Python_library_report

Aditi (202318005) and Garvika (202318061)

▼ Overview

The name of the library is **dastats**. This project involved the creation of a Python library consisting of 1 package and 8 modules. Its has three versions 0.0.1, 0.0.2 and 0.0.3. The packages are uploaded to the central repositry

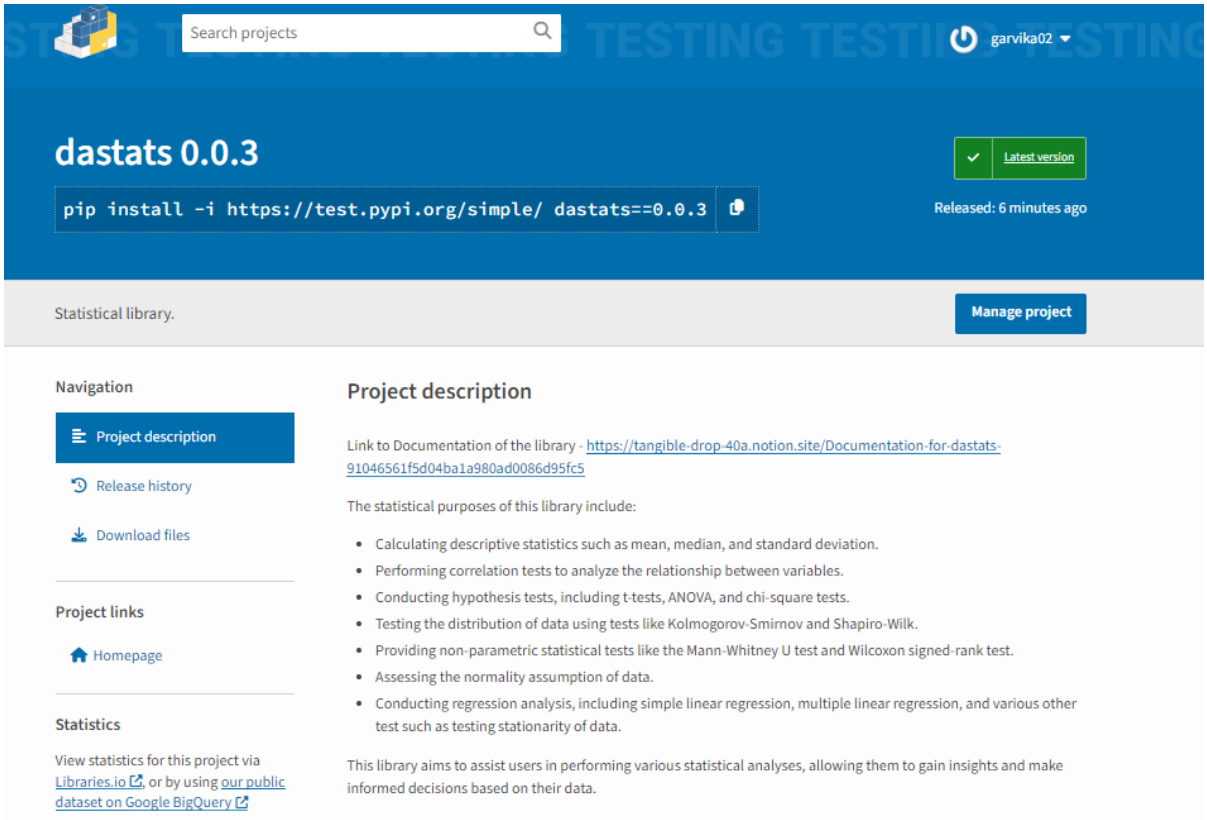
try and host PyPI for easier access and use.

Link to the PYPI test site containing the package information and file - https://test.pypi.org/project/dastats/0.0.3/

Link to project report -

Link to the official documentation - https://tangible-drop-40a.notion.site/Documentation-for-dastats-91046561f5d04ba1a980ad0086d95fc5

PyPI test website project view of the library -



▼ Project Deliverables

The project deliverables include:

- Python library with 1 package and 8 statistical modules
- Documentation for each module, including descriptions, usage examples, and function/class reference
- Test suite for each module, ensuring proper functionality and reliability.

▼ Objective

Creating a library is a great opportunity to learn and enhance our programming skills. While there are existing libraries that include all of these functionalities but they require separate installations. To address this problem, we have encapsulated all the necessary functions and tests in a single package to perform basic statistical analysis on any data.

Benefits of creating this library:

1. **Reusability**: By encapsulating statistical formulas and methods into a library, we make our code reusable. This means we can use the same set of functions across multiple projects without having to rewrite the statistical calculations each time.
2. **Modularity**: Breaking down statistical functionalities into separate modules allows for a modular design. We can import and use only the specific modules or functions we need, reducing complexity.
3. **Documentation**: Developing a library encourages the creation of documentation that explains how to use each statistical function, resulting in more clarity of concepts.
4. **Customization**: We can customize or extend our library to suit our specific needs. The open-source nature of Python encourages collaboration and contributions from others, leading to a more versatile and powerful toolset.
5. **Educational Purposes**: A library that includes various statistical methods can serve as an educational resource. It can help users understand how different statistical concepts are implemented and applied in practice.

By creating this library, we aim to provide a convenient and comprehensive tool for performing statistical analysis in Python. The modular structure and documentation make it easier to use and understand, while also allowing for customization and further development.

We hope that this library will be a valuable resource for both beginners and experienced users in the field of data analysis and statistics.

▼ Package and Module Structure

The library is organized into the following package and module structure:

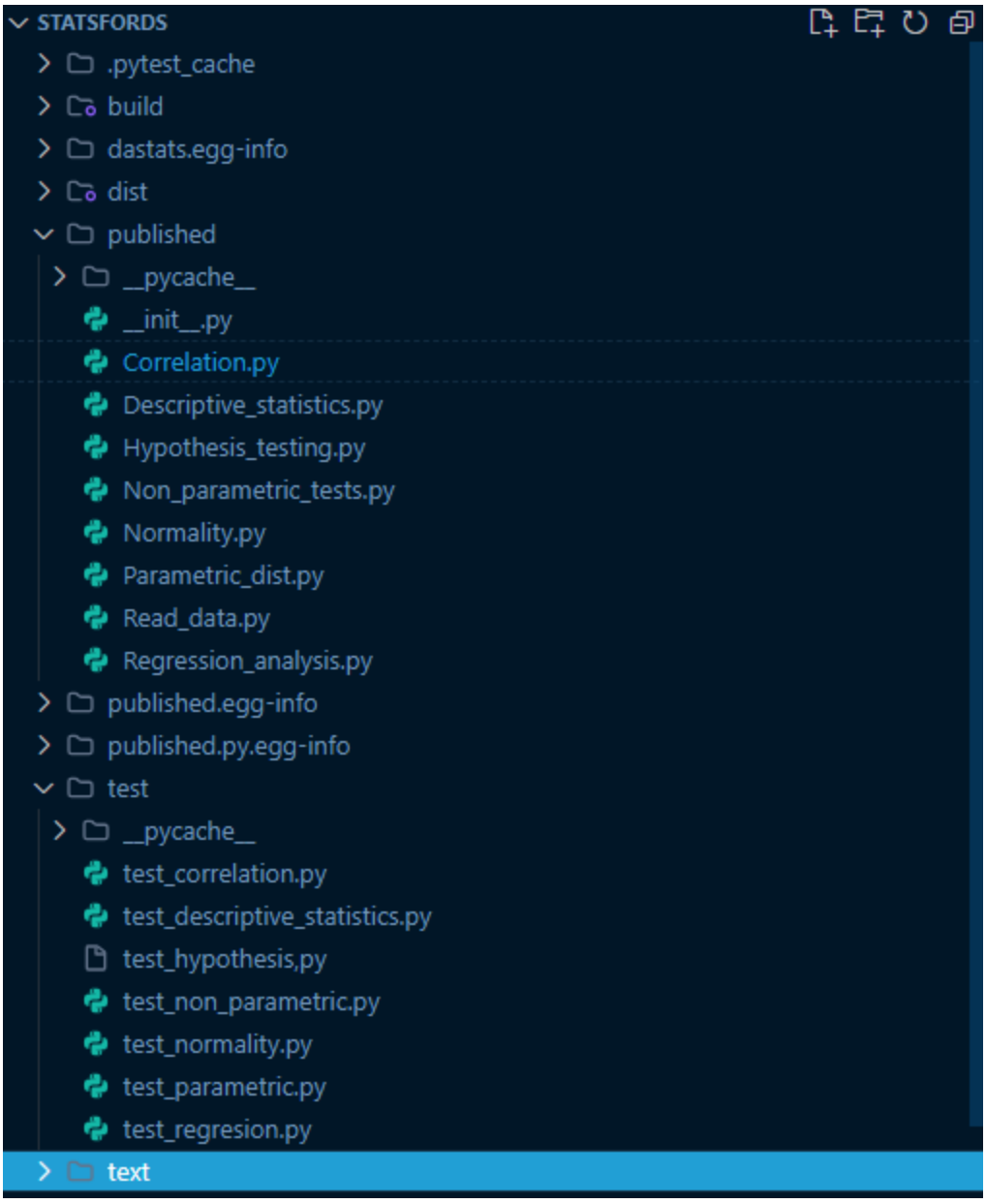
- Package: ['Published']
  - Module 1: [Read_file]
  - Module 2: [Descriptive_Statistics]
  - Module 3: [Correlation_tests]
  - Module 4: [Hypothesis_testing]

- Module 5: [Data_Distribution_Test]
- Module 6: [Non_Parametric_Test]
- Module 7: [Normality_Test]
- Module 8: [Regression_analysis]

Here is a brief overview of each module in the Python library:

1. Read_file: This module provides functions for reading and parsing data files. It allows users to load data into their Python environment for further analysis.
2. Descriptive_Statistics: This module contains functions for calculating various descriptive statistics, such as mean, median, standard deviation, and more. It helps users summarize and understand their data.
3. Correlation_tests: This module focuses on performing correlation tests between variables. It includes functions to calculate correlation coefficients and test their significance.
4. Hypothesis_testing: This module is dedicated to conducting hypothesis tests. It provides functions for performing t-tests, ANOVA, chi-square tests, and more.
5. Data_Distribution_Test: This module focuses on testing the distribution of data. It includes functions for conducting tests such as the Kolmogorov-Smirnov test and the Shapiro-Wilk test.
6. Non_Parametric_Test: This module provides non-parametric statistical tests. It includes functions for conducting tests like the Mann-Whitney U test and the Wilcoxon signed-rank test.
7. Normality_Test: This module focuses on testing the normality assumption of data.
8. Regression_analysis: This module is dedicated to regression analysis. It includes functions for performing simple linear regression, multiple linear regression, and logistic regression.

Please refer to the documentation link for more detailed information about each module, including descriptions and usage examples of the functions provided.



▼ Folder Structure and File Purposes

- `.pytest_cache`: This folder contains cached data from running pytest, a testing framework for Python. It helps improve test execution speed by storing previously computed results.
- `build`: This folder is generated during the build process and contains temporary files used in the creation of the Python library. It is typically used by build tools like setuptools.
- `dastats.egg-info`: This folder contains metadata about the installed package, such as its name, version, and dependencies. It is automatically generated when the library is installed using setuptools.=
- `dist`: This folder contains the distribution files of the library. These files are typically in the form of compressed archives (e.g., `.tar.gz` or `.zip`) and are used for distributing the library to other users.
- `published`: This folder contains the published version of the library. It includes the 8 modules that make up the library. These modules provide the core functionality and features of the library.
- `published.egg-info`: Similar to the `dastats.egg-info` folder, this folder contains metadata about the published version of the library.
- `test`: This folder contains test files for each module in the library. These test files are used to ensure that the library functions correctly and to catch any potential bugs or issues.
- `text`: This folder likely contains text or documentation files related to the library. It may include additional explanations, usage examples, or guidelines for users.
- `README`: This file typically contains a high-level overview and instructions for using the library. It serves as a starting point for users to understand the library's purpose and get started with its usage.
- `setup.py`: This file is used to define the configuration and dependencies required to build and distribute the library. It provides instructions to tools like setuptools on how to package and install the library.

▼ Tools Used

The following libraries and tools were used for setting up and publishing the Python library:

1. `setuptools`: This library is used for packaging and distributing Python projects. It provides the necessary setup functions and metadata for building and installing the library.
2. `pytest`: This is a testing framework for Python. It allows you to write and run tests to ensure the functionality and reliability of your library.
3. `wheel`: This tool is used for creating a distribution package in the form of an `.egg` file. It simplifies the process of packaging and distributing Python libraries.
4. `bdist`: This is a command provided by setuptools for creating binary distributions of the library. It packages the library into a format that can be easily installed on different platforms.
5. `sdist`: This is another command provided by setuptools for creating source distributions of the library. It packages the library's source code and resources into a compressed archive that can be distributed and installed on different platforms.
6. `twine`: This utility is used for publishing Python packages on PyPI. It simplifies the process of uploading your library to PyPI for others to install and use.

▼ Implementation of Tools

1. Code for `setuptools` in `setup.py`:

```
import setuptools
pip install --upgrade setuptools
pip install --upgrade build
python -m build
```

```python
from setuptools import setup
import os

setup(
    name="dastats",
    version="0.0.3",
    packages=["published"],
    install_requires=[],
    license="license.txt",  # Replace with your actual license
    url="https://tangible-drop-40a.notion.site/Documentation-for-dastats-91046561f5d04b
    author="A&G",
    author_email="202318065@daiict.ac.in",
    description="Statistical library.",
    long_description=open("readme.txt").read() if os.path.isfile("readme.txt") else "No
    long_description_content_type="text/x-rst",  # or "text/x-rst" if in reStructuredTe
    test_suite="nose.collector",
    tests_require=["nose"],
)
```

2. Code for `pytest` in test files:

```
import pytest
pytest -v
```

```
test/test_correlation.py::test_mean PASSED                                    [  4%]
test/test_correlation.py::test_rank PASSED                                    [  8%]
test/test_correlation.py::test_pearson_correlation PASSED                     [ 12%]
test/test_correlation.py::test_spearman_rank_correlation PASSED               [ 16%]
test/test_correlation.py::test_kendalls_rank_correlation PASSED               [ 20%]
test/test_descriptive_statistics.py::test_frequency_output PASSED             [ 25%]
test/test_descriptive_statistics.py::test_mean PASSED                         [ 29%]
test/test_descriptive_statistics.py::test_median PASSED                       [ 33%]
test/test_descriptive_statistics.py::test_variance PASSED                     [ 37%]
test/test_descriptive_statistics.py::test_standard_deviation PASSED           [ 41%]
test/test_descriptive_statistics.py::test_mode PASSED                         [ 45%]
test/test_descriptive_statistics.py::test_summary_statistics PASSED           [ 50%]
test/test_descriptive_statistics.py::test_quantiles PASSED                    [ 54%]
test/test_non_parametric.py::test_mann_whitney_u_test PASSED                  [ 58%]
test/test_non_parametric.py::test_wilcoxon_signed_rank_test PASSED            [ 62%]
test/test_non_parametric.py::test_friedman_test PASSED                        [ 66%]
test/test_normality.py::test_kolmogorov_smirnov_test_normal PASSED            [ 70%]
test/test_normality.py::test_kolmogorov_smirnov_test_non_normal PASSED        [ 75%]
test/test_normality.py::test_shapiro_wilk_test_normal PASSED                  [ 79%]
test/test_normality.py::test_shapiro_wilk_test_non_normal PASSED              [ 83%]
test/test_parametric.py::test_parametric_test PASSED                          [ 87%]
test/test_parametric.py::test_nonparametric_test PASSED                       [ 91%]
test/test_regresion.py::test_adf_test PASSED                                  [ 95%]
test/test_regresion.py::test_kpss_test PASSED                                 [100%]

============================================== 24 passed in 0.22s ==============================================
```

3. Code for `wheel` to create a distribution package:

```
"C:/Users/DSR/AppData/Local/Programs/Python/Python310/python.exe" setup.py install
python setup.py bdist_wheel
```

4. Code for `bdist` to create binary distributions:

```
python setup.py bdist
```

5. Code for `sdist` to create source distributions:

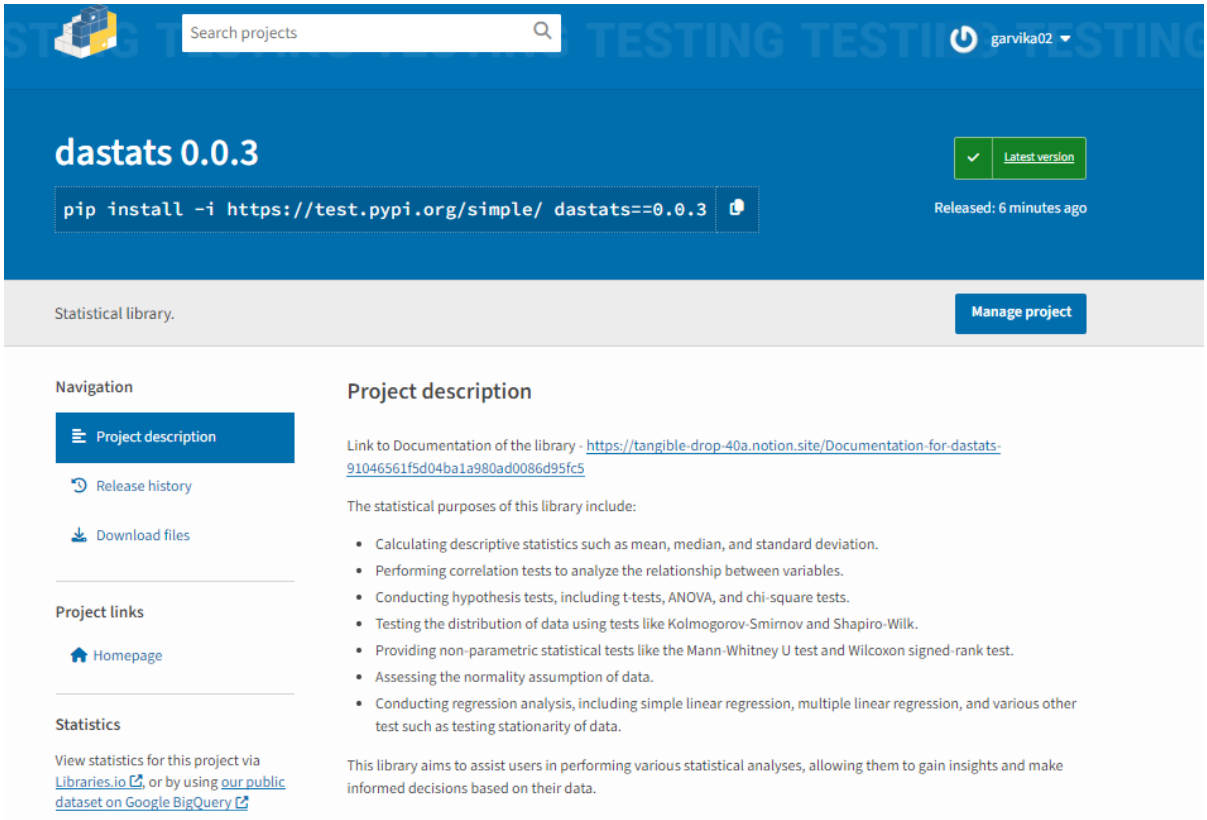
```
pip install .
python setup.py sdist
C:/Users/DSR/AppData/Local/Programs/Python/Python310/python.exe setup.py sdist
```

6. Code for `twine` to upload the library to PyPI test:

```
rm -r dist

python setup.py sdist bdist_wheel

twine upload --repository-url https://test.pypi.org/legacy/ --username __token__ --pas
sword pypi-AgENdGVzdC5weXBpLm9yZwIkM2M2ZWNmZjAtYjgxYy00MmUzLTlhYWItYmU3ZjQ4OTZjYzFmAAI
qWzMsIjMzNzQwYjAyLTNhOTctNDMyOS1hOTYzLTFhMjI5Mjg2YWMzMyJdAAAGIBGgdoiMOjlJxPaKidT7f2GWR
AbNfZgD7WeN6NuH-0yx dist/*
#or
twine upload -r testpypi dist/* #(pypi test)
#or
twine upload dist/* #(pypi)
```



These codes were used to configure and utilize the necessary tools for developing, testing, packaging, and distributing the Python library.

▼ Output / Result

We imported the library in google collab and tested it on california housing data set. Here are the results. All tests ran successfully.

Link to the google collab file - https://colab.research.google.com/drive/1fneuT44n40yBFjapHhuj3Y2JOzmwlRqG?usp=sharing

Module_1

Read_data

```
[ ] pip install -i https://test.pypi.org/simple/ dastats
```

```
Looking in indexes: https://test.pypi.org/simple/
Collecting dastats
  Downloading https://test-files.pythonhosted.org/packages/d5/cb/c6830579ad97a20f667990b8681cfd3f04d51d51ac4d08d82a06b9e9c936/dastats-0.0.1-py3-none-any.whl (11 kB)
Installing collected packages: dastats
Successfully installed dastats-0.0.1
```

```
[ ] from published.Read_data import read_csv, head, tail, view_complete_data
    data='/content/sample_data/california_housing_train.csv'
    col_names, rows = read_csv(data)
    # head(data)
    # tail(data)
    col_names
```

```
['longitude',
 'latitude',
 'housing_median_age',
 'total_rooms',
 'total_bedrooms',
 'population',
 'households',
 'median_income',
 'median_house_value']
```

Descriptive_Statistics

```
[ ] from published.Descriptive_statistics import *
    print(f"mean of {col_names[0]}",mean(rows[0]))
    print(f"median of {col_names[1]}",median(rows[1]))
    print(f"variance of {col_names[3]}", variance(rows[3]))
    print(f"standard deviation of {col_names[3]}", standard_deviation(rows[3]))
    print(f"mode of {col_names[4]}",mode)
    print(f"summary statistic for {col_names[1]}",summary_statistics(rows[1]))
    print(f"quantiles of {col_names[2]}",quantiles(rows[3]))
    frequency(rows[0])
```

```
mean of longitude 8357.597066666667
median of latitude 463.0
variance of total_rooms 593738736.9043549
standard deviation of total_rooms 24366.75474080187
mode of total_bedrooms <function mode at 0x78933c4440d0>
summary statistic for latitude {'Mean': 10131.527777777777, 'Median': 463.0, 'Mode': [-114.47, 34.4, 19.0, 7650.0, 1901.0, 1129.0, 463.0, 1.82, 80100.0], 'Variance': 6
quantiles of housing_median_age {'25%': 14.0, '50%': 226.0, '75%': 515.0}
x   frequency
-114.31   1
34.19   1
15.0   1
5612.0   1
1283.0   1
1015.0   1
472.0   1
1.4936   1
66900.0   1
```

```
[ ] import statistics
    statistics.variance(rows[3])
```

```
593738736.9043549
```

Module_3

Correlation_test

```
from published.Correlation import *
print(f"pearsons_corr_coeff for {col_names[0]} and {col_names[1]}:",pearson_correlation(rows[0],rows[1]))
print(f"spearman_rank_corr_coeff for {col_names[0]} and {col_names[1]}:",spearman_rank_correlation(rows[0],rows[1]))
print(f"kendall_rank_corr_coeff for {col_names[0]} and {col_names[1]}:",kendalls_rank_correlation(rows[0],rows[1]))
```

```
pearsons_corr_coeff for longitude and latitude: 0.9999213822170953
spearman_rank_corr_coeff for longitude and latitude: 1.0
kendall_rank_corr_coeff for longitude and latitude: 1.0
```

Module_4

Hypothesis_testing

```
[ ] from published.Hypothesis_testing import *
    print(f"Independent t test statistic for {col_names[2]} and {col_names[3]}:",ttest_ind(rows[2],rows[3]))
    print("\n")
    print(f"Paired t test statistic for {col_names[2]} and {col_names[3]}:",ttest_rel(rows[2],rows[3]))
    print("\n")
    print(f"Annova test statistic for {col_names[2]} and {col_names[3]}:",annova(rows[2],rows[3]))
```

```
Independent t test statistic for housing_median_age and total_rooms: 0.09834965424626195


Paired t test statistic for housing_median_age and total_rooms: 0.0002132831965206932


Source of Variation     Sum of Squares  Degrees of Freedom      Mean Square     F
Between Groups  11261243668.31072       1       11261243668.31072       16.00967265449036
Within Groups   11254439899.02161       16      703402493.6888506       None
Total   22515683567.33233       17      None    None
Annova test statistic for housing_median_age and total_rooms: [['Source of Variation', 'Sum of Squares', 'Degrees of Freedom', 'Mean Square', 'F'], ['Between Groups',
```

Module_5

Test_Type

```
from published.Parametric_dist import *
parametric_test(rows[3])
nonparametric_test(rows[3])
```

```
'Nonparametric Test'
```

Module_6

Non_Parametric_tests

```
[ ] from published.Non_parametric_tests import *
    print(f"mann_whitney_u_test coeff for {col_names[0] and col_names[1]}: {mann_whitney_u_test(rows[0],rows[1])}")
    print(f"wilcoxon_signed_rank_test coeff for {col_names[1] and col_names[1]}: {wilcoxon_signed_rank_test(rows[0],rows[1])}")
    print(f"friedman_test coeff {col_names[0] and col_names[1]}:",{friedman_test(rows[0],rows[1])})
    help(friedman_test)
```

```
mann_whitney_u_test coeff for latitude: (45, 0.0003485751742130505)
wilcoxon_signed_rank_test coeff for latitude: (6, 0.3283832142406830 6)
friedman_test coeff latitude: {(495.0, 0.0)}
Help on function friedman_test in module published.Non_parametric_tests:

friedman_test(*samples)
    Perform the Friedman test.

    Parameters:
    - samples (list or array-like): Multiple samples for the Friedman test.

    Returns:
    - float: The test statistic.
    - float: The two-tailed p-value.
```

Module 8

Regression_analysis

```
[ ] from published.Regression_analysis import *
```

```
[ ] data='/content/sample_data/california_housing_train.csv'
```

```
[ ] X = data[4]
    y = data[9]
```

```
[ ]


    intercept, slope = linear_regression_fit(rows[4], rows[5])

    print(f"linear regression coeff for {col_names[4]} and {col_names[5]}")
    print("Intercept:", intercept)
    print("Slope:", slope)
```

```
    linear regression coeff for total_bedrooms and population
    Intercept: -57.28290553802253
    Slope: 1.1305743131090151
```

```
[ ] kpss=kpss_test(rows[4])

    print(f"kpss test statistic for {col_names[4]}")
    print('result_statistic:',kpss)
```

```
    kpss test statistic for total_bedrooms
    result_statistic: (1.0, False)
```

```
[ ] adf=adf_test(rows[5])
    print(f"adf test statistic for {col_names[5]}")
    print('result_statistic:',adf)
```

```
    adf test statistic for population
    result_statistic: (2.0, False)
```

```
[ ] mse=calculate_mse(rows[4],rows[5])
    print(f"mse for {col_names[4]}")
    print(mse)

    rmse=calculate_rmse(rows[4],rows[5])
    print(f"rmse for {col_names[4]}")
    print(rmse)
```

```
    mse for total_bedrooms
    8029490.001854827
    rmse for total_bedrooms
    2833.635474413536
```

```
[ ] r_sq=calculate_r_squared(rows[6],rows[7])
    print(f"r_squared (coefficient) for {col_names[4]}")
    print(rmse)
```

```
    r_squared (coefficient) for total_bedrooms
    2833.635474413536
```