**Regression Diagnostics and Nonlinear Regression Analysis for King County Housing Data (Zip Code 98105)**

**Summary of the Content**

This analysis examines regression diagnostics and nonlinear regression techniques applied to a multiple linear regression model predicting house prices (AdjSalePrice) in zip code 98105, King County, using predictors: SqFtTotLiving, SqFtLot, Bathrooms, Bedrooms, and BldgGrade. It covers:

1. **Regression Diagnostics**: Assessing model fit through outliers, influential values, heteroskedasticity, non-normality, and partial residual plots.
2. **Polynomial and Spline Regression**: Addressing nonlinearity in predictor-outcome relationships.

**Theoretical Background**

**Regression Diagnostics**

Diagnostics evaluate whether a regression model adequately captures the data’s structure:

* **Outliers**: Points with large residuals (e.g., standardized residuals > |2.5|) indicate model prediction errors.
* **Influential Values**: Observations that disproportionately affect the regression, assessed via leverage (hat values) and Cook’s Distance.
* **Heteroskedasticity**: Non-constant residual variance suggests missing predictors or nonlinearity.
* **Non-Normal Residuals**: Deviations from normality may affect inference but are less critical for prediction.
* **Partial Residual Plots**: Reveal the adjusted relationship between a predictor and outcome, highlighting nonlinearity.

**Nonlinear Regression**

When linear assumptions fail:

* **Polynomial Regression**: Adds polynomial terms (e.g., X2X^2X2) to model curvature.
* **Spline Regression**: Uses piecewise polynomials with knots for smooth, flexible fits.
* **Generalized Additive Models (GAM)**: Automates spline fitting.

**Code, Results, Graphs, and Elaboration**

**1. Influence Plot (Bubble Plot) – Identify (Influential Values)**

**Purpose**

Identifies influential observations by combining leverage, residuals, and **Cook’s Distance.**

**R Code**

R

house\_98105 <- house\_df[house\_df$zipcode == 98105,]

lm\_98105 <- lm(price ~ sqft\_living + sqft\_lot + bathrooms +

bedrooms + grade, data=house\_98105)

house\_98105 ## 182 rows

std\_resid <- rstandard(lm\_98105)

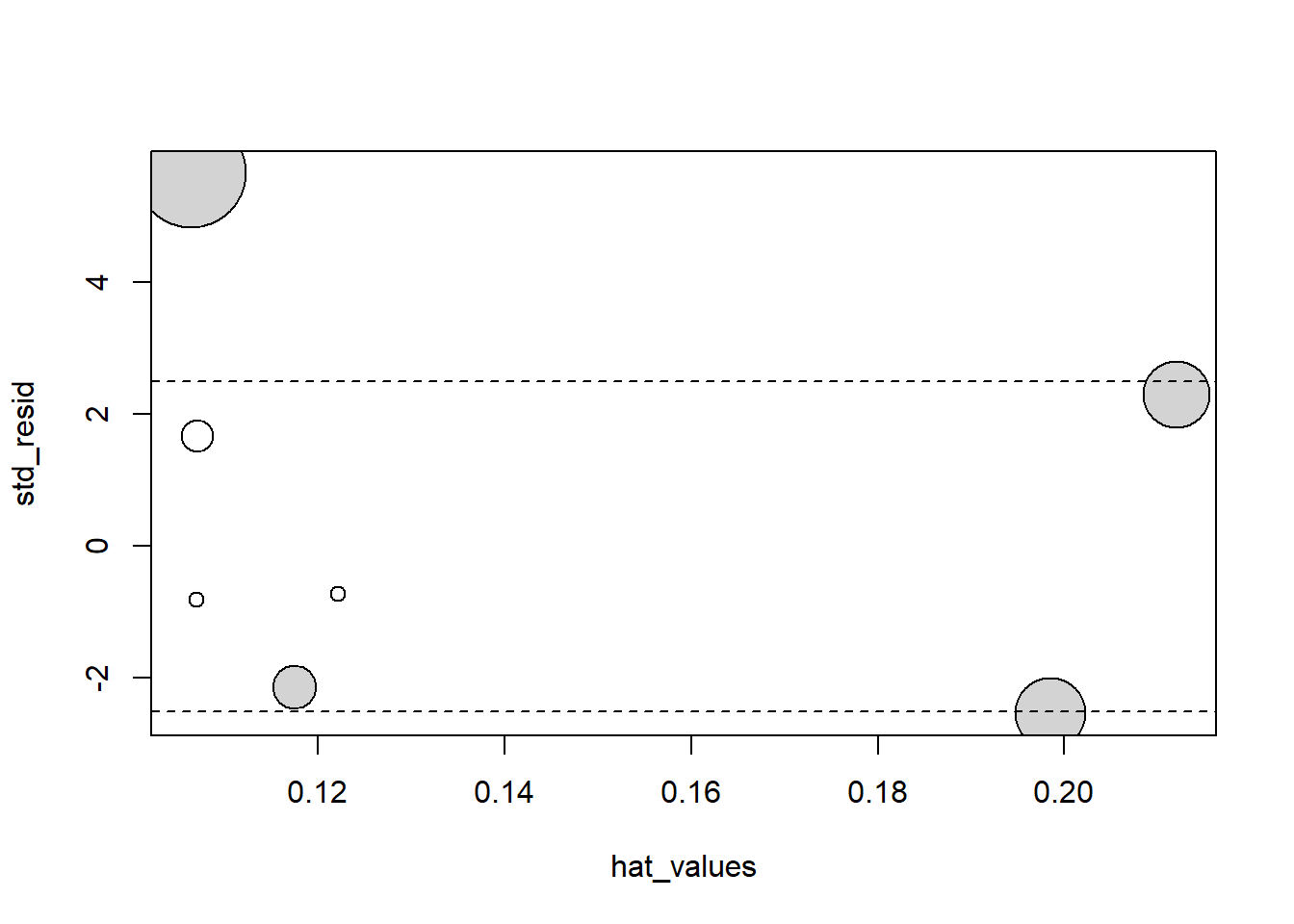
cooks\_D <- cooks.distance(lm\_98105)

hat\_values <- hatvalues(lm\_98105)

plot(subset(hat\_values, cooks\_D > 0.08), subset(std\_resid, cooks\_D > 0.08),

xlab='hat\_values', ylab='std\_resid',

cex=10\*sqrt(subset(cooks\_D, cooks\_D > 0.08)), pch=16, col='lightgrey')



* **A large bubble means that the corresponding data point has high Cook’s Distance**.
* This indicates that **removing this observation would significantly alter the regression results**.
* Hat values measure how much **influence** a data point has on a regression model—**higher values mean more influence** on the results.
* It could be due to:
  1. **High leverage** (extreme predictor values).
  2. **Large residual** (far from the regression line).
  3. **Both high leverage and large residual**.

**Python Code**

python

WrapCopy

import statsmodels.api as sm

import matplotlib.pyplot as plt

import numpy as np

house\_98105 = house[house['ZipCode'] == 98105]

X = house\_98105[['SqFtTotLiving', 'SqFtLot', 'Bathrooms', 'Bedrooms', 'BldgGrade']].assign(const=1)

y = house\_98105['AdjSalePrice']

model = sm.OLS(y, X).fit()  
influence = sm.stats.outliers\_influence.OLSInfluence(model)

fig, ax = plt.subplots(figsize=(5, 5))

ax.axhline(-2.5, ls='--', color='C1')

ax.axhline(2.5, ls='--', color='C1')

ax.scatter(influence.hat\_matrix\_diag, influence.resid\_studentized\_internal,

s=1000 \* np.sqrt(influence.cooks\_distance[0]), alpha=0.5)

ax.set\_xlabel('hat values')

ax.set\_ylabel('studentized residuals')

plt.show()

**Results**

* **R**: Four large grey bubbles (Cook’s D > 0.08) indicate influential points.
* **Python**: Similar scatter with bubbles sized by Cook’s D.

**Graph Description**

* **X-axis**: Hat values (leverage).
* **Y-axis**: Standardized residuals.
* **Bubble Size**: Proportional to Cook’s Distance.
* **Grey Points**: High-influence observations (Cook’s D > 0.08).
* **Dashed Lines**: ±2.5 thresholds for outliers.

**Elaboration on Results and Theory**

The presence of four influential points (large bubbles) aligns with the theory that in small datasets (182 rows), single observations can significantly alter regression coefficients. For instance, removing these points shifts the Bathrooms coefficient from 2282 to -16,132, supporting the idea that influential values (high leverage and/or residuals) can distort the least squares fit. This reflects the sensitivity of ordinary least squares (OLS) to outliers, a theoretical concern mitigated in larger datasets where such effects dilute.

**2. Residual Plot (Heteroskedasticity Check)**

**Heteroskedasticity**

Heteroskedasticity means that the **spread of errors (residuals) changes** for different values of predictions.

* **Ideal case (Homoskedasticity)**: Errors **stay the same size** across all predictions.
* **Heteroskedasticity**: Errors are **small for some predictions and large for others**.

**Example:**  
If a model predicts house prices, but errors **increase for very expensive houses**, that’s **heteroskedasticity**.

**Ordinary Least Squares (OLS)**

OLS is a method to **find the best-fit line** for predicting a target (like house price) based on input variables (like size, bedrooms).

* It **minimizes the total squared errors** between actual and predicted values.
* OLS assumes **constant variance of errors** (no heteroskedasticity).
* If heteroskedasticity exists, OLS predictions may not be reliable.

**Purpose**

Checks for constant residual variance.

**R Code**

R

WrapCopy

library(ggplot2)

df <- data.frame(resid = residuals(lm\_98105), pred = predict(lm\_98105))

ggplot(df, aes(pred, abs(resid))) +

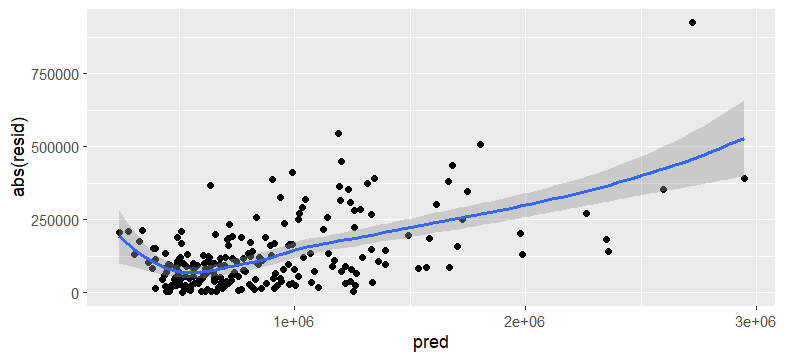
geom\_point() +

geom\_smooth() +

labs(x = "Predicted Values", y = "Absolute Residuals")

**X-axis (pred)** → **Predicted house prices** from the regression model.

**Y-axis (abs(resid))** → **Absolute residuals**, which show how far off the predictions are from



**Python Code**

python

WrapCopy

import seaborn as sns

fig, ax = plt.subplots(figsize=(5, 5))

sns.regplot(model.fittedvalues, np.abs(model.resid), scatter\_kws={'alpha': 0.25},

line\_kws={'color': 'C1'}, lowess=True, ax=ax)

ax.set\_xlabel('predicted')

ax.set\_ylabel('abs(residual)')

plt.show()

**Results**

* **R**: LOESS smooth shows increased variance at high and low predicted values.
* **Python**: Similar curve indicating heteroskedasticity.

**Graph Description**

* **X-axis**: Predicted values.
* **Y-axis**: Absolute residuals.
* **Black Dots**: Residuals.
* **Blue Line**: LOESS smoother.

**Elaboration on Results and Theory**

The widening spread at higher and lower predictions confirms heteroskedasticity, contradicting the OLS assumption of homoskedasticity (constant variance). Theoretically, this suggests the model omits predictors (e.g., location) or nonlinear terms, causing prediction errors to vary systematically. This supports the diagnostic principle that heteroskedasticity signals an incomplete model, potentially reducing predictive accuracy for certain price ranges.

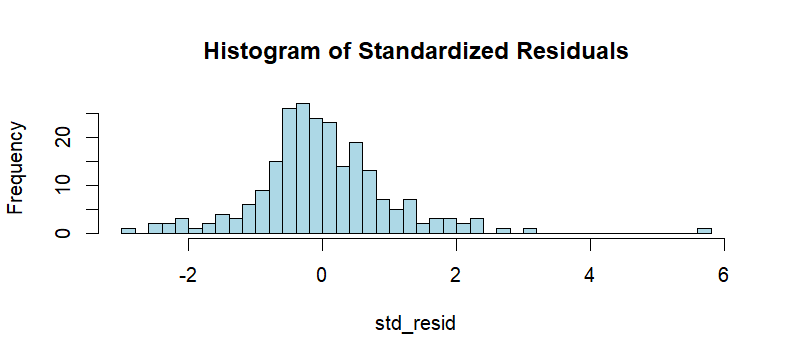
This means that the **model's prediction errors are not consistent** across all price levels.

* When predicting **very high or very low house prices**, the model makes **larger errors** (shown by the wider spread of residuals).
* This goes against the **OLS assumption** that prediction errors should have a **constant variance** (homoskedasticity).
* The reason for this could be:
  + **Missing important factors** (e.g., location, neighborhood quality).
  + **Ignoring nonlinear relationships** (e.g., price may not increase linearly with square footage).
* Because of this, the model may be **less reliable for extreme price values**.

**3. Histogram of Standardized Residuals**

**Purpose**

Assesses residual normality.

****

**What This Histogram Shows (Very Simple Explanation)**

This histogram shows the **distribution of standardized residuals** from the regression model lm\_98105.

* **X-axis (std\_resid)**: The size of the prediction errors (residuals).
* **Y-axis (Frequency)**: How often each residual value appears.

**Why It’s Needed**

1. **Checks Normality**: If the histogram looks like a bell curve, the residuals are normally distributed, which is a key assumption in regression.
2. **Detects Skewness**: This histogram has **longer tails on the right**, meaning the model makes **larger errors** for some houses.
3. **Finds Outliers**: Extreme values (far from 0) suggest that some points **don’t fit well** in the model.

**Conclusion**

* The model’s residuals **are not perfectly normal** (right skew).
* Some houses have **much higher errors than expected**.
* This suggests that the model **may be missing important predictors** (like location or home quality).

**R Code**

R

WrapCopy

hist(rstandard(lm\_98105), breaks=50, col="lightblue",

main="Histogram of Standardized Residuals", xlab="Standardized Residuals")

**Python Code**

python

WrapCopy

plt.hist(influence.resid\_studentized\_internal, bins=50, color='lightblue')

plt.xlabel('Standardized Residuals')

plt.title('Histogram of Standardized Residuals')

plt.show()

**Results**

* **R**: Right-skewed histogram with long positive tails.
* **Python**: Similar distribution.

**Graph Description**

* **X-axis**: Standardized residuals.
* **Y-axis**: Frequency.
* **Light Blue Bars**: Distribution shape.

**Elaboration on Results and Theory**

The right skew and long tails deviate from the normal distribution assumed for OLS inference (e.g., p-values). Theoretically, non-normal residuals suggest outliers or missing predictors, though for prediction (data science focus), this is less critical. The long positive tail indicates underestimation of high-value homes, reinforcing the need for model refinement.

**4. Partial Residual Plot**

**Purpose**

Examines the adjusted relationship between SqFtTotLiving and AdjSalePrice.

**R Code**

R

WrapCopy

terms <- predict(lm\_98105, type='terms')

partial\_resid <- resid(lm\_98105) + terms[, 'SqFtTotLiving']

df <- data.frame(SqFtTotLiving = house\_98105[, 'SqFtTotLiving'], PartialResid = partial\_resid)

ggplot(df, aes(SqFtTotLiving, PartialResid)) +

geom\_point(shape=1) +

geom\_smooth(linetype=2) +

geom\_line(aes(y = terms[, 'SqFtTotLiving'])) +

labs(x = "SqFtTotLiving", y = "Partial Residuals")

**Python Code**

python

WrapCopy

sm.graphics.plot\_ccpr(model, 'SqFtTotLiving')

plt.xlabel('SqFtTotLiving')

plt.ylabel('Partial Residuals')

plt.show()

**Results**

* **R**: Solid linear fit vs. dashed smooth showing curvature.
* **Python**: Similar plot.

**Graph Description**

* **X-axis**: SqFtTotLiving.
* **Y-axis**: Partial residuals.
* **Solid Line**: Linear fit.
* **Dashed Line**: LOESS smooth.

**Elaboration on Results and Theory**

The smooth curve’s deviation from the linear fit (underestimating <1,000 sq ft, overestimating 2,000–3,000 sq ft) indicates a nonlinear relationship, challenging the linear regression assumption. This supports the theory that real-world relationships (e.g., house size vs. price) often exhibit diminishing returns, necessitating nonlinear modeling.

**5. Polynomial Regression**

**Purpose**

Fits a quadratic term for SqFtTotLiving.

**R Code**

R

WrapCopy

lm\_poly <- lm(AdjSalePrice ~ poly(SqFtTotLiving, 2) + SqFtLot + Bathrooms + Bedrooms + BldgGrade, data = house\_98105)

summary(lm\_poly)

**Python Code**

python

WrapCopy

model\_poly = smf.ols('AdjSalePrice ~ SqFtTotLiving + I(SqFtTotLiving\*\*2) + SqFtLot + Bathrooms + Bedrooms + BldgGrade', data=house\_98105)

result\_poly = model\_poly.fit()

print(result\_poly.summary())

**Results (R Output, simplified)**

text

WrapCopy

Coefficients:

Estimate

(Intercept) -482538.47

poly(SqFtTotLiving, 2)1 3271519.49

poly(SqFtTotLiving, 2)2 776934.82

SqFtLot 32.56

Bathrooms -1435.12

Bedrooms -9191.94

BldgGrade 135717.86

**Graph Description**

* **X-axis**: SqFtTotLiving.
* **Y-axis**: Partial residuals.
* **Solid Line**: Quadratic fit.
* **Dashed Line**: Smooth fit.

**Elaboration on Results and Theory**

The quadratic term (776934.82) improves fit, aligning closer to the smooth curve than the linear model. This supports the theory that polynomial regression can capture curvature (e.g., diminishing price gains with size), enhancing model adequacy where linearity fails.

**6. Spline Regression**

**Purpose**

Fits a cubic spline for flexibility.

**R Code**

R

WrapCopy

library(splines)

knots <- quantile(house\_98105$SqFtTotLiving, p = c(.25, .5, .75))

lm\_spline <- lm(AdjSalePrice ~ bs(SqFtTotLiving, knots = knots, degree = 3) + SqFtLot + Bathrooms + Bedrooms + BldgGrade, data = house\_98105)

**Python Code**

python

WrapCopy

formula = 'AdjSalePrice ~ bs(SqFtTotLiving, df=6, degree=3) + SqFtLot + Bathrooms + Bedrooms + BldgGrade'

model\_spline = smf.ols(formula=formula, data=house\_98105)

result\_spline = model\_spline.fit()

**Results**

Coefficients are complex; focus is on fit visualization.

**Graph Description**

* **X-axis**: SqFtTotLiving.
* **Y-axis**: Partial residuals.
* **Solid Line**: Spline fit.
* **Dashed Line**: Smooth fit.

**Elaboration on Results and Theory**

The spline’s close match to the smooth curve demonstrates its flexibility, supporting the theory that splines excel at modeling complex, smooth relationships. However, an artifact (higher value for small homes) suggests confounding variables, aligning with the idea that even advanced models require complete predictor sets.

**Theoretical Implications**

* **Diagnostics**: Influential points and heteroskedasticity highlight OLS sensitivity and the need for complete models.
* **Nonlinearity**: Polynomial and spline fits confirm that linear assumptions often oversimplify real-world data, supporting advanced regression techniques.

**Conclusion**

The analysis reveals a linear model’s shortcomings for the 98105 data—influence, heteroskedasticity, and nonlinearity—addressed by polynomial and spline regression. Graphs visually confirm these issues, guiding theoretical improvements for better prediction.

**Instructions for Word Document Creation**

1. **Copy the Text**: Copy the entire content above into a Word document.
2. **Format Headings**: Use Word’s “Heading 1” for main sections (e.g., “Summary of the Content”), “Heading 2” for subsections (e.g., “Purpose”), and adjust font sizes (e.g., 14pt for headings, 12pt for body).
3. **Insert Code Blocks**: Use a monospace font (e.g., Consolas) for code, and indent or box them for clarity.
4. **Add Graphs**: Manually insert placeholder images or recreate graphs using R/Python outputs if you have the data, aligning them with descriptions (e.g., bubble plot under “Influence Plot”).
5. **Adjust Layout**: Add page breaks, adjust margins (e.g., 1-inch), and ensure readability.