Command	Explanation	Notes
`smd.linear_reset()`	RESET test	statsmodels.formula.api as smf
`ss.jarque_bera()`	Jarque-Bera test	scipy.stats as ss
`smd.het_breuschpagan()`	Breusch-Pagan test	statsmodels.stats.diagnostic as smd
`ols.get_robustcov_results()`	robust standard errors	statsmodels.formula.api as smf
`smf.logit()`	logit regression	statsmodels.formula.api as smf
`smf.probit()`	probit regression	statsmodels.formula.api as smf
`reg.pred_table()`	confusion matrix for `reg`	statsmodels.formula.api as smf

Tests

I do a RESET test, a JB test for normality, a test for heteroskedasticity, and calculate robust standard errors.

```
# Read in data
import pandas as pd
df = pd.read_csv(r'https://www.wimivo.com/data.csv')

# RESET test for nonlinear terms
import statsmodels.formula.api as smf
import statsmodels.stats.diagnostic as smd
ols = smf.ols('var2 ~ var1', data=df).fit()
reset = smd.linear_reset(ols, power=3,
                         test_type='fitted',use_f='true')
print(reset)

# JB test for normality
import scipy.stats as ss
resid = ols.resid
JB = ss.jarque_bera(resid)
print(JB)

# BP test for heteroskedasticity
BPtest = smd.het_breuschpagan(ols.resid, ols.model.exog)
print(BPtest)

# Robust standard errors
olsR = ols.get_robustcov_results(cov_type='HC3')
print(olsR.summary())
```

Probability Models

The form for a logit or probit regression is the same as an ordinary regressor. A nicely formatted confusion matrix is a bit more work, though.

```
logit = smf.logit('p ~ x1 + x2 + x3', data=df).fit()

logitPred = pd.DataFrame(logit.pred_table())
logitPred.columns = ['Predicted 0','Predicted 1']
logitPred = logitPred.rename(index={0: "Actual 0", 1: "Actual 1"})
print(logitPred)
```