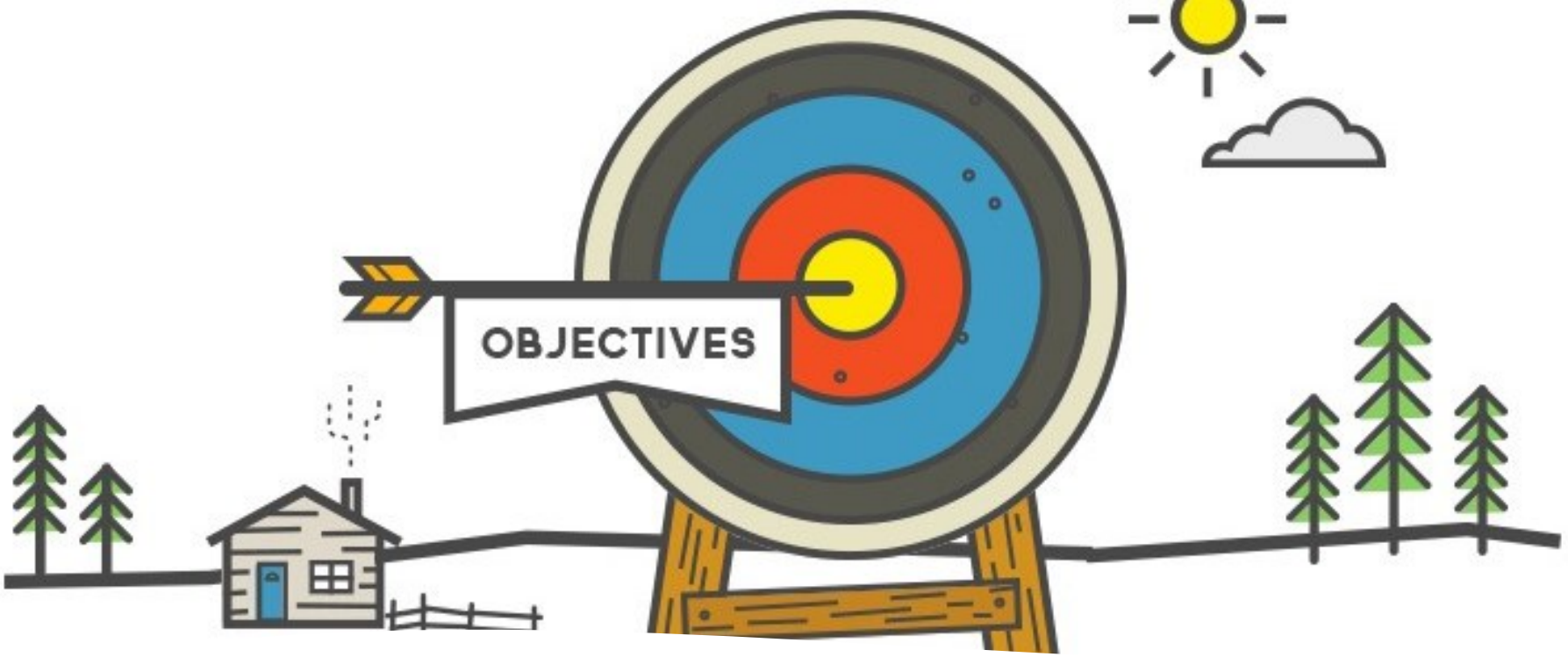
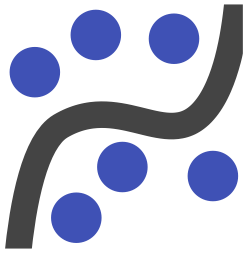
Regression

Carlos J. Costa



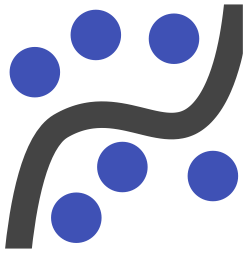
Learning Goals

- Students should use the main libraries to create and fit regression model.
- Students should use the main libraries to analyse regression model.



statsmodels

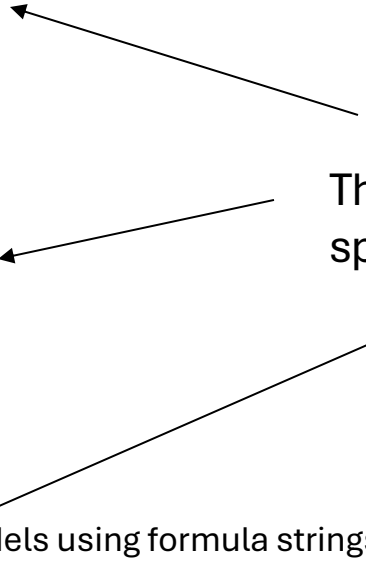
- Python module
- provides classes and functions
- estimation statistical models,
- statistical tests,
- statistical data exploration.
- open source Modified BSD (3-clause) license.
- https://www.statsmodels.org/



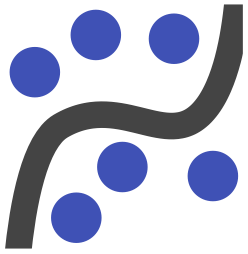
statsmodels

- Regression and Linear Models
- Time Series Analysis
- Other Models (e.g. Non parametric models)
- Statistics and Tools
- Data Sets

- **statsmodels.api:**
  - Cross-sectional models and methods.
  - Canonically imported using
  - `import statsmodels.api as sm`
- **statsmodels.tsa.api:**
  - Time-series models and methods.
  - Canonically imported using:
  - `import statsmodels.tsa.api as tsa`
- **statsmodels.formula.api:**
  - A convenience interface for specifying models using formula strings and DataFrames.
  - Canonically imported using:
  - `import statsmodels.formula.api as smf`



The main statsmodels API is split into models:

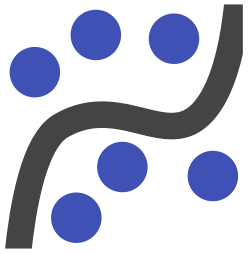


statsmodels

Regression and Linear Models

- Linear Regression
- Generalized Linear Models
- Generalized Estimating Equations
- Generalized Additive Models (GAM)
- Robust Linear Models
- Linear Mixed Effects Models
- Regression with Discrete Dependent Variable
- Generalized Linear Mixed Effects Models
- ANOVA

https://www.statsmodels.org/stable/user-guide.html#regression-and-linear-models



statsmodels

Linear Regression

- Linear models with independently and identically distributed errors, and for errors with heteroscedasticity or autocorrelation.
- This module allows estimation by:
  - ordinary least squares (OLS),
  - weighted least squares (WLS),
  - generalized least squares (GLS), and
  - ..

OLS (Ordinary Least Squares)

- **Definition:** OLS is the standard method for **Linear Regression**, which estimates model coefficients by minimizing the sum of squared residuals.
- **Equation:**

$$y=\beta_0+\beta_1X_1+\beta_2X_2+\ldots+\beta_nX_n+\epsilon$$

- **Assumptions:**
  - Linearity
  - Homoscedasticity (constant variance of errors)
  - Independence of errors
  - No multicollinearity
  - Normally distributed residuals
- **Example:** Predicting house prices based on square footage and number of bedrooms.

Generalized Linear Models (GLM)

- **Definition:** Extends OLS regression to allow **non-normal response variables** (e.g., binary, count data).
- Instead of assuming normally distributed errors, it uses an **exponential family of distributions** (e.g., Poisson, Binomial).
- **Components:**
  - **Random Component**: The distribution of the response variable (Normal, Poisson, Binomial).
  - **Systematic Component**: A linear combination of predictors.
  - **Link Function**: Transforms the mean of the response variable (e.g., Logit, Log, Identity).
- **Example:** Logistic Regression (for binary classification) is a GLM with a **logit link function**.

Generalized Estimating Equations (GEE)

- **Definition:** Used when dealing with **correlated data**, such as repeated measurements on the same subjects (longitudinal data). Instead of assuming independent observations, GEE accounts for **within-subject correlation**.
- **Advantage:** Provides **robust standard errors**, even if the correlation structure is misspecified.
- **Example:** Analyzing repeated blood pressure measurements for patients over time.

Generalized Additive Models (GAM)

- **Definition:** Extends Generalized Linear Models (GLM) by allowing **nonlinear relationships** between predictors and the response variable.
- Instead of assuming linearity, GAM uses **smooth functions** (splines, loess) to model complex relationships.
- **Equation:**

$$y=\beta_0+f_1(X_1)+f_2(X_2)+\ldots+f_n(X_n)+\epsilon$$

where $f_i(X_i)$ are smooth functions rather than simple linear terms.

- **Example:** Modeling temperature effects on crop yield, where the relationship is non-linear.

Robust Linear Models

- **Definition**: A modification of OLS that reduces the influence of outliers. Instead of minimizing squared errors, it minimizes absolute errors or uses M-estimators.
- When to Use:
  - When data contains outliers.
  - When residuals are not normally distributed.

- Example: Predicting income based on education level when data contains extreme salaries.

Linear Mixed Effects Models (LMM)

- **Definition**: Used for hierarchical or grouped data, where observations are not fully independent. LMM includes:
  - Fixed effects (like standard regression coefficients).
  - Random effects (to model group-level variations).

Example: Modeling student test scores across different schools, where each school introduces random variability.

Compare some alterantives

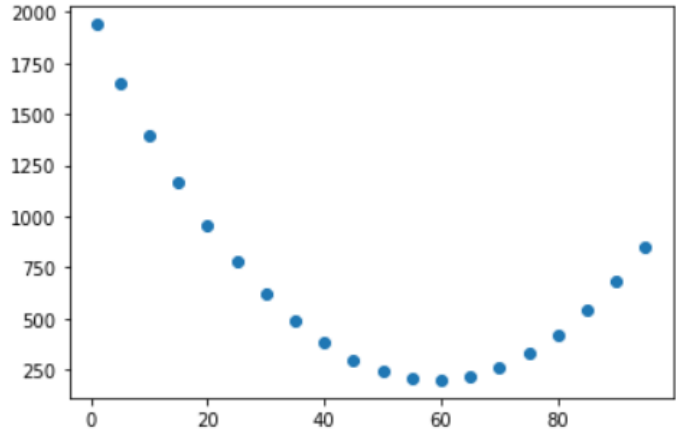
Model Type	Handles **Non-Normal** Data?	Handles **Correlated** Data?	Handles **Nonlinear** Relationships?	Robust to **Outliers**?
OLS / Linear Regression	No	No	No	No
GLM	Yes	No	No	No
GEE	Yes	Yes	No	Yes
GAM	Yes	No	Yes	No
Robust Linear Model	No	No	No	Yes
Linear Mixed Effects Model	No	Yes	No	No

statsmodels

```
import matplotlib.pyplot as plt
```

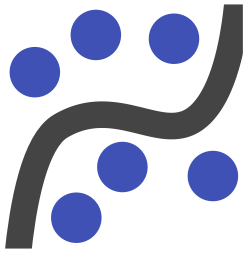
```
x=[1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95]
y=[1940,1650,1400,1170,960,780,620,490,380,300,240,210,200,220,260,330,420,540,680,850]
```

```
plt.plot(x,y,'o')
plt.draw()
```



Linear Regression

A Data Set



statsmodels

- What is the best model to explain
- `Y=f(X)` ?
- A linear Model?

statsmodels

```
import statsmodels.api as sm
x1 = sm.add_constant(x)
model=sm.OLS(y, x1)
result=model.fit()
z=result.predict(x1)
result.summary()
```

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.441
Model:	OLS	**Adj. R-squared:**	0.410
Method:	Least Squares	**F-statistic:**	14.21
Date:	Wed, 03 Mar 2021	**Prob (F-statistic):**	0.00140
Time:	14:32:02	**Log-Likelihood:**	-146.69
No. Observations:	20	**AIC:**	297.4
Df Residuals:	18	**BIC:**	299.4
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1226.9153	168.921	7.263	0.000	872.025	1581.805
x1	-11.4598	3.040	-3.770	0.001	-17.847	-5.073

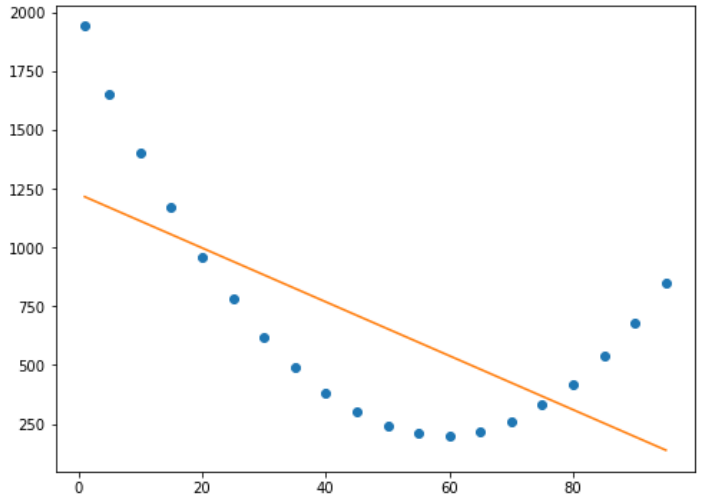
Omnibus:	2.599	**Durbin-Watson:**	0.132
Prob(Omnibus):	0.273	**Jarque-Bera (JB):**	1.974
Skew:	0.625	**Prob(JB):**	0.373
Kurtosis:	2.101	**Cond. No.**	107.

- Import modules
- Add constant
- Create model
- Fit model
- Use model to predict
- Use model and show summary

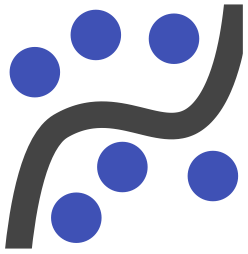
statsmodels

```
fig, ax = plt.subplots(figsize=(8,6))
ax.plot(x,y,'o')
ax.plot(x,z,'-')
```

```
[<matplotlib.lines.Line2D at 0x208ef0e25b0>]
```



- Graphical analysis
- Estimation vs. effective data



statsmodels

- What is the best model to explain
- `Y=f(X)` ?
- A quadratic model?

Create Model

- Create data frame
- Add columns
- Identify X and Y
- Create model
- Fit model

```
import pandas as pd
xy=[x,y]
df=pd.DataFrame(xy)
df1=df.T
df1.columns = ['x','y']
```

```
df1['x2']=df1['x']**2
y=df1['y']
xx=df1[['x','x2']]
```

```
xx1=sm.add_constant(xx)
model=sm.OLS(y, xx1)
result=model.fit()
result.summary()
```

Results

- Bad statistics
- But...

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.999
Model:	OLS	**Adj. R-squared:**	0.999
Method:	Least Squares	**F-statistic:**	1.663e+04
Date:	Wed, 03 Mar 2021	**Prob (F-statistic):**	1.05e-28
Time:	14:49:02	**Log-Likelihood:**	-76.719
No. Observations:	20	**AIC:**	159.4
Df Residuals:	17	**BIC:**	162.4
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	1959.4584	7.518	260.650	0.000	1943.598	1975.319
x	-59.7517	0.367	-162.928	0.000	-60.525	-58.978
x2	0.5065	0.004	136.291	0.000	0.499	0.514

Omnibus:	22.125	**Durbin-Watson:**	1.693
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	39.243
Skew:	1.691	**Prob(JB):**	3.01e-09
Kurtosis:	8.971	**Cond. No.**	1.16e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.16e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

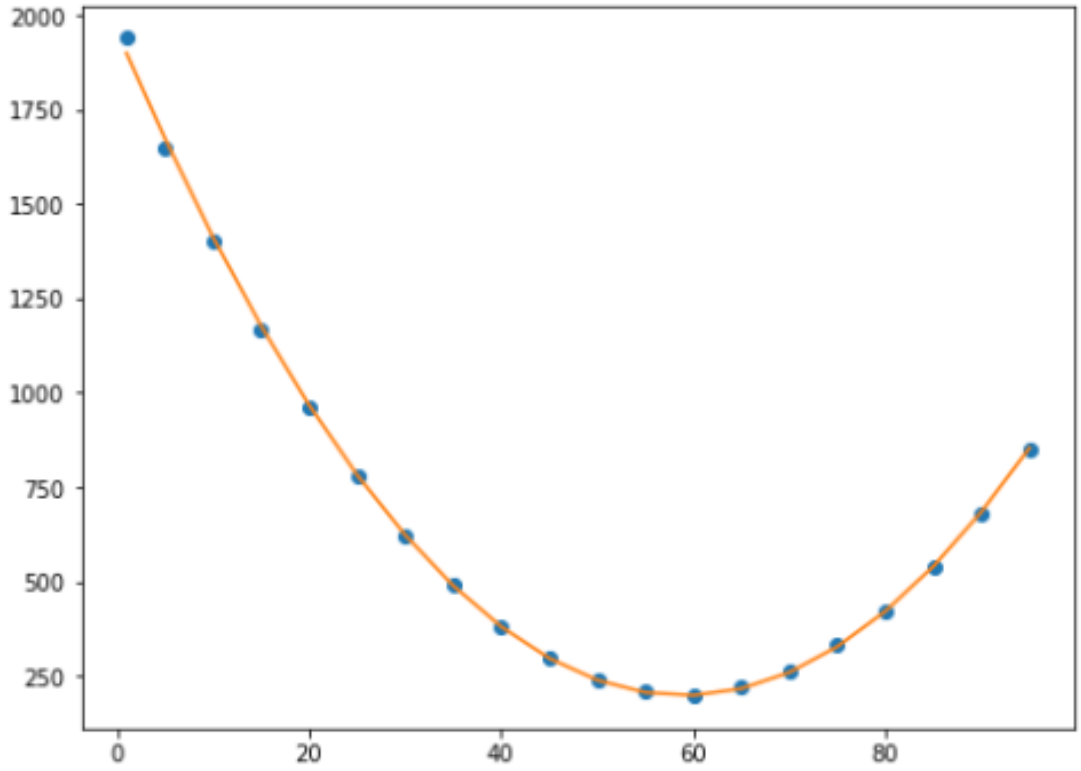
Results

A graphical analysis allows to verify that we obtained a good fit.

```
z=result.predict(xx1)
```

```
fig, ax = plt.subplots(figsize=(8,6))
ax.plot(x,y,'o')
ax.plot(x,z,'-')
```

```
[<matplotlib.lines.Line2D at 0x208ef20d100>]
```



OLS Assumptions:

- **1. Linearity**
- **2. No Multicollinearity**
- 3. Homoscedasticity (Constant Variance of Errors)
- **4. No Autocorrelation of Errors**
- **5. Normality of Residuals**

1. Linearity

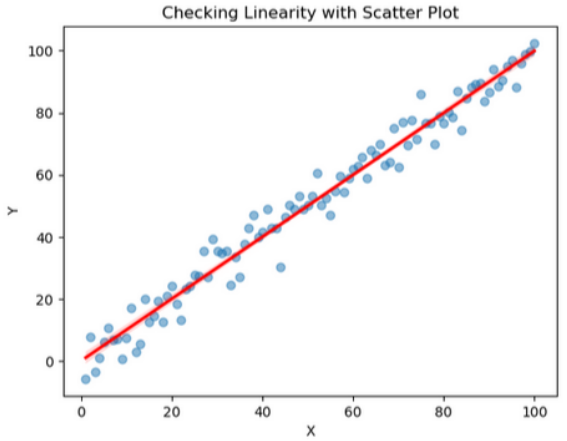
- OLS assumes a linear relationship between the independent variables and the dependent variable.
- Verification:
  - Scatter plots: Visualize the relationship between independent variables and the dependent variable.
  - Residual plots: Plot residuals against fitted values to check for nonlinearity.
  - Polynomial features: Fit a polynomial regression to see if higher-order terms improve the model.

1. Linearity

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import matplotlib.pyplot as plt
import seaborn as sns
# Example dataset
df = pd.DataFrame({
    'X': np.linspace(1, 100, 100),
    'Y': np.linspace(1, 100, 100) + np.random.normal(0, 5, 100)
})
# Fit OLS model
X = sm.add_constant(df['X'])  # Add intercept
model = sm.OLS(df['Y'], X).fit()
```

1. Linearity

```
# Scatter plot

sns.regplot(x=df['X'], y=df['Y'],
scatter_kws={'alpha':0.5},
line_kws={"color":"red"})

plt.title("Checking Linearity with Scatter Plot")

plt.show()

# Residual plot

plt.scatter(model.fittedvalues, model.resid,
alpha=0.5)

plt.axhline(y=0, color='red', linestyle='--')

plt.xlabel("Fitted Values")

plt.ylabel("Residuals")

plt.title("Residual Plot for Linearity Check")

plt.show()
```



2. No Multicollinearity

- Multicollinearity occurs when independent variables are highly correlated.
- **Verification:**
  - **Correlation matrix**: Check pairwise correlations.
  - **Variance Inflation Factor (VIF)**: VIF > 10 suggests high multicollinearity.

2. No Multicollinearity

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Create feature matrix

X = pd.DataFrame({'X1': np.random.rand(100), 'X2': np.random.rand(100) * 2,
'X3': np.random.rand(100) * 3})

X = sm.add_constant(X)

# Compute VIF for each variable

vif_values = []

for i in range(X.shape[1]):

    vif = variance_inflation_factor(X.values, i)

    vif_values.append((X.columns[i], vif))


# Convert to DataFrame

vif_data = pd.DataFrame(vif_values, columns=["Variable", "VIF"])


# Display the results

print(vif_data)
```

2. No Multicollinearity

	Variable	VIF
0	const	11.685989
1	X1	1.010828
2	X2	1.013169
3	X3	1.002550

VIF close to or above 10 suggests multicollinearity, and one of the variables should be removed.

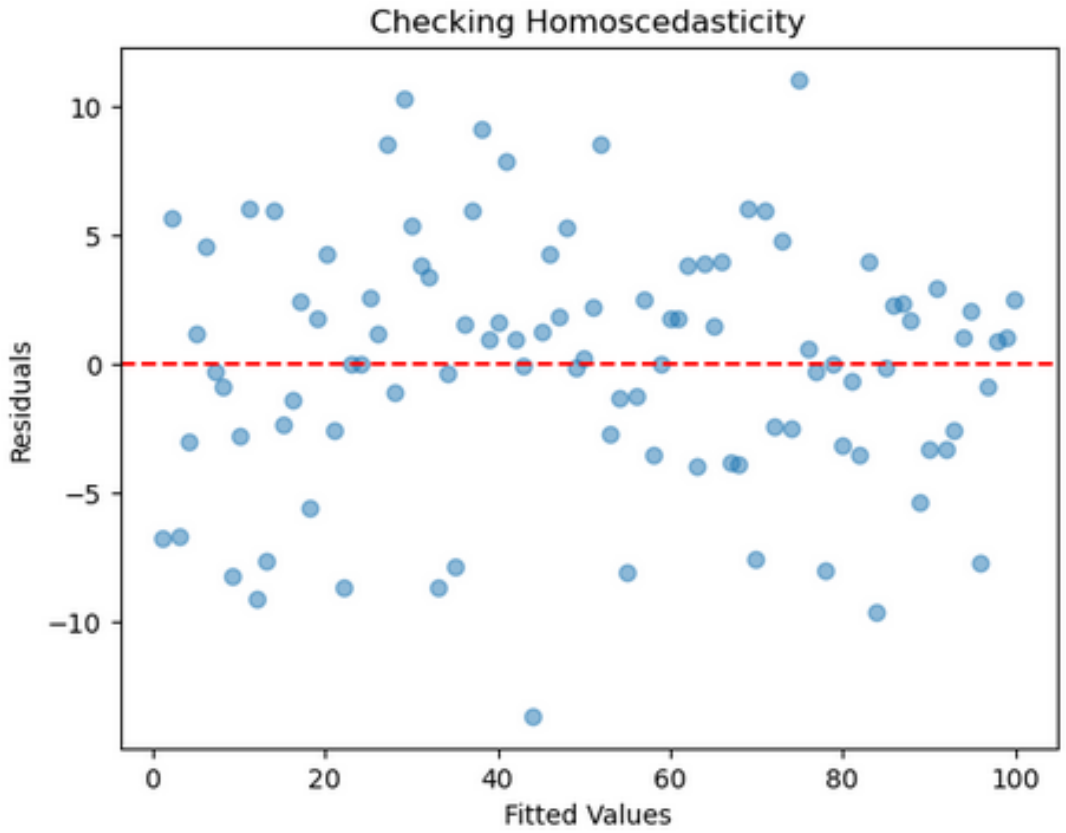
3. Homoscedasticity (Constant Variance of Errors)

- Residuals should have constant variance across all levels of the independent variables.
- **Verification:**
  - **Residual plot**: Plot residuals against fitted values.
  - **Breusch-Pagan test**: A statistical test for heteroscedasticity.
  - $P < 0.05$ problem of homoscedasticity

3. Homoscedasticity

```
import statsmodels.stats.diagnostic as smd

# Residual plot
plt.scatter(model.fittedvalues, model.resid, alpha=0.5)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel("Fitted Values")
plt.ylabel("Residuals")
plt.title("Checking Homoscedasticity")
plt.show()

# Breusch-Pagan Test
bp_test = smd.het_breuschpagan(model.resid,
model.model.exog)
print(f"Breusch-Pagan p-value: {bp_test[1]}")
```

3. Homoscedasticity



Breusch-Pagan p-value: 0.14590872049409248

4. No Autocorrelation of Errors

- Residuals should not be correlated (important in time series data).
- **Verification:**
  - **Durbin-Watson test**: Checks for first-order autocorrelation.
  - A Durbin-Watson statistic close to 2 suggests no autocorrelation.
  - Values < 1 or > 3 indicate strong autocorrelation.

4. No Autocorrelation of Errors

```
from statsmodels.stats.stattools import durbin_Watson
dw_stat = durbin_watson(model.resid)
print(f"Durbin-Watson Statistic: {dw_stat}")

Durbin-Watson Statistic: 2.118513424817343
```

5. Normality of Residuals

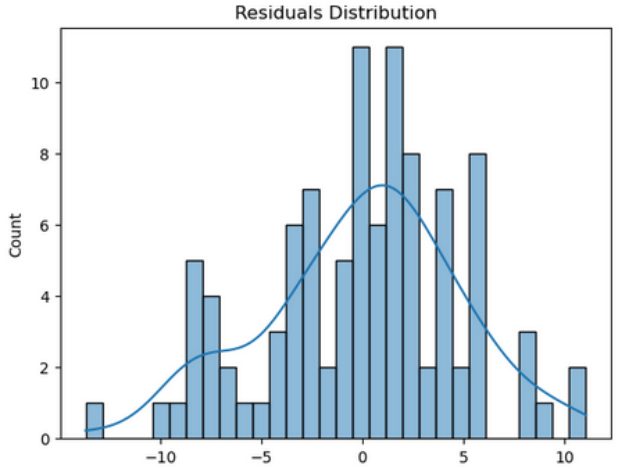
- Residuals should be normally distributed.
- **Verification:**
  - **Histogram & KDE Plot**: Visualize residual distribution.
  - **Shapiro-Wilk test**: Formal statistical test for normality.
  - **Q-Q Plot**: Check residuals against a normal distribution.
  - A normal residual distribution supports OLS assumptions.
  - A p-value < 0.05 in the Shapiro-Wilk test suggests non-normality.

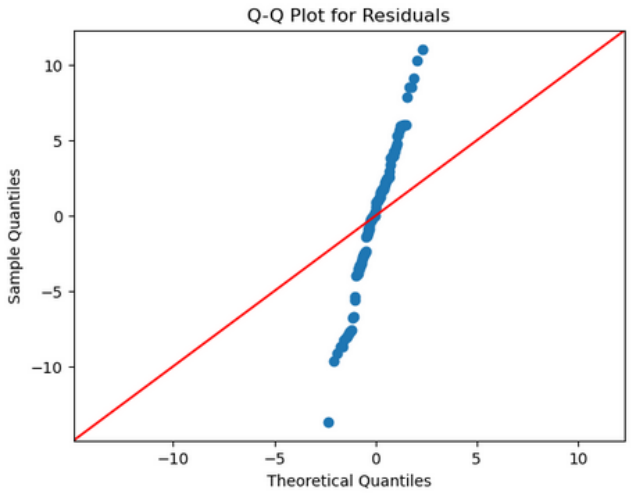
5. Normality of Residuals

```
import scipy.stats as stats

# Histogram and KDE
sns.histplot(model.resid, kde=True, bins=30)
plt.title("Residuals Distribution")
plt.show()

# Q-Q Plot
sm.qqplot(model.resid, line="45")
plt.title("Q-Q Plot for Residuals")
plt.show()

# Shapiro-Wilk Test
shapiro_test = stats.shapiro(model.resid)
print(f"Shapiro-Wilk p-value: {shapiro_test.pvalue}")
```

5. Normality of Residuals

- A p-value < 0.05 in the Shapiro-Wilk test suggests non-normality.
- In this case you do not reject the hypothesis of normality.





Shapiro-Wilk p-value: 0.22804526039392653

Statsmodels.summary

summary() function in **Statsmodels** provides important regression output, but it does **not** directly test OLS assumptions. However, it gives useful clues about potential violations.

What Information Does summary() Provide?

```
import statsmodels.api as sm
model = sm.OLS(y, X).fit()
print(model.summary())
```

Output:
1.R-squared & Adjusted R-squared – Indicate model fit but do not test assumptions.
2.F-statistic & Prob (F-statistic) – Tests if at least one predictor is significant.
3.Coefficients & p-values – Show predictor significance but do not check multicollinearity.
4.Standard Errors – Can be affected by heteroscedasticity.
5.Durbin-Watson Statistic – Helps detect **autocorrelation** (should be close to 2).
6.Omnibus & Jarque-Bera Tests – Check for **normality of residuals** (should have a high p-value).

What Assumptions Does summary() Not Test Directly?
- **Homoscedasticity (constant variance of residuals)** → Use **Breusch-Pagan** or **White test**.
- **Multicollinearity** → Check **Variance Inflation Factor (VIF)**.
- **Independence of Errors** → Use **Durbin-Watson test** (partially included in summary).
- **Linear Relationship** → Use **residual plots** or **Ramsey RESET test**.