Python Analytics
Quick Start

Eleven lessons translated from the supplied R Shiny app
Prepared for Sara Brumbaugh / Ceres Analytics • October 6, 2026

1. Extract the entire ZIP

Keep the scripts, **input_data.py**, **model_tools.py**, and **data** folder together. Each supervised lesson runs independently. Clustering uses three scripts in sequence. Read **START_HERE.txt** for the full teaching guide.

2. Install the libraries once

Use Python 3.10 or later. Open a terminal in the extracted **Python_Analytics_Teaching** folder:

```
python -m pip install -r requirements.txt
```

Windows may use **py**; macOS/Linux may use **python3**. Use the same Python interpreter for installation and execution. R is not required.

3. Run your first example

```
python linear_ols.py
```

In Jupyter, put the notebook in the extracted folder and run:

```
%run linear_ols.py
```

4. Inspect the results

Open **outputs/linear_ols**. You will find a model summary, coefficients, metrics, predictions, and PNG charts. Change the SETTINGS block at the top of a lesson script to try a different Y, set of Xs, or holdout count.

The **example_results** folder contains the default runs for all lessons. Re-running a lesson replaces its own outputs. Your input data files are unchanged.

Choose a Lesson

Script	Analysis / supplied example
linear_ols.py	Ordinary least squares / quarterly sales
linear_stepwise.py	Bidirectional F-test selection / quarterly sales
linear_all_subsets.py	Every subset; AIC, BIC, adjusted R^2 / quarterly sales
linear_lasso.py	Lasso; minimum-error or one-SE lambda / quarterly sales
cluster_ward.py	Step 1: Ward hierarchy / wine customers
cluster_evaluate.py	Step 2: compare numbers of clusters
cluster_assign.py	Step 3: assign selected cluster memberships
logistic_regression.py	Binomial logistic / Challenger launches
logistic_stepwise.py	Bidirectional likelihood-ratio tests / Challenger launches
tree_classification.py	Binary classification tree / telecom churn
tree_regression.py	Regression tree / quarterly sales

Clustering: run these in order

```
python cluster_ward.py
python cluster_evaluate.py
python cluster_assign.py
```

Set **N_CLUSTERS** in the third script after reviewing the evaluation. All three must share **WORKFLOW_FOLDER**. The default is two clusters, like Shiny; the wine example has its highest silhouette score at three. Re-run step 1 if you change its source data or variables.

Use your own CSV or tab-delimited file

```
DATA_FILE = 'data/houses.csv'
DELIMITER = None
Y_COLUMN = 'Sale price'
X_COLUMNS = ['Area', 'Age', 'Neighborhood']
CATEGORICAL_COLUMNS = ['Neighborhood']
SORT_BY = None
HOLDOUT_COUNT = 0
```

The first row must contain unique headings. Use those headings exactly, including spaces. Text Xs are automatically categorical; list numeric category codes explicitly. Do not select an ID or Y as an X. Relative filenames are resolved beside the script.

Data and Interpretation

Included input file	Origin / observations
quarterlySales.csv	Supplied CSV; 48 rows: 43 observed sales quarters and 5 future quarters
churn.csv	Supplied CSV; 3,333 telecom customers; original headings retained
clusterData_forWine.txt	Supplied tab-delimited file; 32 wine customers
challengerLaunch.csv	Converted from the unique RDS; 23 launches

Duplicate RDS files were checked against the text data and omitted. **tempData.RData** contains saved functions and a random seed, not another table. Student scripts read text files and require no R-data conversion tools.

Holdouts and missing observations

Regression: HOLDOUT_COUNT reserves the last N observed Y rows after sorting. Missing future Y values still receive forecasts when Xs are complete. For example, HOLDOUT_COUNT=4 yields 39 training rows, four holdouts and five forecasts for the supplied sales file.

Classification: tree HOLDOUT_PERCENT=20 means 20%; logistic TEST_FRACTION=0.2 means 20%. Both default to zero. Training metrics describe fit; holdout metrics assess performance on unused observations.

Rows with a missing selected X cannot be estimated or predicted, but remain in output tables. Missing unselected columns do not affect analysis. Cluster membership is blank for omitted rows. Unseen holdout categories are announced and left unscored.

Details to explain when comparing with Shiny

• Stepwise uses the app's SLE=0.15 and SLS=0.05. These can cause add/remove cycles; a guard explains and stops them. A conventional alternative is SLE=0.05, SLS=0.10.
• Ward uses the mean absolute deviation from the mean, matching agnes standardization.
• Lasso offers both lambda choices and an optional OLS refit. Its grid, folds and numerical results can differ from glmnet.
• Python CCP_ALPHA is **not numerically equivalent** to R rpart's CP. Tree splits and missing-data handling also differ.
• Logistic and classification-tree scripts use binary responses, matching the app's probability/ROC workflow.

Validation

All eleven default examples ran successfully. Additional checks covered holdouts, forecasts, categorical selection, both Lasso prediction modes, forward CV, file parsing and missing-row cluster alignment. Exact live R-versus-Python numerical parity was not tested. See **VALIDATION.txt** and **START_HERE.txt** for details.