Regression Analysis Project

Statistical Computing Module

1. Overview

This document provides the complete technical specification for the LinearRegressionStatisticalView module, a Python implementation of a simple linear regression analysis framework. The module evaluates the following statistical hypothesis:

- H_0: The slope (β_1) is zero, indicating no linear relationship between the predictor and the response variable.
- H_1: The slope (β_1) is not zero, indicating a statistically significant linear relationship.`

The implementation computes model statistics and inference values and exposes them through a well-defined public interface.

2. Project Objective

The objective of this project is to construct a small, self-contained statistical analysis module for simple linear regression using two numeric columns, designated X and Y. All statistical quantities must be computed from first principles using the NumPy and SciPy libraries, ensuring reproducibility and transparency of results with the slope fitting calculations.

3. Class Definition

Create a Python source file (e.g., linear_regression_statistical_view.py) and implement the class as specified below.

3.1 Constructor

```
class LinearRegressionStatisticalView:
    def __init__(self, alpha: float = 0.05):
        ...
```

The constructor accepts a single optional parameter, alpha, which sets the significance level for hypothesis testing. The default value is 0.05, corresponding to a 5% level of significance.

4. Required Methods

The class must implement the following seven public methods. Method signatures, return types, and dictionary key names must be preserved exactly as specified in order to ensure compatibility with the automated test suite.

4.1 fit

```
def fit(self, X, Y):
    ...
```

Fits a simple linear regression model on the supplied data. X is a one-dimensional array-like object representing the predictor variable, and Y is a one-dimensional array-like object representing the response variable. Both arrays must be drawn from the same dataset and must be of equal length. This method must be called before any other method is invoked.

4.2 f_statistic

```
def f_statistic(self) -> float:
    ...
```

Returns the model F-statistic as a Python float. The F-statistic quantifies the ratio of explained variance to unexplained variance and is used to assess the overall significance of the regression model.

4.3 f_critical

```
def f_critical(self) -> float:
    ...
```

Returns the critical value of the F-distribution at the configured significance level alpha. The critical value is computed using:

- Numerator degrees of freedom: df_model
- Denominator degrees of freedom: df_resid

The F-critical value is obtained from scipy.stats and corresponds to the upper-tail critical region of the F-distribution.

4.4 beta_confidence_intervals

```
def beta_confidence_intervals(self) -> dict:
    ...
```

Returns a dictionary containing the confidence intervals for both regression coefficients:

```
{
    "beta_0": (lower_bound, upper_bound),
    "beta_1": (lower_bound, upper_bound)
}
```

The interval for beta_0 represents the confidence interval for the intercept, and the interval for beta_1 represents the confidence interval for the slope. Both intervals are computed at the significance level alpha.

4.5 degrees_of_freedom

```
def degrees_of_freedom(self) -> dict:
    ...
```

Returns a dictionary containing the degrees of freedom associated with the fitted model. For simple linear regression with n observations, the values are defined as follows:

```
{
    "df_model": 1,
    "df_resid": n - 2,
    "df_total": n - 1
}
```

4.6 r_squared

```
def r_squared(self) -> float:
    ...
```

Returns the coefficient of determination (R^2) as a Python float. R^2 measures the proportion of the total variance in Y that is explained by the linear relationship with X. Values range from 0 (no explanatory power) to 1 (perfect fit).

4.7 r

```
def r(self) -> float:
    ...
```

Returns the Pearson correlation coefficient (R) as a Python float. The sign of R is consistent with the direction of the slope of the fitted model: positive when the slope is positive, and negative when the slope is negative.

4.8 summary

```
def summary(self) -> dict:
    ...
```

Returns a single dictionary consolidating all primary statistical outputs of the model. The required structure is as follows:

```
{
    "f_critical": ...,
    "f_stat": ...,
    "beta_confidence_intervals": {
        "beta_0": (..., ...),
        "beta_1": (..., ...)
    },
    "degrees_of_freedom": {
        "df_model": ...,
        "df_resid": ...,
        "df_total": ...
    },
    "r_squareddef": ...,
    "r": ...
}
```

5. Comparison Function

In addition to the class, the source file must include a module-level function named exactly compare. This function enables direct comparison of the predictive utility of two independent variables against a common response variable.

5.1 Signature

```
def compare(df, x_col_1: str, x_col_2: str, y_col: str, alpha: float = 0.05) -> dict:
    ...
```

5.2 Behaviour

The function must perform the following steps in order:

1. Instantiate two separate LinearRegressionStatisticalView objects.
2. Fit Model A using x_col_1 as the predictor and y_col as the response.
3. Fit Model B using x_col_2 as the predictor and y_col as the response.
4. Retrieve the F-statistic from each fitted model.
5. Identify the predictor associated with the larger F-statistic as the best predictor.

5.3 Return Format

The function must return a dictionary with the following structure:

```
{
    "predictor_1": x_col_1,
    "predictor_2": x_col_2,
    "f_stat_1": ...,
    "f_stat_2": ...,
    "best_predictor": …
}
```

The value of best_predictor must be the column name (string) corresponding to the model with the higher F-statistic.

6. Grading Constraints

The following constraints are imposed to ensure compatibility with the automated unit test suite. Failure to comply with any of these requirements may result in loss of marks.The class must be named exactly LinearRegressionStatisticalView.

- All method names must match those specified in Section 4 exactly, including letter case.
- The top-level function must be named exactly compare.

- All methods must return Python-native numeric or dictionary values; printed output alone is insufficient.
- Dictionary keys in all return values must match the specified key names exactly.

7. Notebook Validation

Each student must submit a Jupyter Notebook demonstrating the correctness of their implementation. The notebook serves as a validation artefact and must include the following:

- Loading a dataset of the student's choice.
- Selection of a response column Y and at least two predictor columns.
- Fitting the LinearRegressionStatisticalView model on the chosen data.
- Display of the complete output of the summary() method.
- Execution of the compare() function and a written interpretation of which predictor is superior based on the F-statistic.

Note: Notebook outputs are provided for demonstration and validation purposes only. Final grading is determined exclusively by automated unit tests against the required class interface.

8. Recommended Libraries

The following Python libraries are recommended for this project:

- numpy — Numerical array operations and linear algebra.
- pandas — Data loading, manipulation, and column extraction.
- scipy.stats — F-critical value computation and confidence interval calculations.

Students are encouraged to rely solely on these libraries to maintain consistency with the expected computational approach. Use of high-level regression wrappers (e.g., statsmodels OLS, scikit-learn LinearRegression) is permitted for verification purposes only and must not be used as the primary implementation.