statsmodels

The statsmodels essentials, covering classical statistics with readable summaries across regression, GLMs, time series, and tests.

Fit a Model with the Formula API

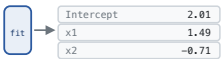
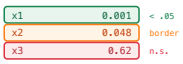
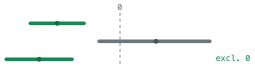
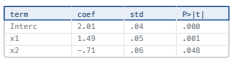
A formula string plus a DataFrame, then .fit().

PURPOSE	COMMAND	RESULT
Fit ordinary least squares.	`model = smf.ols("y ~ x1 + x2", df).fit()`	
Add a categorical factor.	`smf.ols("y ~ x1 +C(grp)", df).fit()`	
Add an interaction term.	`smf.ols("y ~ x1* x2", df).fit()`	
Transform a variable inline.	`smf.ols("y ~ np.log(x1) + I(x2**2)", df).fit()`	
Robust standard errors.	`smf.ols("y ~ x1", df).fit(cov_type="HC3")`	
Arrays instead of a formula.	`sm.OLS(y,sm.add_constant(X)).fit()`	

`smf.ols("y ~ x", df).fit()` turns a formula + DataFrame into a fitted RegressionResults; the array API needs sm.add_constant

Read the Summary Table

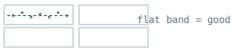
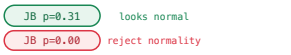
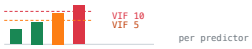
One .summary() gives coefficients, p-values, R-squared, AIC.

PURPOSE	COMMAND	RESULT
Print the full report.	`print(model.summary())`	
Coefficient estimates.	`model.params`	
Significance (p-values).	`model.pvalues`	
95% confidence intervals.	`model.conf_int()`	
Goodness of fit.	`model.rsquared, .rsquared_adj, model.aic, model.bic`	
Tidy DataFrame of results.	`model.summary2().tables[1]`	

`summary()` prints the whole report; params, pvalues, conf_int, rsquared, aic pull each piece; summary2().tables[1] is a tidy DataFrame

Residual Diagnostics

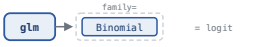
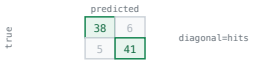
Check the assumptions behind the p-values.

PURPOSE	COMMAND	RESULT
Residuals vs fitted plot.	`sm.graphics.plot_regress_exog(m, "x1")`	
Normal Q-Q plot of residuals.	`sm.qqplot(m.resid, line="45")`	
Normality test.	`from statsmodels.stats.stattools import jarque_bera; jarque_bera(m.resid)`	
Heteroskedasticity test.	`from statsmodels.stats.diagnostic import het_breuschpagan(m.resid, m.model.exog)`	
Multicollinearity (VIF).	`from statsmodels.stats.outliers_influence import variance_inflation_factor(X, i)`	
Influential points.	`infl = m.get_influence()` `infl.cooks_distance[0]`	

A p-value is only as trustworthy as its assumptions: check residual normality, constant variance, multicollinearity, and influence

Logistic Regression & GLM

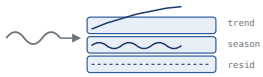
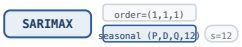
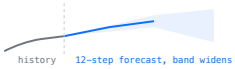
Non-normal outcomes: binary, counts, rates.

PURPOSE	COMMAND	RESULT
Logistic regression.	`logit = smf.logit( "y ~ x1 + x2", data=df).fit()`	
Coefficients as odds ratios.	`np.exp(logit.params)`	
Same model as a GLM.	`smf.glm("y ~ x1 + x2", data=df, family=sm.families.Binomial()).fit()`	
Count data (Poisson).	`smf.glm("count ~ x1", data=df, family=sm.families.Poisson()).fit()`	
Classification table.	`logit.pred_table()`	
Marginal effects.	`logit.get_margeff().summary()`	

smf.logit is a GLM with a Binomial family; swap family= for Poisson counts; exp(params) gives odds ratios, get_margeff() probabilities

Time series

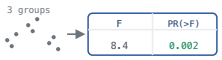
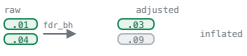
Decompose, fit ARIMA / SARIMAX, forecast.

PURPOSE	COMMAND	RESULT
Split trend / season / residual.	`from statsmodels.tsa.seasonal import seasonal_decompose` `seasonal_decompose(y, model="additive", period=12)`	
Robust decomposition (STL).	`from statsmodels.tsa.seasonal import STL` `STL(y, period=12).fit()`	
Test for stationarity.	`from statsmodels.tsa.stattools import adfuller` `adfuller(y)`	
Fit ARIMA.	`from statsmodels.tsa.arima.model import ARIMA` `ARIMA(y, order=(1,1,1)).fit()`	
Seasonal model (SARIMAX).	`from statsmodels.tsa.statespace.sarimax import SARIMAX` `SARIMAX(y, order=(1,1,1), seasonal_order=(1,1,0,12)).fit()`	
Forecast with intervals.	`fc = model.get_forecast(steps=12)`	

decompose to see structure, adfuller to check stationarity, ARIMA / SARIMAX to fit, get_forecast for intervals that fan out

Hypothesis Tests

t-tests, ANOVA, proportions, and tests on coefficients.

PURPOSE	COMMAND	RESULT
Two-sample t-test.	`ttest_ind(a, b)`	
Two-proportion z-test.	`proportions_ztest(` `[40, 30], [100, 100])`	
One-way ANOVA from a model.	`anova_lm(model, typ=2)`	
Test a coefficient restriction.	`model.f_test("x1 = 0")`	
Correct for many comparisons.	`multipletests(pvals, method="fdr_bh")`	
Post-hoc group comparisons.	`pairwise_tukeyhsd(y, groups)`	

statsmodels.stats carries the classical tests scipy and sklearn scatter or skip: t-test, z-test, ANOVA, F-test, FDR, Tukey

Predict with Confidence Intervals

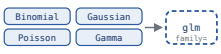
Point predictions plus mean and observation intervals.

PURPOSE	COMMAND	RESULT
Point predictions on new data.	`model.predict(new_df)`	
Prediction with uncertainty.	`pred = model.get_prediction(new_df)`	
Mean vs observation intervals.	`pred.summary_frame(alpha=0.05)`	
Tighter or looser intervals.	`pred.summary_frame(alpha=0.10)`	
In-sample fitted values.	`model.fittedvalues`	
Predict from a GLM (response scale).	`glm.get_prediction(new_df).summary_frame()`	

predict() gives points; get_prediction(...).summary_frame() adds the narrow mean CI and the wider observation interval, with alpha setting the width

Which API door?

smf for formulas, sm for arrays and families.

PURPOSE	COMMAND	RESULT
Formula API (R-style).	`import statsmodels.formula.api as smf`	
Arrays API plus extras.	`import statsmodels.api as sm`	
Same model, two spellings.	`smf.ols("y ~ x", df)` `sm.OLS(y, sm.add_constant(X))`	
Pick a GLM family.	`sm.families.Binomial() · Poisson() …`	
Load a sample dataset.	`sm.datasets.get_rdataset("mtcars").data`	
Built-in diagnostic graphics.	`sm.graphics.influence_plot(model)`	

smf holds lowercase formula functions (ols, glm, logit); sm holds uppercase array classes (OLS, GLM) plus families, datasets, graphics