the root mean squared error, but adjusted for degrees of freedom.



Synonym

RSE

R-squared

The proportion of variance explained by the model, from 0 to 1.

Synonyms

coefficient of determination, R^2

t-statistic

The coefficient for a predictor, divided by the standard error of the coefficient, giving a metric to compare the importance of variables in the model. See "t-Tests" on page 110.

Weighted regression

Regression with the records having different weights.

All of the other concepts in simple linear regression, such as fitting by least squares and the definition of fitted values and residuals, extend to the multiple linear regression setting. For example, the fitted values are given by:

$$\hat{Y}_i = \hat{b}_0 + \hat{b}_1 X_{1,i} + \hat{b}_2 X_{2,i} + \ldots + \hat{b}_p X_{p,i}$$

Example: King County Housing Data

An example of using multiple linear regression is in estimating the value of houses. County assessors must estimate the value of a house for the purposes of assessing taxes. Real estate professionals and

home buyers consult popular websites such as Zillow to ascertain a fair price. Here are a few rows of housing data from King County (Seattle), Washington, from the house data.frame:

	AdjSalePrice	SqFtTotLiving	SqFtLot	Bathrooms	Bedrooms	BldgGrade
1	300,805.00	2,400	9,373	3	6	7
2	1,076,162.00	3,764	20,156	3.75	4	10
3	761,805.00	2,060	26,036	1.75	4	8
4	442,065.00	3,200	8,618	3.75	5	7
5	297,065.00	1,720	8,620	1.75	4	7

…
.
.
.

The goal is to predict the sales price from the other variables. The lm function handles the multiple regression case simply by including more terms on the righthand side of the equation,

scikit-learn's LinearRegression can be used for multiple linear regression.

```
LinearRegression()
Intercept: -521871.368
Coefficients:
 SqFtTotLiving: 228.83060360240793
 SqFtLot: -0.06046682065307607
 Bathrooms: -19442.840398321066
 Bedrooms: -47769.95518521438
 BldgGrade: 106106.96307898081
```

those are 'Slopes'

The interpretation of the coefficients is as with simple linear regression: the predicted value Y changes by the coefficient bj for each unit change in Xj assuming all the other variables remain the same.

Interpretation:

- **Intercept (-521,871.368):** This is the predicted house price when all predictors are 0. It doesn't have a real-world meaning here, since a house with 0 square feet, 0 bathrooms, etc., doesn't exist.
- **SqFtTotLiving (228.831):** For each additional square foot of living space, the house price increases by roughly $229, holding all other factors constant.
- **SqFtLot (-0.060):** Each extra square foot of lot size slightly **reduces** the predicted price by about 6 cents. This may indicate that larger lots are often associated with cheaper properties in this dataset.
- **Bathrooms (-19,442.84):** Surprisingly, more bathrooms slightly reduce price, which could be due to multicollinearity with other features (like square footage or bedrooms).

- **Bedrooms (-47,769.96):** More bedrooms reduce predicted price, likely because larger houses already capture value in `SqFtTotLiving`—adding bedrooms without increasing total space may lower efficiency.
- **BldgGrade (106,106.96):** A higher building grade increases predicted price by $106k per unit, showing that quality significantly affects house value.

III.1 Assessing the Model

The most important performance metric from a data science perspective is root mean squared error, or RMSE. RMSE is the square root of the average squared error in th predicted yi values:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$$

This measures the overall accuracy of the model and is a basis for comparing it to other models (including models fit using machine learning techniques). Similar to RMSE is the residual standard error, or RSE. In this case we have p predictors, and the RSE is given by:

$$RSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{(n-p-1)}}$$

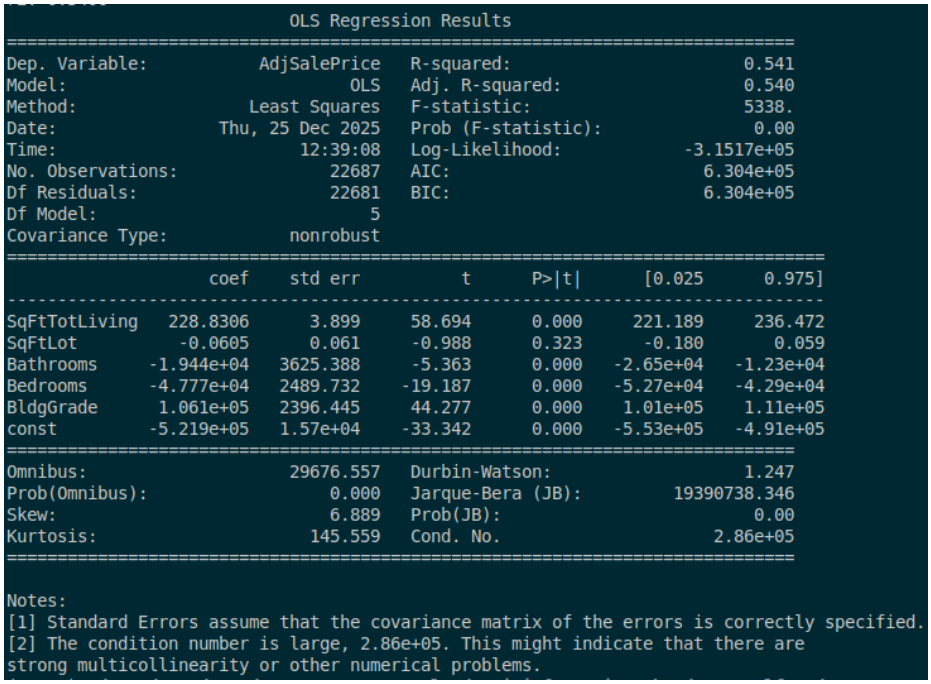
The only difference is that the denominator is the degrees of freedom, as opposed to number of records (see “Degrees of Freedom” on page 116). In practice, for linear regression, the difference between RMSE and RSE is very small, particularly for big data applications.

scikit-learn provides a number of metrics for regression and classification. Here, we use mean_squared_error to get RMSE and r2_score for the coefficient of determination:

```
RMSE: 261220
r2: 0.5406
```

Use statsmodels to get a more detailed analysis of the regression model in Python:

The pandas method assign, as used here, adds a constant column with value 1 to the predictors. This is required to model the intercept.



```
                            OLS Regression Results
==============================================================================
Dep. Variable:           AdjSalePrice   R-squared:                       0.541
Model:                          OLS   Adj. R-squared:                  0.540
Method:               Least Squares   F-statistic:                     5338.
Date:              Thu, 25 Dec 2025   Prob (F-statistic):               0.00
Time:                      12:39:08   Log-Likelihood:            -3.1517e+05
No. Observations:             22687   AIC:                         6.304e+05
Df Residuals:                 22681   BIC:                         6.304e+05
Df Model:                         5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
SqFtTotLiving  228.8306      3.899     58.694      0.000     221.189     236.472
SqFtLot         -0.0605      0.061     -0.988      0.323      -0.180       0.059
Bathrooms    -1.944e+04   3625.388     -5.363      0.000   -2.65e+04   -1.23e+04
Bedrooms     -4.777e+04   2489.732    -19.187      0.000   -5.27e+04   -4.29e+04
BldgGrade     1.061e+05   2396.445     44.277      0.000    1.01e+05    1.11e+05
const        -5.219e+05   1.57e+04    -33.342      0.000   -5.53e+05   -4.91e+05
==============================================================================
Omnibus:                    29676.557   Durbin-Watson:                   1.247
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         19390738.346
Skew:                           6.889   Prob(JB):                         0.00
Kurtosis:                     145.559   Cond. No.                     2.86e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.86e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

R-squared (R^2) is a measure of how well your regression model explains the variation in the outcome variable. It ranges from 0 to 1:

- **0** means the model explains none of the variation—basically, it's no better than just using the mean.
- **1** means the model explains all the variation perfectly.

For example, if you're predicting house prices, an R^2 of 0.8 means 80% of the differences in house prices are explained by the features in your model (like square footage, number of bathrooms, etc.), while 20% is due to factors your model doesn't capture, such as location or age of the house.

R^2 is especially useful when your goal is **explanation**, to see how well your predictors account for changes in the outcome. It's less important if your main goal is prediction or ranking results.

$$R^2 = 1 - \frac{\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}{\Sigma_{i=1}^{n}(y_i - \bar{y})^2}$$

Adjusted R^2 is a slightly refined version: it penalizes the model for adding extra predictors that don't really improve the fit. For example, if you predict house prices using square footage and bedrooms, adding "color of the front door" might increase R^2 slightly just by chance—but adjusted R^2 would correct for that, keeping the value from inflating unnecessarily.

In **large datasets**, adjusted R^2 is usually very close to R^2, because the penalty for extra predictors becomes relatively small.

When you run a regression, you get not only the coefficients (slopes) for each predictor, but also a **standard error (SE)** for each coefficient. This SE measures how much the estimated coefficient might vary if you repeated the experiment.

The **t-statistic** is calculated as: $$t_b = \frac{\hat{b}}{\text{SE}(\hat{b})}$$

A high t-statistic (and a low corresponding **p-value**) means the predictor is likely **truly related** to the outcome, not just due to random chance.

For example, if you predict house prices with square footage and bathrooms, and square footage has a very high t-statistic and tiny p-value, it's a strong predictor. On the other hand, if bathrooms have a low t-statistic and high p-value, its effect might be weak or just noise.

This helps in **model selection**: you can focus on significant predictors to keep the model simple and interpretable.

In addition to the t-statistic, *R* and other packages will often report a *p-value* (`Pr(>|t|)` in the *R* output) and *F-statistic*. Data scientists do not generally get too involved with the interpretation of these statistics, nor with the issue of statistical significance. Data scientists primarily focus on the t-statistic as a useful guide for whether to include a predictor in a model or not. High t-statistics (which go with p-values near 0) indicate a predictor should be retained in a model, while very low t-statistics indicate a predictor could be dropped. See "p-Value" on page 106 for more discussion.

IV. Cross validation

If you **study using a set of questions** and then **test yourself on the exact same questions**, you'll probably score very high—but that doesn't mean you truly understand the subject. That's exactly what **in-sample metrics** do.

In-sample metrics (R^2, F-statistic, p-values)

- They are calculated **on the same data used to fit the model**
- They tell you **how well the model explains the data it already saw**
- They can be **overly optimistic**, especially with many predictors

Example: You fit a house-price model on 10,000 houses and get **$R^2 = 0.90$**. That looks great—but it only means the model fits *those* 10,000 houses well.

Holdout (train–test) idea

Now imagine you:

- Use **80% of the data** to train the model
- Keep **20% aside** (the *holdout set*)
- Test the model on that unseen 20%

This answers the real question:

"How well will my model perform on new data?"

Example:

- Train $R^2 = 0.90$
- Test $R^2 = 0.62$

Now you see the **true predictive ability**.

Why data scientists prefer this

- Business and science care about **future performance**, not past fit
- Holdout testing reveals **overfitting**
- This idea leads naturally to **cross-validation**

One-sentence takeaway

> **Classic regression metrics explain how well a model fits known data; holdout testing shows how well it predicts new data.**

Cross validation idea

Why a single holdout can be misleading ?

With small or medium data:

- You train on 80%, test on 20%
- But results depend on **which 20% you picked**
- A different split might give noticeably different performance

So you ask:

> *"What if I had chosen a different holdout sample?"*

Cross-validation: many small exams instead of one

k-fold cross-validation solves this by repeating the test many times.

In **cross-validation**, you do the *same thing*, but **many times**:

- For each fold 1 :
    - Train the model on remaining k−1 folds
    - Compute **R^2 on the remaining fold**

- At the end:
    - **Average the R^2 values across all folds**

Imagine **k = 5** and 100 observations:

1. Split data into **5 equal parts (folds)** of 20
2. Hold out fold 1, train on the other 80 → test → record score
3. Hold out fold 2, train on the other 80 → test → record score
4. …
5. Hold out fold 5

6. Average or otherwise combine the model assessment metrics.

V. Model Selection and Stepwise Regression

In some problems, many variables could be used as predictors in a regression. For example, to predict house value, additional variables such as the basement size or year built could be used.

In Python, we need to convert the categorical and boolean variables into numbers.

Adding more variables, however, does not necessarily mean we have a better model. Statisticians use the principle of Occam's razor to guide the choice of a model: all things being equal, a simpler model should be used in preference to a more complicated model.

إن إضافة متغيرات إضافية يؤدي دائماً إلى تقليل **جذر متوسط مربع الخطأ (RMSE)** وزيادة **معامل التحديد (R^2)** بالنسبة لبيانات التدريب. ولذلك، فإن هذه المقاييس ليست مناسبة للمساعدة في توجيه اختيار النموذج. أحد التوجهات المتبعة لإدراج تعقيد النموذج في الاعتبار هو استخدام **معامل التحديد المعدل (Adjusted R^2)**

$$R^2_{adj} = 1 - (1 - R^2)\frac{n-1}{n-P-1}$$

Here, n is the number of records and P is the number of variables in the model.

AIC

In the 1970s, Hirotugu Akaike created **Akaike's Information Criterion (AIC)** to help choose regression models. The idea is simple: while adding more variables can improve the fit of a model, too many variables can overcomplicate it. AIC balances **fit versus complexity** using this formula:

$$\text{AIC} = 2P + n\log(\text{RSS}/n)$$

- **P** = number of predictors
- **n** = number of observations
- **RSS** = residual sum of squares

The lower the AIC, the better the model. Adding extra variables increases AIC by 2k2k2k for each additional variable, acting as a **penalty for unnecessary complexity**.
Other related metrics include:

- **AICc**: Corrected AIC for small sample sizes.
- **BIC (Bayesian Information Criterion)**: Like AIC but penalizes extra variables more strongly.
- **Mallows Cp**: Another variant similar to AIC, used for model selection.

Example in practice:

Suppose you are predicting house prices. You could start with a simple model using only `SqFtTotLiving`. Adding `Bedrooms`, `Bathrooms`, and `BldgGrade` might improve the fit, but AIC helps check whether the improvement is worth the added complexity. If the AIC decreases, the new model is better; if it rises, the extra variables may not be helpful.
For data scientists, these metrics are mainly **in-sample checks**—useful for comparing models on training data—but when you validate on a holdout set, the differences between AIC, BIC, and Cp are less critical. This way, you balance **accuracy** and **simplicity**, avoiding overfitting while keeping the model interpretable.

When trying to find the **best regression model**, two common goals are to **minimize AIC** or **maximize adjusted R^2**. One brute-force way is **all subset regression**, where you check every possible combination of predictors. But with many variables, this quickly becomes **computationally impractical**.

A more practical approach is **stepwise regression**, which simplifies the search:

1. **Backward elimination** – start with a full model including all predictors, then **remove variables one by one** that contribute little to the model. For example, if `SqFtLot` doesn't significantly improve predicting house prices, it can be dropped.
2. **Forward selection** – start with a simple model (maybe just the intercept) and **add predictors one by one**, keeping those that meaningfully improve the model. For instance, first add `SqFtTotLiving`, then `Bathrooms` if it helps.
3. **Stepwise (both directions)** – a hybrid where predictors can be **added or removed** at each step to improve the model based on AIC or adjusted R^2.

Example: Suppose you're modeling house prices. You could start with `SqFtTotLiving` (forward selection), add `BldgGrade`, then notice `Bedrooms` doesn't help and remove it (stepwise). The final model balances **fit and simplicity**, achieving a lower AIC or higher adjusted R^2 without unnecessary predictors

scikit-learn has no implementation for stepwise regression. We implemented functions stepwise_selection, forward_selection, and backward_elimination in our dmba package.

Output:

```
Adding SqFtTotLiving, AIC: 568628.44
Adding BldgGrade, AIC: 566408.82
Adding YrBuilt, AIC: 563845.38
Adding Bedrooms, AIC: 563399.24
Adding Bathrooms, AIC: 563217.29
Adding PropertyType_Townhouse, AIC: 563140.74
Adding SqFtFinBasement, AIC: 563140.16
Adding PropertyType_Single Family, AIC: 563140.06
Adding SqFtLot, AIC: 563140.16
Adding YrRenovated, AIC: 563141.73
Adding NewConstruction, AIC: 563143.55
Adding NbrLivingUnits, AIC: 563145.45

Intercept: 6181909.910
Coefficients:
 SqFtTotLiving: 198.636
 BldgGrade: 137266.532
 YrBuilt: -3574.221
 Bedrooms: -51865.345
 Bathrooms: 42860.330
 PropertyType_Townhouse: 92860.061
 SqFtFinBasement: 7.061
 PropertyType_Single Family: 29972.344
 SqFtLot: 0.077
 YrRenovated: -2.531
 NewConstruction: -2489.112
 NbrLivingUnits: 5723.844
```

1. Define a function that returns a fitted model for a given set of variables.

2. Define a function that returns a score for a given model and set of variables. In this case, we use the AIC_score implemented in the dmba package.

The function chose a model in which several variables were dropped from house_full: SqFtLot, NbrLivingUnits, YrRenovated, and NewConstruction.

- **forward selection** starts with no predictors at all. You add one predictor at a time—the one that increases the model's R^2 the most. You keep adding predictors as long as each new one makes a statistically meaningful improvement. For example, if you want to predict house prices, you might first add `SqFtTotLiving` because it explains the most variation. Next, `BldgGrade` might be added because it further improves R^2. You stop when adding another variable doesn't meaningfully improve the model.

- **Backward elimination** works the opposite way. You start with all predictors in the model. Then, one by one, you remove the least significant variable—the one whose p-value is highest or contributes least to R^2—until all remaining predictors are statistically significant. Using the same house example, if `YrRenovated` doesn't improve the model, you drop it, and continue until every variable left matters.

In short: forward = build up, backward = trim down, both aiming for a model that explains well without unnecessary predictors.

Penalized regression is a clever way to prevent overfitting, similar to the logic behind AIC. Instead of trying lots of different combinations of predictors like in forward or backward selection, it adds a "penalty" directly into the model-fitting process for having too many or too large coefficients.

- **Ridge regression** shrinks all coefficients toward zero, but usually none reach exactly zero. It's useful when many predictors are correlated, so it keeps them in the model but reduces their impact.

- **Lasso regression** also shrinks coefficients, but some can be reduced exactly to zero, effectively performing variable selection automatically.

For example, if you're predicting house prices and include both `SqFtTotLiving` and `Bedrooms`, which are highly correlated, ridge will reduce their coefficients proportionally to avoid overfitting. Lasso might even drop `Bedrooms` entirely if it doesn't add enough predictive value, leaving a simpler model.

In short: penalized regression controls complexity by shrinking coefficients rather than manually adding or removing predictors.

Stepwise regression and all-subset regression help you pick the "best" model using the same data you fit the model on. This is called **in-sample** evaluation. The problem is that the model might be tuned too closely to this specific data, capturing random noise rather than true patterns a phenomenon called **overfitting**.

To check if the model will really perform well on new, unseen data, you can use **cross-validation**. Here, you repeatedly split the data into training and testing sets, fit the model on the training part, and evaluate it on the testing part. This gives a better estimate of how the model will generalize.

For **linear regression**, overfitting is usually less of a concern because the model is simple and linear. But for complex models—like trees, ensembles, or iterative procedures that adapt to local patterns—cross-validation becomes crucial to avoid overfitting and ensure the model works well on new data.

Example: If you use stepwise regression to predict house prices with many variables, the model may look perfect on your current dataset. But using 5-fold cross-validation, you might see that some variables don't actually improve predictions on unseen data, helping you build a more robust model.

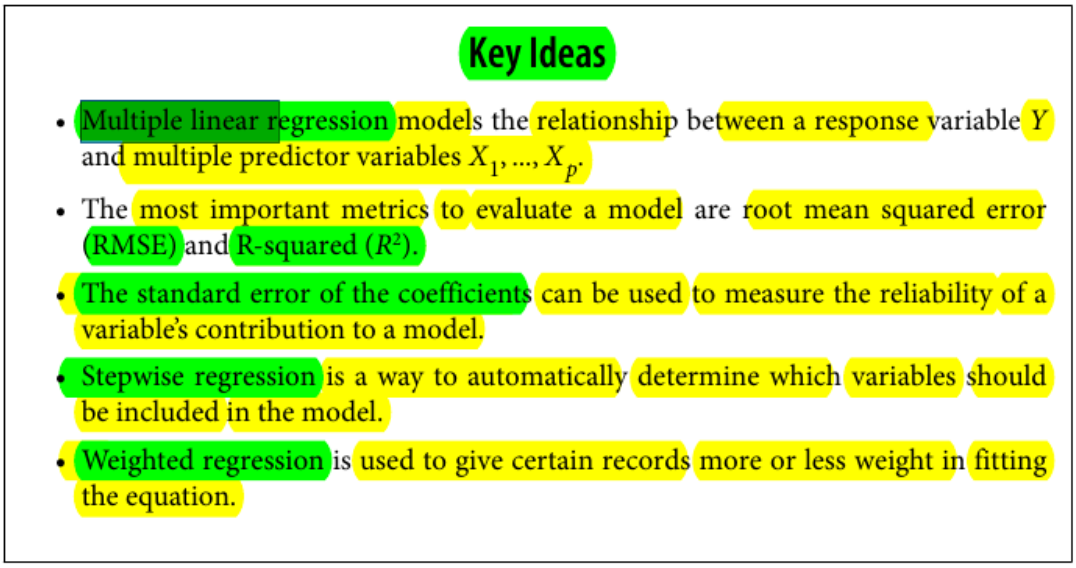
V. Weighted Regression

Weighted regression is a way to give some observations more influence than others when fitting a regression model. It's often used in complex survey analysis, but data scientists might use it in two main situations:

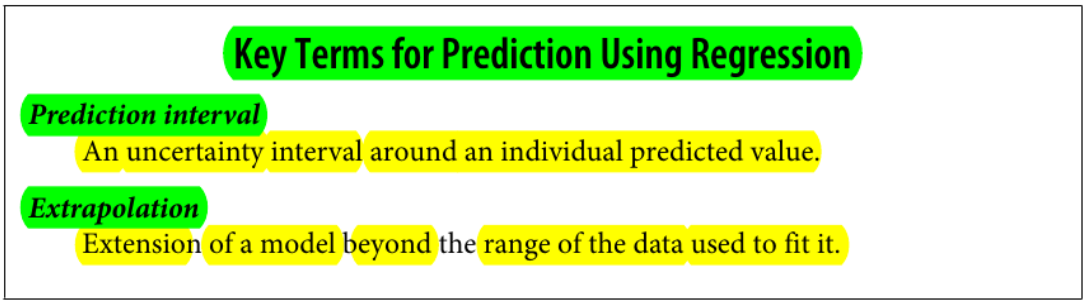
1. **Inverse-variance weighting:** Some observations are measured more precisely than others. Observations with high variance (less precise) get lower weights, so they influence the model less.

2. **Aggregated data:** Sometimes each row in your dataset represents multiple original cases. A weight variable tells the model how many cases each row stands for, so the regression accounts for that.

Example: In housing data, older home sales may be less reliable than recent ones. By giving older sales lower weights, the regression focuses more on recent, more relevant transactions, improving the accuracy of predictions like current home prices.



VI. Prediction Using Regression



VI.1. The Dangers of Extrapolation

This passage is emphasizing a key limitation of regression models: **they are only reliable within the range of the data used to train them**.

For example, imagine you fit a regression model to predict house prices (`AdjSalePrice`) based on features like living area, lot size, number of bathrooms, bedrooms, and building grade. All the data in your dataset are for properties **with buildings**—there are no vacant lots.

Now, if you try to use this model to predict the price of a **5,000-square-foot empty lot**, the model has no information about vacant land. All the building-related features (bathrooms, bedrooms, grade, etc.) would be zero. The regression equation might then give a nonsensical result, like:

$$\text{Predicted Price} = -521{,}900 + 5{,}000 \times (-0.0605) \approx -522{,}202$$

This negative value is absurd because the model was never trained on properties like this—it has no knowledge of how lot size alone affects price.

VI.2 Confidence and Prediction Intervals

In regression, the coefficients (slopes) tell us how predictors affect the outcome. But these estimates are uncertain because they come from a sample of data, not the entire population. While t-statistics and p-values formalize this uncertainty, **confidence intervals are often more intuitive**: they give a range where the true coefficient likely lies.

Bootstrap approach for regression coefficients:

1. Imagine each row of your dataset (predictors + outcome) is a “ticket” in a box.
2. Randomly pick a ticket, record it, and put it back in the box.
3. Repeat this **n times** (where n = number of rows) to create a **bootstrap sample**.
4. Fit a regression on this bootstrap sample and record the estimated coefficients.
5. Repeat the whole process many times (e.g., 1,000 bootstrap samples).
6. For each coefficient, you now have 1,000 estimates. The **5th and 95th percentiles** of these estimates give a **90% confidence interval**.

Example: Suppose you have a regression predicting house price from square footage. After bootstrapping 1,000 times, the coefficient for square footage might have a 90% confidence interval of

[220, 240]. This means you can be 90% confident that the true increase in price per square foot lies between $220 and $240.

The **difference between confidence intervals for regression coefficients and confidence intervals for predictions**.

In regression, you can estimate **confidence intervals for the coefficients** (slopes) using either:

- **Bootstrap methods** (resampling your data), or
- **Formula-based intervals** (standard output in R or Python).

Conceptually, both methods give the same meaning: a range where the true coefficient likely lies. For example, a 90% CI for the square footage coefficient might be **[220, 240]**, meaning the true effect of one additional square foot is likely between $220 and $240.

But for data scientists, intervals around predicted values (YiY_iYi) are often more useful, because they show the uncertainty in predictions for new data points.

The uncertainty in predicted YiY_iYi comes from **two sources**:

1. **Uncertainty in the coefficients and model selection** – even with a good model, we don't know the exact true slopes.
2. **Random error in individual observations** – each actual outcome can vary around the predicted value due to noise.

Example: Suppose your regression model predicts a house price Y^i = 300,000, a 90% prediction interval might be **$280,000 to $320,000**. This interval is wider than the coefficient CI because it includes both the model uncertainty and the natural variability of house prices.

Even if our regression model is perfect, **individual houses with the same features can have different prices**. For instance, three houses with 8 rooms, a 6,500 sq ft lot, 3 bathrooms, and a basement might sell for $310,000, $325,000, and $300,000. This variability is captured by the **residuals**—the differences between actual and predicted prices.
To account for both **model uncertainty** and **individual variability**, we can use a **bootstrap procedure**:

1. Randomly sample the data with replacement (bootstrap sample).
2. Fit the regression model to this sample and predict the outcome for a house.
3. Add a residual randomly picked from the original model fit to the predicted value. This simulates individual variability.
4. Repeat steps 1–3 many times (e.g., 1,000 times) to generate a distribution of predicted prices.
5. Take the 2.5th and 97.5th percentiles of this distribution to get a **95% prediction interval**.

Example:

- Predicted price from model: $315,000
- After bootstrapping and adding residuals 1,000 times, the 2.5th percentile = $300,000, the 97.5th percentile = $330,000.

- So, the **prediction interval** is $300,000–$330,000, reflecting both model uncertainty and natural variation in house prices.

This approach gives a more realistic range for a single prediction than just using the regression equation alone.

Key Ideas

- Extrapolation beyond the range of the data can lead to error.
- Confidence intervals quantify uncertainty around regression coefficients.
- Prediction intervals quantify uncertainty in individual predictions.
- Most software, *R* included, will produce prediction and confidence intervals in default or specified output, using formulas.
- The bootstrap can also be used to produce prediction and confidence intervals; the interpretation and idea are the same.

Prediction Interval or Confidence Interval?

A prediction interval pertains to uncertainty around a single value, while a confidence interval pertains to a mean or other statistic calculated from multiple values. Thus, a prediction interval will typically be much wider than a confidence interval for the same value. We model this individual value error in the bootstrap model by selecting an individual residual to tack on to the predicted value. Which should you use? That depends on the context and the purpose of the analysis, but, in general, data scientists are interested in specific individual predictions, so a prediction interval would be more appropriate. Using a confidence interval when you should be using a prediction interval will greatly underestimate the uncertainty in a given predicted value.

VII. Factor Variables in Regression

Factor variables, also termed categorical variables, take on a limited number of discrete values. For example, a loan purpose can be “debt consolidation,” “wedding,” “car,” and so on. The binary (yes/no) variable, also called an indicator variable, is a special case of a factor variable. Regression requires numerical inputs, so factor variables need to be recoded to use in the model. The most common approach is to convert a variable into a set of binary dummy variables.

Key Terms for Factor Variables

Dummy variables

Binary 0–1 variables derived by recoding factor data for use in regression and other models.

Reference coding

The most common type of coding used by statisticians, in which one level of a factor is used as a reference and other factors are compared to that level.

Synonym

treatment coding

One hot encoder

A common type of coding used in the machine learning community in which all factor levels are retained. While useful for certain machine learning algorithms, this approach is not appropriate for multiple linear regression.

Deviation coding

A type of coding that compares each level against the overall mean as opposed to the reference level.

Synonym

sum contrasts

```
1         Multiplex
2     Single Family
3     Single Family
4     Single Family
5     Single Family
Name: PropertyType, dtype: obj
```

There are three possible values: Multiplex, Single Family, and Townhouse. To use this factor variable, we need to convert it to a set of binary variables. We do this by creating a binary variable for each possible value of the factor variable.

The function model.matrix converts a data frame into a matrix suitable to a linear model. The factor variable PropertyType, which has three distinct levels, is represented as a matrix with three columns. In the machine learning community, this representation is referred to as one hot encoding (see “One Hot Encoder” on page 242).

In Python, we can convert categorical variables to dummies using the pandas method get_dummies.

```
One-hot encoded PropertyType (all columns):
   Multiplex  Single Family  Townhouse
1       True          False      False
2      False           True      False
3      False           True      False
4      False           True      False
5      False           True      False

One-hot encoded PropertyType (drop first column
   Single Family  Townhouse
1          False      False
2           True      False
3           True      False
4           True      False
5           True      False
```

In certain machine learning algorithms, such as nearest neighbors and tree models, one hot encoding is the standard way to represent factor variables (for example, see “Tree Models” on page 249).

In regression, when you have a categorical variable (a “factor”) with **P levels**—for example, `PropertyType` with 3 types: `Single`, `Condo`, `Townhouse`—you can’t just add a column for each type if your model already has an **intercept**.

Why? Because the intercept already captures the baseline level. If you include all **P columns**, the last column is fully determined by the others—if the first two are 0, the third must be 1. This creates **multicollinearity**, meaning the model can’t uniquely estimate coefficients.

So instead, we include only **P – 1 columns**. For our example:

- `Single` → baseline (intercept)
- `Condo` → 1 if condo, 0 otherwise
- `Townhouse` → 1 if townhouse, 0 otherwise

Now the model can estimate the effect of `Condo` and `Townhouse` relative to `Single`. This avoids redundancy and keeps the regression stable.

In Python/pandas, `pd.get_dummies` with `drop_first=True`, it automatically **drops the first level** so that the remaining columns represent the difference from the reference.

```
Intercept: -446841.366
Coefficients:
 SqFtTotLiving: 223.374
 SqFtLot: -0.070
 Bathrooms: -15979.013
 Bedrooms: -50889.732
 BldgGrade: 109416.305
 PropertyType_Single Family: -84678.216
 PropertyType_Townhouse: -115121.979
```

In the regression output, `PropertyType` is a categorical variable with three levels: **Multiplex**, **Single Family**, and **Townhouse**. Because we include an intercept in the model, we only need **P – 1 columns** (here, 2) to represent the factor:

- `PropertyTypeSingle Family`
- `PropertyTypeTownhouse`

The **reference level**, Multiplex, is implicitly defined. This means:

- If both `PropertyTypeSingle Family` and `PropertyTypeTownhouse` are 0, the property is a Multiplex.
- The coefficients for the other two types are interpreted **relative to Multiplex**.

For example:

- If `PropertyTypeSingle Family` has a coefficient of **–85,000**, a Single Family home is estimated to sell for $85,000 **less** than a Multiplex, all else equal.

- If `PropertyTypeTownhouse` has a coefficient of **–150,000**, a Townhouse sells for $150,000 **less** than a Multiplex, all else equal.

Different Factor Codings

There are several different ways to encode factor variables, known as *contrast coding* systems. For example, *deviation coding*, also known as *sum contrasts*, compares each level against the overall mean. Another contrast is *polynomial coding*, which is appropriate for ordered factors; see the section "Ordered Factor Variables" on page 169. With the exception of ordered factors, data scientists will generally not encounter any type of coding besides reference coding or one hot encoder.

VII.1. Factor Variables with many Levels

Some categorical variables can have **many levels**, like zip codes in the US—there are over 43,000. If you create a dummy variable for each level, the model will have tens of thousands of columns, which is **impractical** and can lead to overfitting.

Before including such a variable in regression, you should:

1. **Explore the data**: Check if zip codes actually have a relationship with the outcome (e.g., house prices). Maybe some areas have systematically higher prices.
2. **Decide on retention or consolidation**:
    - If some categories have similar effects, you can **group them**. For example, combine zip codes into neighborhoods or regions.
    - If the variable contains little useful information, it may be better to **exclude it**.

Example: Instead of 43,000 zip codes, you could use **10 regions**, capturing most of the variation without overloading the model. This approach keeps the model **manageable and interpretable** while preserving important information.

Zip codes can be **very informative** in predicting house prices because they reflect location, but including every zip code as a separate variable can be problematic:

- For example, if there are 79 zip codes, you'd need 79 coefficients—far more than the 5 degrees of freedom in the simple model `house_lm`.
- Some zip codes may have **only one sale**, giving unreliable estimates.

To handle this, you can **consolidate zip codes**:

1. **By digits**: Use the first two or three digits to create broader geographic regions (submetropolitan areas). Example: 980xx or 981xx. In some counties, this may not help if most sales fall into just a few codes.

2. **By outcome variable**: Group zip codes according to sale price, so regions with similar prices are combined.
3. **By residuals**: Fit an initial model, calculate residuals, and cluster zip codes with similar residual patterns. This captures location effects not explained by other predictors.

An alternative approach is to group the zip codes according to another variable, such as sale price. Even better is to form zip code groups using the residuals from an initial model. Consolidates the 80 zip codes into five groups based on the median of the residual from the house_lm regression.

	ZipCode	ZipGroup
1	98002	2
2	98166	2
3	98166	2
4	98168	2
5	98168	2
...	...	...
27057	98126	3
27058	98040	4
27061	98055	0
27062	98166	2
27063	98133	4

The concept of using the residuals to help guide the regression fitting is a fundamental step in the modeling process; see “Regression Diagnostics” on page 176.

Ordered Factor Variables

Some categorical variables have a natural order, called **ordered factor variables**. For example, a loan grade might be A, B, C… where each higher letter indicates more risk.

Instead of creating separate dummies for each level, these variables can often be **converted to numbers** that preserve the order. This is useful in regression because the numeric values carry meaningful ranking information.

For example, in housing data, **BldgGrade** reflects the quality of a building. Lower numbers indicate lower-quality homes, higher numbers indicate higher-quality homes. By treating BldgGrade as a numeric variable in the regression model, we can capture the effect of **increasing quality on price** without creating multiple dummy variables.

Example:

- Grade 7 → lower-quality home
- Grade 10 → higher-quality home
- Regression coefficient for BldgGrade = 106,000 → means each one-level increase in grade increases predicted house price by $106,000.

This approach keeps the model simple and respects the natural order of the variable.

Key Ideas

- Factor variables need to be converted into numeric variables for use in a regression.
- The most common method to encode a factor variable with P distinct values is to represent them using P – 1 dummy variables.
- A factor variable with many levels, even in very big data sets, may need to be consolidated into a variable with fewer levels.
- Some factors have levels that are ordered and can be represented as a single numeric variable.

VIII. Interpreting the Regression Equation

In data science, the most important use of regression is to predict some dependent (outcome) variable. In some cases, however, gaining insight from the equation itself to understand the nature of the relationship between the predictors and the outcome can be of value. This section provides guidance on examining the regression equation and interpreting it.

Key Terms for Interpreting the Regression Equation

Correlated variables
When the predictor variables are highly correlated, it is difficult to interpret the individual coefficients.

Multicollinearity
When the predictor variables have perfect, or near-perfect, correlation, the regression can be unstable or impossible to compute.

Synonym
collinearity

Confounding variables
An important predictor that, when omitted, leads to spurious relationships in a regression equation.

Main effects
The relationship between a predictor and the outcome variable, independent of other variables.

Interactions
An interdependent relationship between two or more predictors and the response.

VIII.1. Correlated Predictors

In multiple regression, the predictor variables are often correlated with each other.

```
      (Intercept)     SqFtTotLiving          Bathrooms
     6.178645e+06      1.992776e+02       4.239616e+04
         Bedrooms        BldgGrade PropertyTypeSingle Family
    -5.194738e+04      1.371596e+05       2.291206e+04
PropertyTypeTownhouse SqFtFinBasement           YrBuilt
     8.447916e+04      7.046975e+00      -3.565425e+03
```

The coefficient for Bedrooms is negative! This implies that adding a bedroom to a house will reduce its value. How can this be? This is because the predictor variables are correlated: larger houses tend to have more bedrooms, and it is the size that drives house value, not the number of bedrooms. Consider two homes of the exact same size: it is reasonable to expect that a home with more but smaller bedrooms would be considered less desirable.

Having correlated predictors can make it difficult to interpret the sign and value of regression coefficients (and can inflate the standard error of the estimates). The variables for bedrooms, house size, and number of bathrooms are all correlated.

Removing : SqFtTotLiving, SqFtFinBasement, and Bathrooms variables

```
Intercept: 4913973.344
Coefficients:
 Bedrooms: 27150.537
 BldgGrade: 248997.794
 YrBuilt: -3211.745
 PropertyType_Single Family: -19898.495
 PropertyType_Townhouse: -47355.437
```

add or remove variables from a model. Now the coefficient for bedrooms is positive—in line with what we would expect (though it is really acting as a proxy for house size, now that those variables have been removed).

In regression, correlated variables aren't the only thing to worry about. For example, in the `house_lm` model, there's no variable capturing the **location** of the home. As a result, the model is mixing together very different neighborhoods or regions. "Confounding Variables" on page 172 for further discussion.

VIII.2. Multicollinearity

An extreme case of correlated variables produces multicollinearity—a condition in which there is redundance among the predictor variables. Perfect multicollinearity occurs when one predictor variable can be expressed as a linear combination of others. Multicollinearity occurs when:

- A variable is included multiple times by error.
- P dummies, instead of P – 1 dummies, are created from a factor variable (see "Factor Variables in Regression" on page 163).
- Two variables are nearly perfectly correlated with one another.

Multicollinearity in regression must be addressed—variables should be removed until the multicollinearity is gone. A regression does not have a well-defined solution in the presence of perfect multicollinearity.

Many software packages, including R and Python, automatically handle certain types of multicollinearity. For example, if SqFtTotLiving is included twice in the regression of the house data, the results are the same as for the house_lm model. In the case of nonperfect multicollinearity, the software may obtain a solution, but the results may be unstable.

Multicollinearity is not such a problem for nonlinear regression methods like trees, clustering, and nearest-neighbors, and in such methods it may be advisable to retain P dummies (instead of $P - 1$). That said, even in those methods, nonredundancy in predictor variables is still a virtue.

VIII.3. Confounding Variables

With correlated variables, the problem is one of commission: including different variables that have a similar predictive relationship with the response. With confounding variables, the problem is one of omission: an important variable is not included in the regression equation. Naive interpretation of the equation coefficients can lead to invalid conclusions.

```
Intercept: -446841.366
Coefficients:
 SqFtTotLiving: 223.374
 SqFtLot: -0.070
 Bathrooms: -15979.013
 Bedrooms: -50889.732
 BldgGrade: 109416.305
 PropertyType_Single Family: -84678.216
 PropertyType_Townhouse: -115121.979
```

Take, for example, the King County regression equation house_lm from “Example: King County Housing Data” on page 151. The regression coefficients of SqFtLot, Bathrooms, and Bedrooms are all negative. The original regression model does not contain a variable to represent location—a very important predictor of house price.

To model location, include a variable ZipGroup that categorizes the zip code into one of five groups, from least expensive (1) to most expensive (5).

```
Coefficients:
 SqFtTotLiving: 220.2737807399172
 SqFtLot: 0.024318768830793758
 Bathrooms: -9304.16350448929
 Bedrooms: -49378.13407498488
 BldgGrade: 112859.08727579907
 ZipCode: 610.9897275780866
 PropertyType_Single Family: -66412.7228043166
 PropertyType_Townhouse: -112992.36865947503
```

ZipGroup is clearly an important variable: a home in the most expensive zip code group is estimated to have a higher sales price by almost $340,000. The coefficients of SqFtLot and Bathrooms are now positive, and adding a bathroom increases the sale price by $5,928.

The coefficient for Bedrooms is still negative. While this is unintuitive, this is a well- known phenomenon in real estate. For homes of the same livable area and number of bathrooms, having more and therefore smaller bedrooms is associated with less valuable homes.

XI.Interactions and Main Effects

Statisticians like to distinguish between main effects, or independent variables, and the interactions between the main effects. Main effects are what are often referred to as the predictor variables in the regression equation. An implicit assumption when only main effects are used in a model is that the relationship between a predictor variable and the response is independent of the other predictor variables. This is often not the case.

For example, the model fit to the King County Housing Data in “Confounding Variables” on page 172 includes several variables as main effects, including ZipCode. Location in real estate is everything, and it is natural to presume that the relationship between, say, house size and the sale price depends on location. A big house built in a low-rent district is not going to retain the same value as a big house built in an expensive area. You include interactions between variables in R using the * operator. For the King County data, the following fits an interaction between SqFtTotLiving and ZipGroup.

In Python, we need to use the statsmodels package to train linear regression models with interactions. This package was designed similar to R and allows defining models using a formula interface.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           AdjSalePrice   R-squared:                       0.549
Model:                            OLS   Adj. R-squared:                  0.549
Method:                 Least Squares   F-statistic:                     3067.
Date:                Sat, 27 Dec 2025   Prob (F-statistic):               0.00
Time:                        12:39:46   Log-Likelihood:            -3.1496e+05
No. Observations:               22687   AIC:                         6.299e+05
Df Residuals:                   22677   BIC:                         6.300e+05
Df Model:                           9
Covariance Type:            nonrobust
================================================================================================
                                   coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------------------------
Intercept                    -1.168e+08   8.01e+06    -14.576      0.000   -1.33e+08   -1.01e+08
PropertyType[T.Single Family] -6.772e+04   1.65e+04     -4.067      0.000   -9.97e+04   -3.49e+04
PropertyType[T.Townhouse]    -1.169e+05   1.8e+04      -6.488      0.000   -1.52e+05   -8.16e+04
SqFtTotLiving                 2.842e+04   3640.294      7.806      0.000    2.13e+04    3.56e+04
ZipCode                       1485.9451     81.704     14.515      0.000    1025.799    1346.091
SqFtTotLiving:ZipCode           -0.2875      0.037     -7.746      0.000      -0.360      -0.215
SqFtLot                          0.0938      0.661      0.082      0.934      -0.115       0.125
Bathrooms                    -6.8e+04    3810.724     -1.784      0.073   -1.43e+04     642.483
Bedrooms                     -4.766e+04   2528.243    -18.853      0.000   -5.26e+04   -4.27e+04
BldgGrade                     1.118e+05   2448.810     45.648      0.000    1.07e+05    1.17e+05
==============================================================================
Omnibus:                    29797.950   Durbin-Watson:                   1.263
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         19873077.938
Skew:                           6.938   Prob(JB):                         0.00
Kurtosis:                     147.320   Cond. No.                     1.04e+12
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.04e+12. This might indicate that there are
strong multicollinearity or other numerical problems.
```

The statsmodels package takes care of categorical variables(e.g., ZipGroup[T.1], PropertyType[T.Single Family]) and interaction terms (e.g., SqFtTotLiv ing:ZipGroup[T.1]).

Location and house size appear to have a strong interaction. For a home in the lowest ZipGroup, the slope is the same as the slope for the main effect SqFtTotLiving, which is $118 per square foot.

That means:

> **For the reference ZipGroup**, each additional square foot increases price by **≈ $28,420 per 1,000 sq ft**, or **≈ $118 per sq ft** (after scaling / unit conversion used in the book).

📌 This is the **baseline effect of house size**.

For a home in the highest ZipGroup, the slope is the sum of the main effect plus SqFtTotLiving:ZipGroup5, or $115 + $227 = $342 per square foot. In other words, adding a square foot in the most expensive zip code group boosts the predicted sale price by a factor of almost three, compared to the average boost from adding a square foot.

Look at these two rows:

```
SqFtTotLiving                 coef = 2.842e+04
SqFtTotLiving:ZipCode         coef = -0.2875
```

Conceptually, the slope for a given ZipGroup is:

```
Slope = SqFtTotLiving
      + (SqFtTotLiving × ZipGroup interaction)
```

In the **highest ZipGroup**, the interaction term adds extra value.
So the book's statement:

$115 + $227 = $342 per square foot

comes from:

- **$115** → main SqFtTotLiving effect
- **$227** → interaction effect for the most expensive ZipGroup

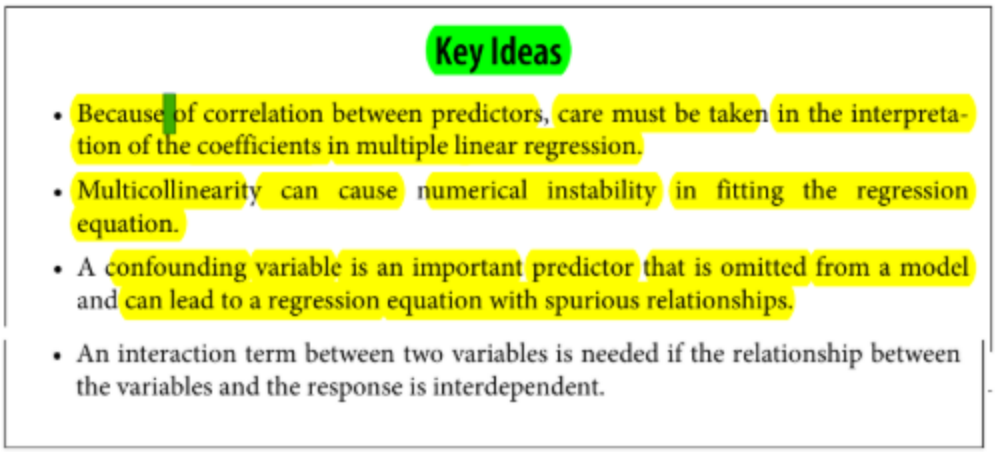
👉 Meaning:

In expensive areas, extra space is much more valuable

Model Selection with Interaction Terms

In problems involving many variables, it can be challenging to decide which interaction terms should be included in the model. Several different approaches are commonly taken:

- In some problems, prior knowledge and intuition can guide the choice of which interaction terms to include in the model.
- Stepwise selection (see "Model Selection and Stepwise Regression" on page 156) can be used to sift through the various models.
- Penalized regression can automatically fit to a large set of possible interaction terms.
- Perhaps the most common approach is to use *tree models*, as well as their descendants, *random forest* and *gradient boosted trees*. This class of models automatically searches for optimal interaction terms; see "Tree Models" on page 249.



2

X. Regression Diagnostics

In explanatory modeling (i.e., in a research context), various steps, in addition to the metrics mentioned previously (see “Assessing the Model” on page 153), are taken to assess how well the model fits the data; most are based on analysis of the residuals. These steps do not directly address predictive accuracy, but they can provide useful 2insight in a predictive setting.

Key Terms for Regression Diagnostics

Standardized residuals
Residuals divided by the standard error of the residuals.

Outliers
Records (or outcome values) that are distant from the rest of the data (or the predicted outcome).

Influential value
A value or record whose presence or absence makes a big difference in the regression equation.

Leverage
The degree of influence that a single record has on a regression equation.

Synonym
hat-value

Non-normal residuals
Non-normally distributed residuals can invalidate some technical requirements of regression but are usually not a concern in data science.

Heteroskedasticity
When some ranges of the outcome experience residuals with higher variance (may indicate a predictor missing from the equation).

Partial residual plots
A diagnostic plot to illuminate the relationship between the outcome variable and a single predictor.

Synonym
added variables plot

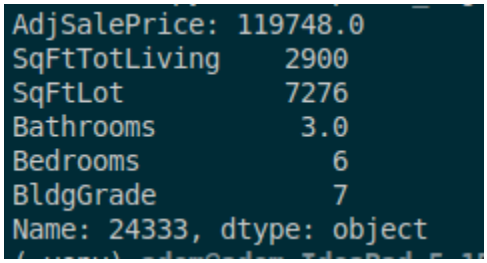
X.1. Outliers

Generally speaking, an extreme value, also called an *outlier*, is one that is distant from most of the other observations. Just as outliers must be handled when estimating **location** and **variability**, they can also cause problems in **regression models**.

In regression, an outlier is a record whose actual *y* value is far from the predicted value. One common way to detect such points is by using the **standardized residual**, which is the **residual divided by the standard error of the residuals**.

There is no strict statistical theory that cleanly separates outliers from non-outliers. Instead, analysts rely on **rules of thumb**. For example, in a **boxplot**, outliers are points that lie **more than 1.5 times the interquartile range** beyond the box boundaries.

In regression analysis, standardized residuals are the primary tool for identifying outliers. They can be interpreted as **the number of standard errors an observation lies away from the regression line**.



That **$119,748** is the **actual (observed) sale price** of the house, not the predicted value from the regression.

We identified it as an **outlier** because the regression model predicted a much higher value for a house of that size in that ZipCode, so the difference between the predicted value and the actual price (the residual) was very large.

The record in question is the **house in ZipCode 98105** that we identified as the **most extreme outlier** using the standardized residuals. Specifically, it is the house with:

- **AdjSalePrice:** $119,748
- **Predictor values** such as `SqFtTotLiving`, `SqFtLot`, `Bathrooms`, `Bedrooms`, and `BldgGrade` as shown in your `outlier[predictors]` output.

In this example, the record is an **outlier** because the house's sale price, $119,748, is far below what is typical for its size in that zip code. Investigating the sale deed reveals that it involved only a **partial interest** in the property, which explains the unusually low price. This shows that not all extreme values reflect normal sales—they can be **anomalous transactions**.

Outliers can also arise from **data entry errors** ("fat-finger" mistakes) or **unit mismatches**, like reporting prices in thousands of dollars instead of actual dollars. Including such outliers in a regression can **skew the model**, so they are usually removed or corrected before analysis.

For big data problems, outliers are generally not a problem in fitting the regression to be used in predicting new data. However, outliers are central to anomaly detection, where finding outliers is the whole point. The outlier could also correspond to a case of fraud or an accidental action. In any case, detecting outliers can be a critical business need.

X.2. Influential Values

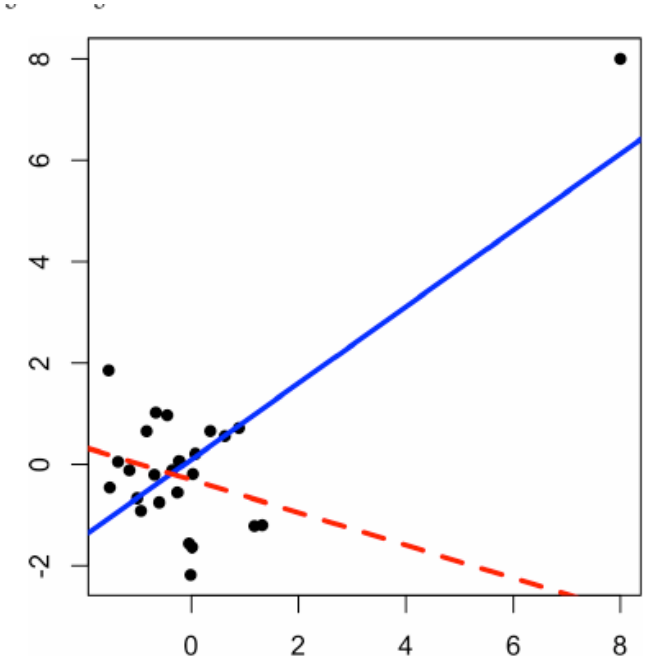


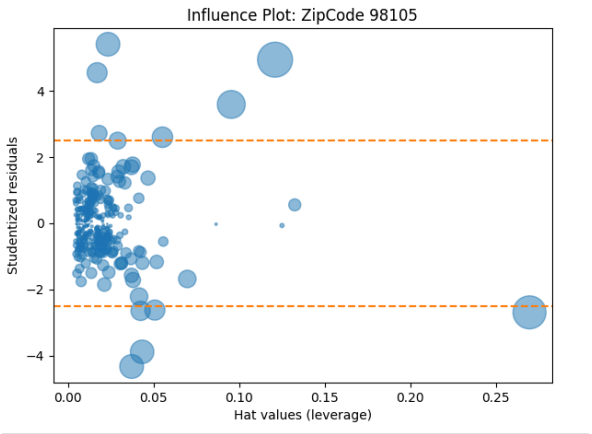
Figure 4-5. An example of an influential data point in regression

Which **record**—a value whose absence would significantly change the **regression equation**—is termed an **influential observation**. In **regression**, such a value need not be associated with a large **residual**. As an example, consider the **regression lines** in *Figure 4-5*. The **solid line** corresponds to the regression with all the data, while the **dashed line** corresponds to the regression with the point in the upper-right corner removed. Clearly, that data value has a huge **influence** on the regression even though it is not associated with a large **outlier** (from the full regression). This data value is considered to have high **leverage** on the regression.

In addition to **standardized residuals** (see *"Outliers"* on page 177), statisticians have developed several metrics to determine the **influence** of a single record on a regression. A common measure of **leverage** is the **hat-value**; values above $2 P + 1 / n$ indicate a high-leverage data value.

Another metric is Cook's distance, which defines influence as a combination of leverage and residual size. A rule of thumb is that an observation has high influence if Cook's distance exceeds $4/ n - P - 1$.

An influence plot or bubble plot combines standardized residuals, the hat-value, and Cook's distance in a single plot. Figure 4-6 shows the influence plot for the King County house data and can be created by python.



There are apparently several data points that exhibit large influence in the regression. Cook's distance can be computed using the function cooks.distance, and you can use hatvalues to compute the diagnostics. The hat values are plotted on the x-axis, the residuals are plotted on the y-axis, and the size of the points is related to the value of Cook's distance.

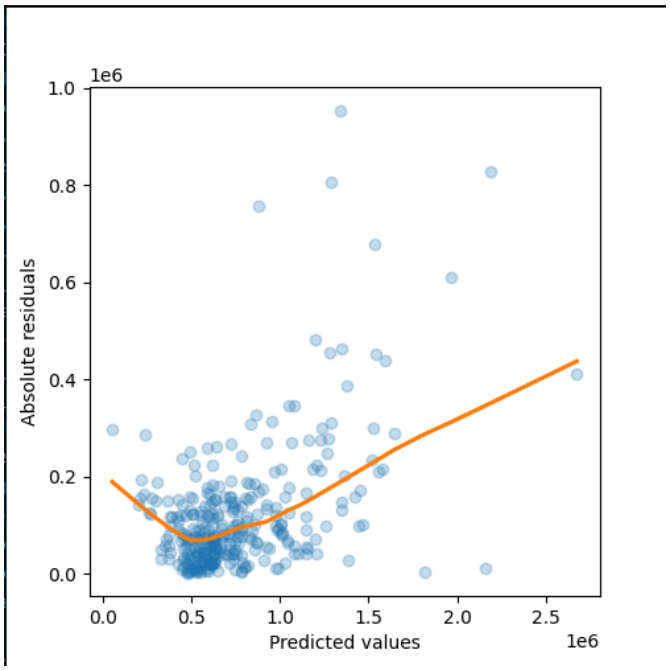
For purposes of fitting a regression that reliably predicts future data, identifying influential observations is useful only in smaller data sets. For regressions involving many records, it is unlikely that any one observation will carry sufficient weight to cause extreme influence on the fitted equation (although the regression may still have big outliers). For purposes of anomaly detection, though, identifying influential observations can be very useful.

X.3. Heteroskedasticity, Non-Normality, and Correlated Errors

The distribution of the residuals is relevant mainly for the validity of formal statistical inference (hypothesis tests and p-values), which is of minimal importance to data scientists concerned mainly with predictive accuracy.

Normally distributed errors are a sign that the model is complete; errors that are not normally distributed indicate the model may be missing something. For formal inference to be fully valid, the residuals are assumed to be normally distributed, have the same variance, and be independent. One area where this may be of concern to data scientists is the standard calculation of confidence intervals for predicted values, which are based upon the assumptions about the residuals (see "Confidence and Prediction Intervals" on page 161).

Heteroskedasticity is the lack of constant residual variance across the range of the predicted values. In other words, errors are greater for some portions of the range than for others. Visualizing the data is a convenient way to analyze residuals.



The following figure is a plot of the absolute residuals versus the predicted values for the lm_98105 regression fit in "Outliers" on page 177)

a smoothed estimate of the relationship between the variables on the x-axis and y-axis in a scatterplot (see "Scatterplot Smoothers" on page 185).

Evidently, the variance of the residuals tends to increase for higher-valued homes but

is also large for lower-valued homes. This plot indicates that lm_98105 has heteroskedastic errors.

Why Would a Data Scientist Care About Heteroskedasticity?

Heteroskedasticity indicates that prediction errors differ for different ranges of the predicted value, and may suggest an incomplete model. For example, the heteroskedasticity in `lm_98105` may indicate that the regression has left something unaccounted for in high- and low-range homes.

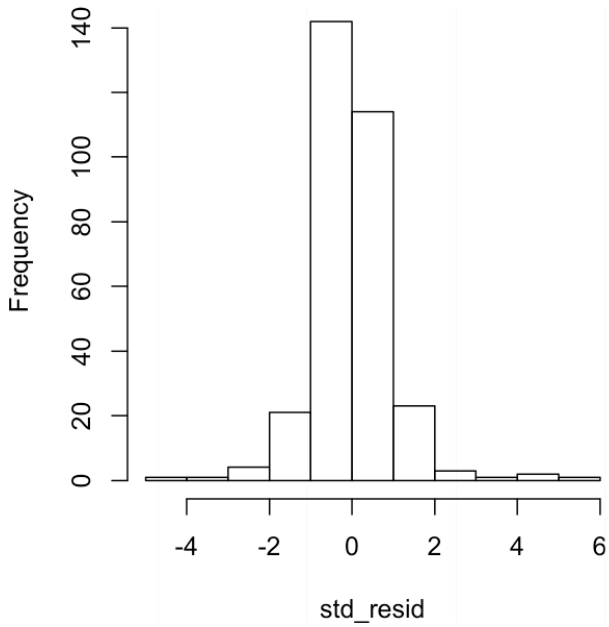


Figure 4-8. A histogram of the residuals from the regression of the housing data

Figure 4-8 is a histogram of the standardized residuals for the lm_98105 regression.

The distribution has decidedly longer tails than the normal distribution and exhibits

mild skewness toward larger residuals.

Even though a regression may violate one of the distributional assumptions, should we care? Most often in data science, the interest is primarily in predictive accuracy, so some review of heteroskedasticity may be in order. You may discover that there is some signal in the data that your model has not captured. However, satisfying distrbutional assumptions simply for the sake of validating formal statistical inference (p-values, F-statistics, etc.) is not that important for the data scientist.

Scatterplot Smoothers

Regression is about modeling the relationship between the response and predictor variables. In evaluating a regression model, it is useful to use a *scatterplot smoother* to visually highlight relationships between two variables.

For example, in Figure 4-7, a smooth of the relationship between the absolute residuals and the predicted value shows that the variance of the residuals depends on the value of the residual. In this

X.4. Partial Residual Plots and Nonlinearity

Partial residual plots are a way to visualize how well the estimated fit explains the relationship between a predictor and the outcome. The basic idea of a partial residual plot is to isolate the relationship between a predictor variable and the response, taking into account all of the other predictor variables. A partial residual might be thought of as a "synthetic outcome" value, combining the prediction based on a single predictor with the actual residual from the full regression equation.

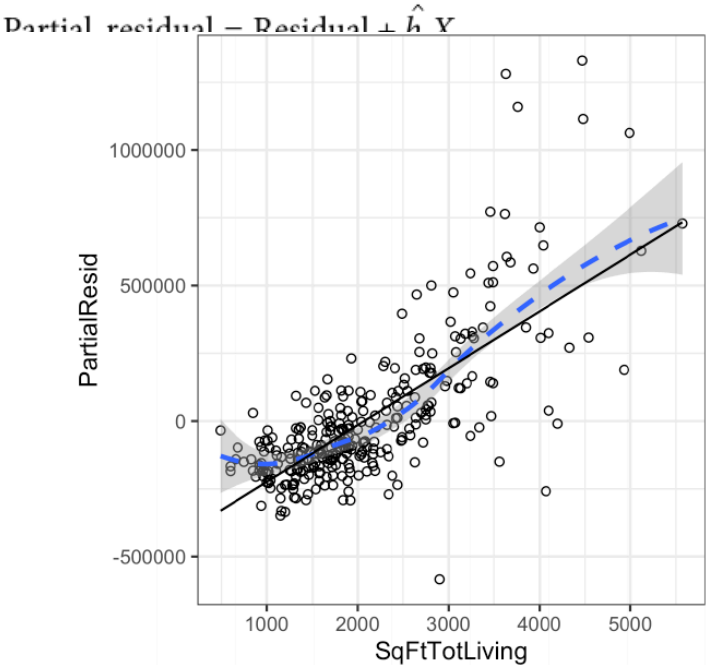


Figure 4-9. A partial residual plot for the variable `SqFtTotLiving`

The resulting plot is shown in Figure 4-9. The partial residual is an estimate of the contribution that SqFtTotLiving adds to the sales price. The relationship between SqFtTotLiving and the sales price is evidently nonlinear (dashed line). The regression line (solid line) underestimates the sales price for homes less than 1,000 square feet and overestimates the price for homes between 2,000 and 3,000 square feet. There are too few data points above 4,000 square feet to draw conclusions for those homes.

This nonlinearity makes sense in this case: adding 500 feet in a small home makes a much bigger difference than adding 500 feet in a large home. This suggests that, instead of a simple linear term for SqFtTotLiving, a nonlinear term should be considered (see “Polynomial and Spline Regression” on page 187).

Key Ideas

- While outliers can cause problems for small data sets, the primary interest with outliers is to identify problems with the data, or locate anomalies.
- Single records (including regression outliers) can have a big influence on a regression equation with small data, but this effect washes out in big data.
- If the regression model is used for formal inference (p-values and the like), then certain assumptions about the distribution of the residuals should be checked. In general, however, the distribution of residuals is not critical in data science.
- The partial residuals plot can be used to qualitatively assess the fit for each regression term, possibly leading to alternative model specification.

XI. Polynomial and Spline Regression

The relationship between the response and a predictor variable isn’t necessarily linear. The response to the dose of a drug is often nonlinear: doubling the dosage generally doesn’t lead to a doubled response. The demand for a product isn’t a linear function of marketing dollars spent; at some point, demand is likely to be saturated. There are many ways that regression can be extended to capture these nonlinear effects.

Key Terms for Nonlinear Regression

Polynomial regression
Adds polynomial terms (squares, cubes, etc.) to a regression.

Spline regression
Fitting a smooth curve with a series of polynomial segments.

Knots
Values that separate spline segments.

Generalized additive models
Spline models with automated selection of knots.

Synonym
GAM

Nonlinear Regression

When statisticians talk about *nonlinear regression*, they are referring to models that can't be fit using least squares. What kind of models are nonlinear? Essentially all models where the response cannot be expressed as a linear combination of the predictors or some transform of the predictors. Nonlinear regression models are harder and computationally more intensive to fit, since they require numerical optimization. For this reason, it is generally preferred to use a linear model if possible.

XI.1. Polynomial

For example, a quadratic regression between the response Y and the predictor X would take the form:

$$Y = b_0 + b_1 X + b_2 X^2 + e$$

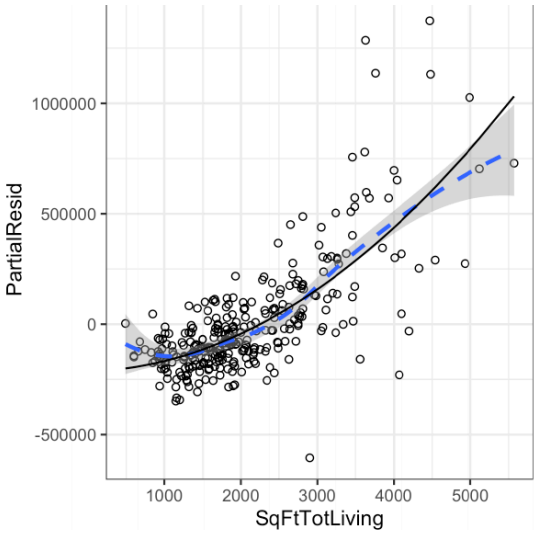
In statsmodels, we add the squared term to the model definition using I(SqFtTotLiving**2)

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           AdjSalePrice   R-squared:                       0.806
Model:                            OLS   Adj. R-squared:                  0.802
Method:                 Least Squares   F-statistic:                     211.6
Date:                Sun, 28 Dec 2025   Prob (F-statistic):          9.95e-106
Time:                        11:11:08   Log-Likelihood:                -4217.9
No. Observations:                 313   AIC:                             8450.
Df Residuals:                     306   BIC:                             8476.
Df Model:                           6
Covariance Type:            nonrobust
===========================================================================================
                              coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------------
Intercept               -6.159e+05   1.03e+05     -5.953      0.000   -8.19e+05   -4.12e+05
SqFtTotLiving              7.4521     55.418      0.134      0.893    -101.597     116.501
I(SqFtTotLiving ** 2)      0.0388      0.010      4.040      0.000       0.020       0.058
SqFtLot                   32.5594      5.436      5.990      0.000      21.863      43.256
Bathrooms              -1435.1231   1.95e+04     -0.074      0.941   -3.99e+04     3.7e+04
Bedrooms               -9191.9441   1.33e+04     -0.693      0.489   -3.53e+04    1.69e+04
BldgGrade               1.357e+05   1.49e+04      9.087      0.000    1.06e+05    1.65e+05
==============================================================================
Omnibus:                       75.161   Durbin-Watson:                   1.625
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              637.978
Skew:                           0.699   Prob(JB):                    2.92e-139
Kurtosis:                       9.853   Cond. No.                     7.37e+07
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.37e+07. This might indicate that there are
strong multicollinearity or other numerical problems.
```

There are now two coefficients associated with SqFtTotLiving: one for the linear term and one for the quadratic term.

The partial residual plot (see "Partial Residual Plots and Nonlinearity" on page 185) indicates some curvature in the regression equation associated with SqFtTotLiving. The fitted line more closely matches the smooth (see "Splines" on page 189) of the partial residuals as compared to a linear fit (see Figure 4-10).



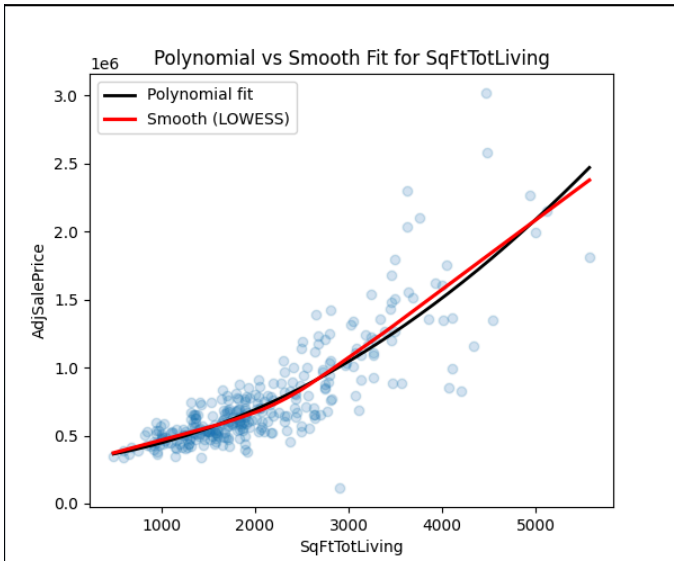


Figure 4-10. A polynomial regression fit for the variable `SqFtTotLiving` (solid line) versus a smooth (dashed line; see the following section about splines)

XI.2. Splines

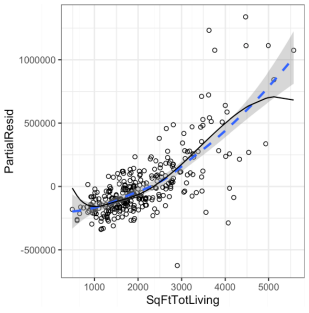
Polynomial regression captures only a certain amount of curvature in a nonlinear relationship. Adding in higher-order terms, such as a cubic quartic polynomial, often leads to undesirable “wiggliness” in the regression equation. An alternative, and often superior, approach to modeling nonlinear relationships is to use splines.

The splines were created by bending a thin piece of wood using weights, referred to as “ducks”; see Figure 4-11.

The polynomial pieces are smoothly connected at a series of fixed points in a predictor variable, referred to as knots. For mulation of splines is much more complicated than polynomial regression; statistical software usually handles the details of fitting a spline.

Two parameters need to be specified: the degree of the polynomial and the location of the knots. the predictor SqFtTotLiving is included in the model using a cubic spline (degree=3)

By default, bs places knots at the boundaries; in addition, knots were also placed at the lower quartile, the median quartile, and the upper quartile.



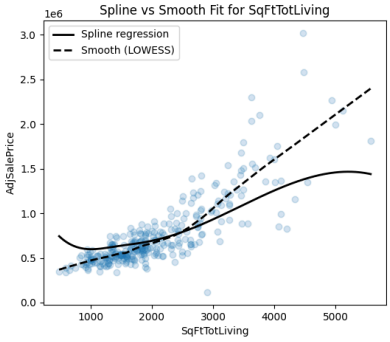


Figure 4-12. A spline regression fit for the variable `SqFtTotLiving` (solid line) compared to a smooth (dashed line)

In contrast to a linear term, for which the coefficient has a direct meaning, the coeffiients for a spline term are not interpretable. Instead, it is more useful to use the visual display to reveal the nature of the spline fit. Figure 4-12 displays the partial residual plot from the regression. In contrast to the polynomial model, the spline model more closely matches the smooth, demonstrating the greater flexibility of splines. In this case, the line more closely fits the data. Does this mean the spline regression is a better model? Not necessarily: it doesn't make economic sense that very small homes (less than 1,000 square feet) would have higher value than slightly larger homes. This is possibly an artifact of a confounding variable; see "Confounding Variables" on page 172.

XI.3. Generalized Additive Models

Suppose you suspect a nonlinear relationship between the response and a predictor variable, either by a priori knowledge or by examining the regression diagnostics. Polynomial terms may not be flexible enough to capture the relationship, and spline terms require specifying the knots. Generalized additive models, or GAM, are a flexible modeling technique that can be used to automatically fit a spline regression.

In Python, we can use the pyGAM package. It provides methods for regression and classification. Here, we use LinearGAM to create a regression model.

The default value for n_splines is 20. This leads to overfitting for larger SqFtTotLiving values. A value of 12 leads to a more reasonable fit.

Key Ideas

- Outliers in a regression are records with a large residual.
- Multicollinearity can cause numerical instability in fitting the regression equation.
- A confounding variable is an important predictor that is omitted from a model and can lead to a regression equation with spurious relationships.
- An interaction term between two variables is needed if the effect of one variable depends on the level or magnitude of the other.
- Polynomial regression can fit nonlinear relationships between predictors and the outcome variable.
- Splines are series of polynomial segments strung together, joining at knots.
- We can automate the process of specifying the knots in splines using generalized additive models (GAM).

Summary

Perhaps no other statistical method has seen greater use over the years than regression—the process of establishing a relationship between multiple predictor variables and an outcome variable. The fundamental form is linear: each predictor variable has a coefficient that describes a linear relationship between the predictor and the outcome. More advanced forms of regression, such as polynomial and spline regression, permit the relationship to be nonlinear. In classical statistics, the emphasis is on finding a good fit to the observed data to explain or describe some phenomenon, and the strength of this fit is how traditional *in-sample* metrics are used to assess the model. In data science, by contrast, the goal is typically to predict values for new data, so metrics based on predictive accuracy for out-of-sample data are used. Variable selection methods are used to reduce dimensionality and create more compact models.