[Local Stress Prediction] User Manual

Table of Contents

---

1. Introduction

The process of calculating the local stress at the nozzle junction of a pressure vessel using the finite element method is highly complex. To alleviate the difficulty of analysis, a predictive software based on ABAQUS and ML has been developed. This software can quickly predict the local stress at the nozzle junction of a pressure vessel, with an accuracy level exceeding 0.999 and a mean square error of only 1.639. This method contrasts with traditional finite element analysis design methods as it bypasses many complex analysis steps. Furthermore, we provide the methods and tools needed for software development, which will assist users in developing predictive software for specific operating conditions and material parameters. It provides a reliable and convenient platform for rapid evaluation and optimization of pressure vessel design, while also serving as a reference for stress prediction under different operating conditions and materials.

2. Product Overview

The Local Stress Prediction package comprises an ABAQUS script and a corresponding plugin (Generate Dataset) for generating datasets, Python code for machine learning and model generation, and a comprehensive stress prediction software (Stress Prediction-ML). This means that users can refer to these publicly available files to create stress prediction software tailored to specific working conditions and materials.
One ABAQUS script and one plugin are publicly available and serve the purpose of generating datasets. This script or plugin can create nozzle models in batches based on user-provided parameters and subsequently calculating the stress within them. All the calculated data is saved in a .txt file, which can be transformed into datasets in various formats as per requirements. Modifying the script or plugin allows users to obtain datasets for different materials and working conditions, although it is recommended to limit modifications to material parameters and avoid

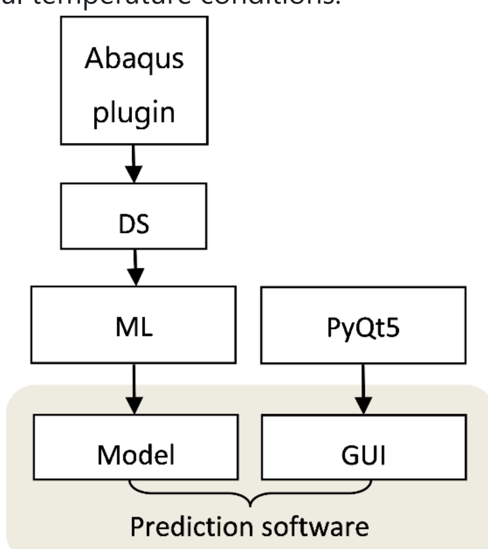
altering other parameters. An example of the dataset is also provided here, which has 2860 samples (date-1.scv).

After generating the dataset, machine learning can be performed using Python code to create personalized predictive models. This enables the generation of stress prediction models for any working conditions and materials. These models are designed to carry out predictive tasks, and users have the option to develop graphical user interfaces (GUIs) to provide fully-fledged software that can be used by non-technical individuals.

The complete stress prediction software is an illustrative example that can be directly utilized. It employs Q345 as the material and assumes normal temperature as the working condition. On top of this, it uses parameters such as pressure and geometry as features for machine learning to derive the software's predictions.

3. Installation/Setup Instructions

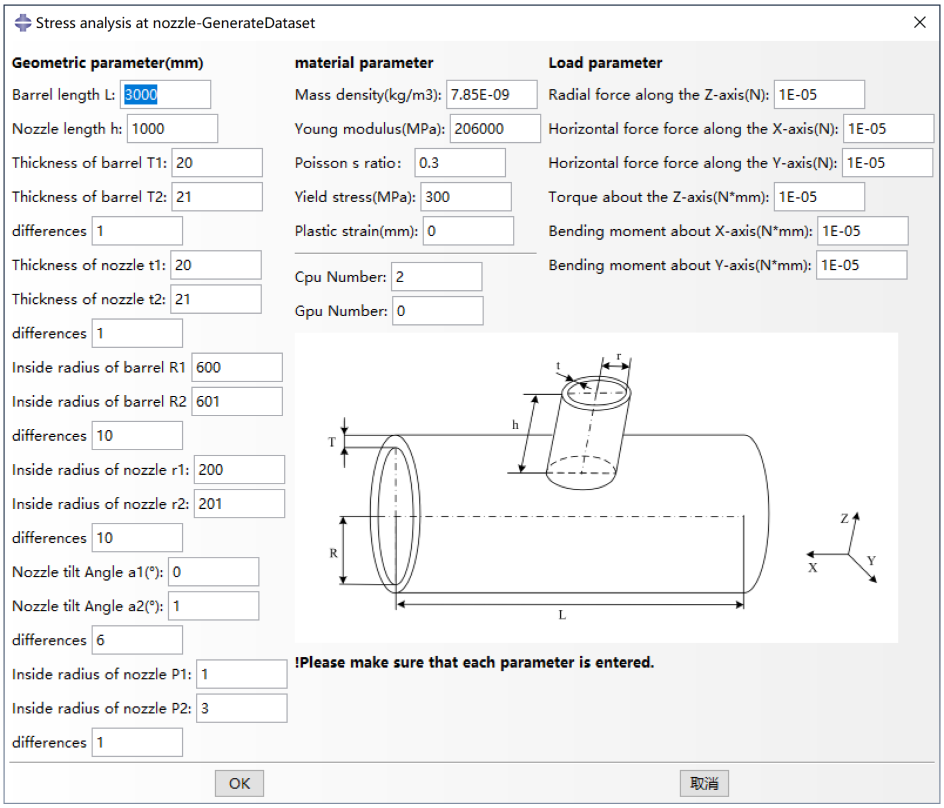
in general, the ABAQUS plugin or script is used for generating datasets (Data Set, DS), which are then employed for machine learning and model generation. The primary task of this model is stress prediction. For convenience, a graphical user interface (Graphical User Interface, GUI) can be created to develop it into a complete software application. Firstly, the ABAQUS plugin or script is used to select features for case calculations to generate datasets. Secondly, machine learning Python code is employed to choose an appropriate model for training and evaluate the trained model to ensure accurate stress predictions. Finally, the trained model is embedded in the software, and a GUI is created. We have already utilized this approach to develop a model for specific working conditions and materials, resulting in a mature stress prediction software (Stress_Prediction-ML). This software is suitable for predicting local stresses in Q345 material pressure vessels under normal temperature conditions.



GenerateDataset

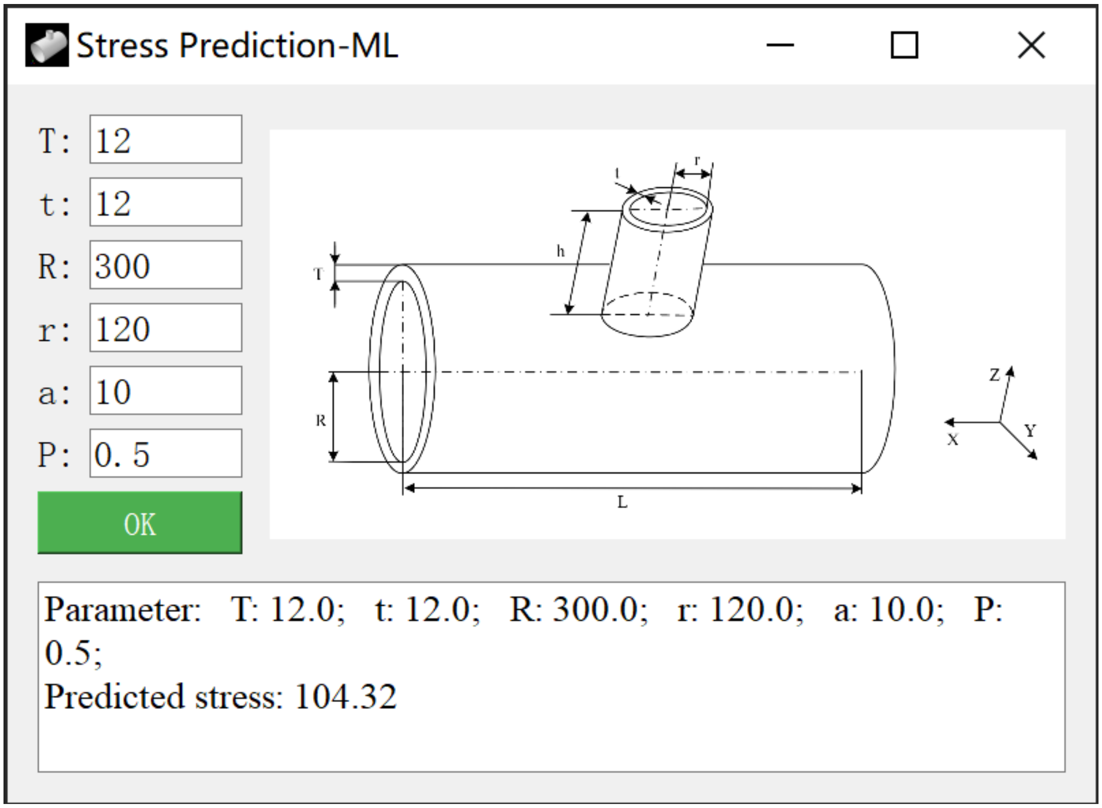
1. Copy the "GenerateDataset" folder to 'C:\Users\Username\abaqus_plugins'.

2. Open ABAQUS, and you will find "GenerateDataset" under the "Plug-ins" menu.
3. Open "GenerateDataset" and enter the parameters. Click "OK" to complete the batch analysis.
4. In the folder "C:\temp\Local_pipe_analysis", you will find the "MaxMises.txt" file.
5. Copy or convert the data into the desired format for your dataset.



Stress_Prediction-ML

1. Press Win+R to open the Run dialog box, then type "cmd" and press Enter. This will open the Command Prompt window.
2. In the Command Prompt window, enter the following commands (install one after another once the previous one is installed): pip install pyqt5 pip install pandas pip install scikit-learn pip install matplotlib pip install seaborn
3. Double-click on "Stress_Prediction-ML.exe" to run the program. Enter the corresponding geometric parameters and click OK to obtain the predicted values.
4. You can copy and save the results for each set of input parameters for further analysis.

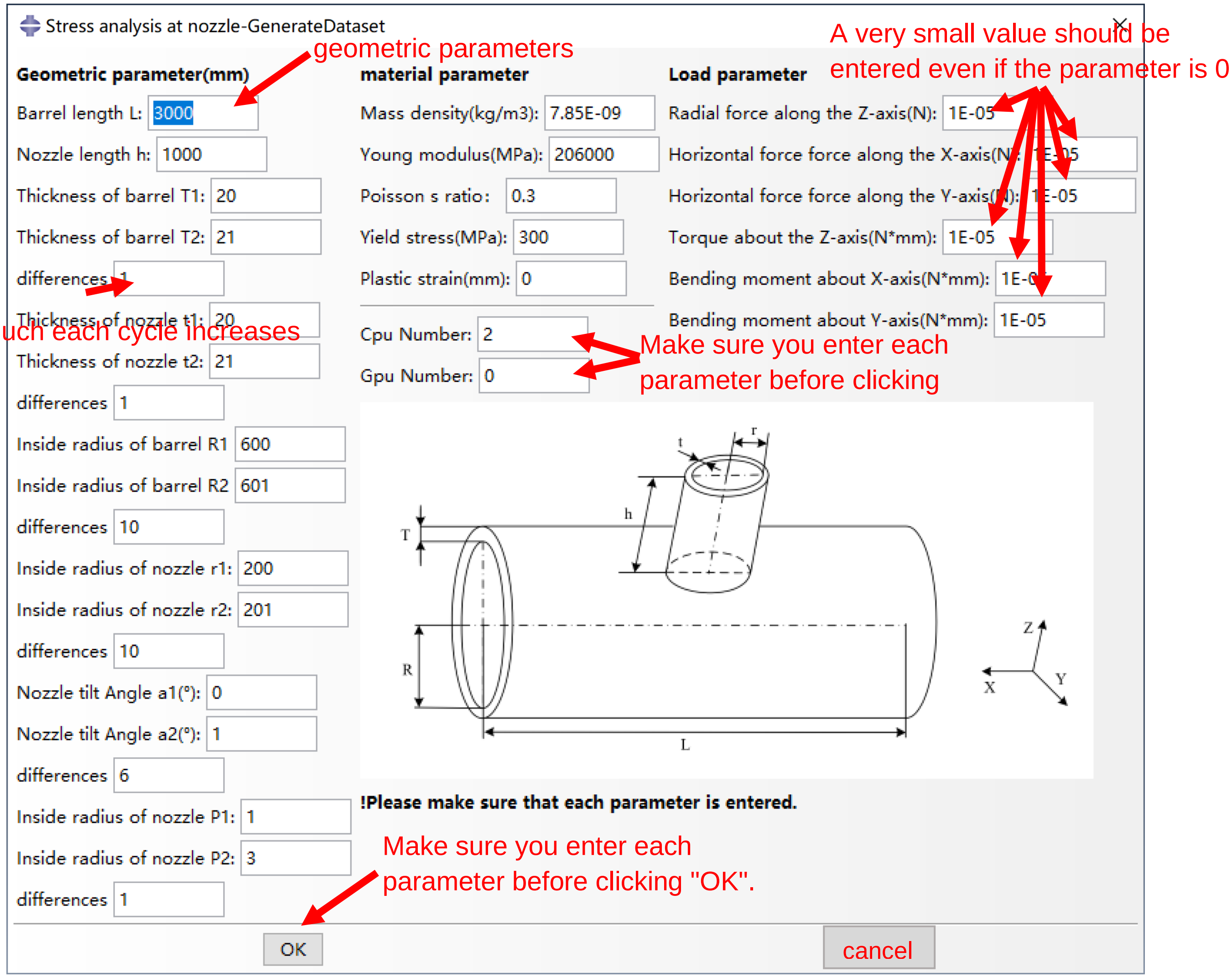

Stress Prediction-ML
T: 12
t: 12
R: 300
r: 120
a: 10
P: 0. 5
OK
t
r
h
T
R
L
Z
Y
X
Parameter: T: 12.0; t: 12.0; R: 300.0; r: 120.0; a: 10.0; P: 0.5;
Predicted stress: 104.32

ML code

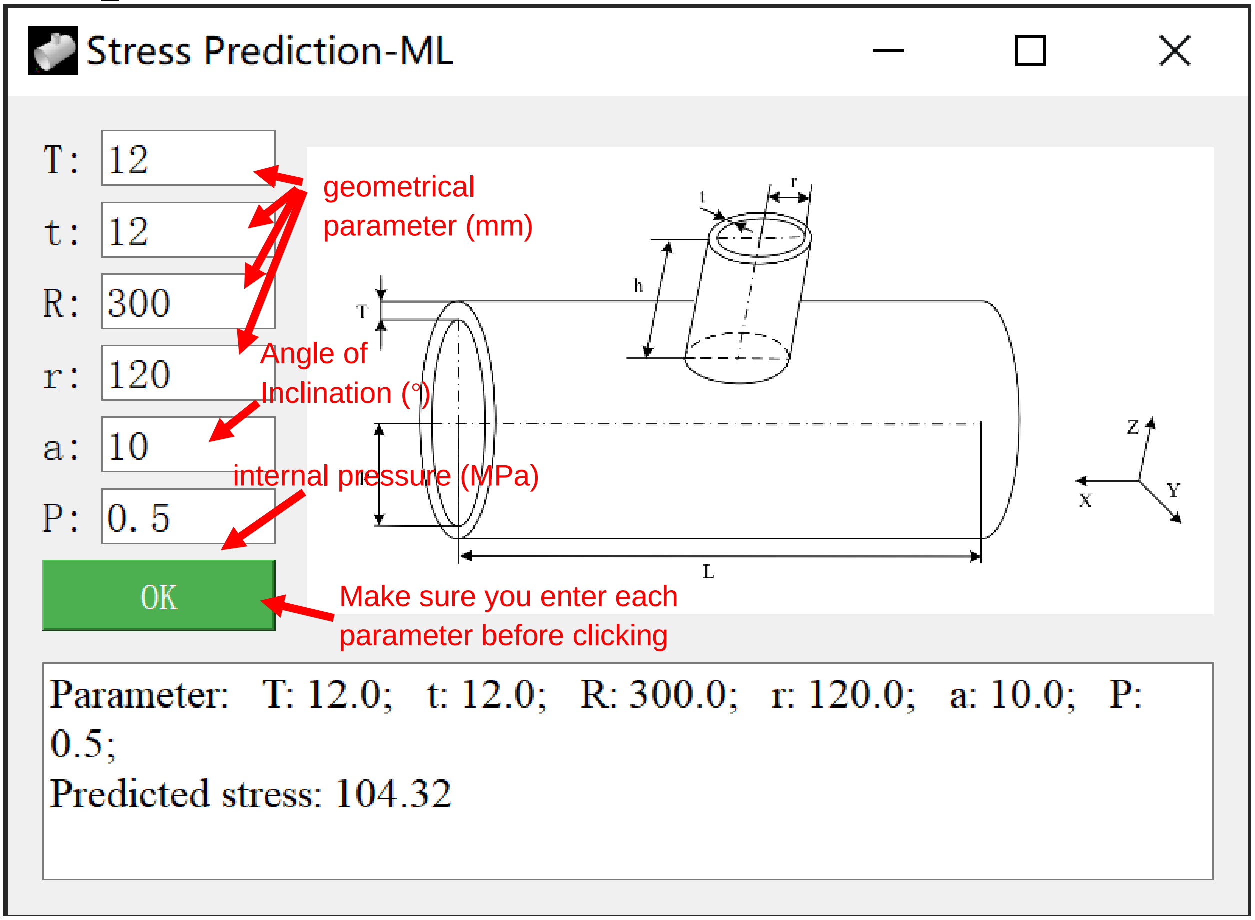
1. **Importing Libraries**: This code starts by importing the necessary Python libraries, including NumPy, Pandas, Matplotlib, Seaborn, and various modules from Scikit-Learn (sklearn). These libraries are essential for data processing, modeling, and evaluation.
2. **Reading Data**: The code reads a dataset from a CSV file named 'date-1.csv' using Pandas' read_csv function. It specifies the encoding as 'GBK' to handle potential character encoding issues in the dataset.
3. **Feature and Target Extraction**: It extracts the features (independent variables) and the target variable (dependent variable) from the dataset. Features are stored in the variable X, which includes columns 'T', 't', 'R', 'r', 'a', and 'P'. The target variable 'stress' is stored in y.
4. **Data Standardization**: The feature data (X) is standardized using Scikit-Learn's StandardScaler. Standardization ensures that all features have the same scale, which can be important for some machine learning algorithms.
5. **Train-Test Split**: The dataset is split into training and testing sets using Scikit-Learn's train_test_split function. 80% of the data is used for training (X_train and y_train), while 20% is reserved for testing (X_test and y_test).
6. **Gradient Boosting Regressor**: A Gradient Boosting Regressor model is created using Scikit-Learn's GradientBoostingRegressor class. This model will be trained to predict the 'stress' based on the input features.
7. **Hyperparameter Grid**: A grid of hyperparameters is defined in param_grid. These hyperparameters include the number of estimators (n_estimators), learning rate (learning_rate), maximum depth of the trees (max_depth), minimum samples required to split a node (min_samples_split), and minimum samples required in a leaf node (min_samples_leaf).
8. **Grid Search**: Grid search is performed using GridSearchCV to find the best combination of hyperparameters that yields the highest R-squared score on the training data.
9. **Best Hyperparameters**: The best hyperparameters from the grid search are printed to the console.
10. **Rebuilding Model**: The model is rebuilt using the best hyperparameters obtained from the grid search.
11. **Model Training**: The model is trained on the training data (X_train and y_train) using the fit method.
12. **Making Predictions**: Predictions are made on the test data (X_test) using the trained model, and the predicted values are stored in y_pred.
13. **Model Evaluation**: Several evaluation metrics are computed to assess the model's performance, including Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-squared (r2). These metrics help gauge how well the model fits the test data.
14. **Cross-Validation**: Cross-validation is performed using cross_val_score to obtain a more robust assessment of the model's performance. The mean R-squared score from cross-validation is printed.
15. **Model Saving**: The trained model is saved as a .pkl (Pickle) file named 'model-2.pkl' for future use.
16. **Data Visualization**: A scatter plot is created to visualize the predicted values (y_pred) against the true values (y_test). A diagonal dashed line represents a perfect prediction. This code essentially demonstrates the process of training and evaluating a Gradient Boosting Regressor model for predicting 'stress' based on input features. It also showcases the use of hyperparameter tuning and cross-validation to optimize and assess model performance.

4. Operating Instructions

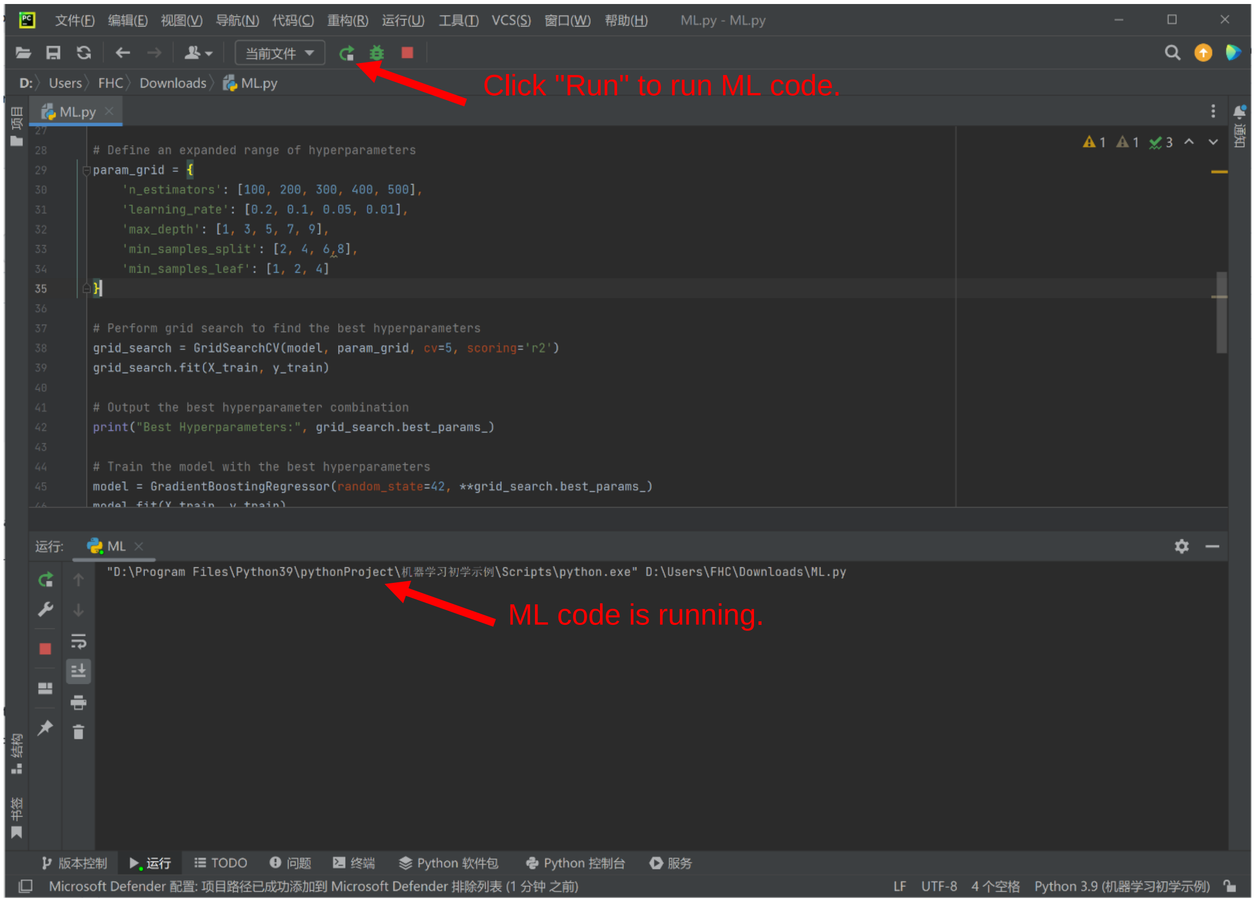
GenerateDataset



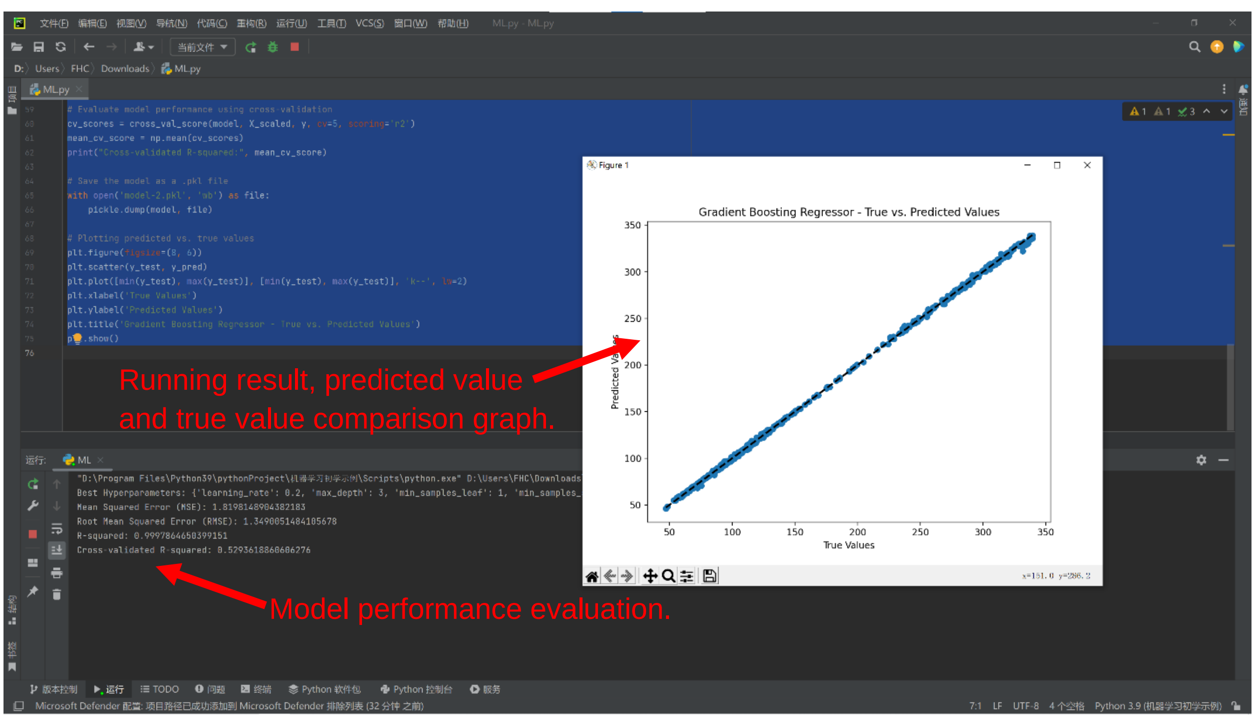
Stress_Prediction-ML



ML code



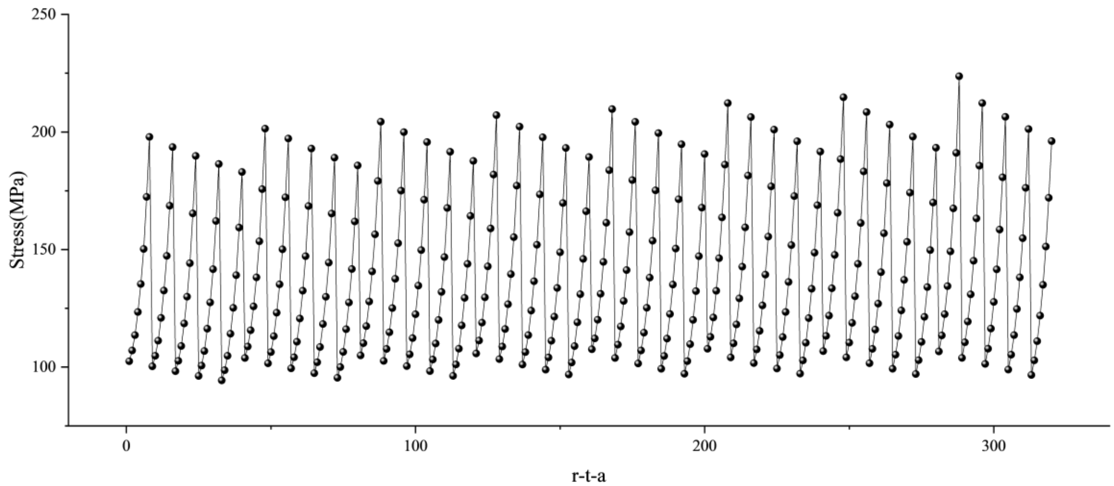
Click "Run" to run ML code.
ML.py
# Define an expanded range of hyperparameters
param_grid = {
'n_estimators': [100, 200, 300, 400, 500],
'learning_rate': [0.2, 0.1, 0.05, 0.01],
'max_depth': [1, 3, 5, 7, 9],
'min_samples_split': [2, 4, 6,8],
'min_samples_leaf': [1, 2, 4]
}
# Perform grid search to find the best hyperparameters
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='r2')
grid_search.fit(X_train, y_train)
# Output the best hyperparameter combination
print("Best Hyperparameters:", grid_search.best_params_)
# Train the model with the best hyperparameters
model = GradientBoostingRegressor(random_state=42, **grid_search.best_params_)
"D:\Program Files\Python39\pythonProject\机器学习初学示例\Scripts\python.exe" D:\Users\FHC\Downloads\ML.py
ML code is running.



Gradient Boosting Regressor - True vs. Predicted Values
True Values
Running result, predicted value
and true value comparison graph.
Best Hyperparameters: {'learning_rate': 0.2, 'max_depth': 3, 'min_samples_leaf': 1, 'min_samples_
Mean Squared Error (MSE): 1.8198148904382183
Root Mean Squared Error (RMSE): 1.3490051484105678
R-squared: 0.9997864658399151
Cross-validated R-squared: 0.5293618860606276
Model performance evaluation.

5. verification

To verify the universality of the plug-in, many models (320 models) were calculated using the batch analysis function, focusing on models that are not suitable for convergence, such as the wide-open model and the oblique intubation model. The maximum stress value automatically output with the plug-in will determine whether all calculations have been completed correctly. According to the numerical law of the output, the calculation of the plug-in is correct and universal. The horizontal axis represents a group of 8 tilt angles, which are arranged and combined under 5 nozzle thicknesses and 8 opening radii to cycle.



In order to further illustrate the accuracy of the plug-in, the existing calculation examples are calculated using the plug-in, and the results of the plug-in are compared with the original results. Example 1: Yunyun Yang [1] gives a calculation example of a pressure vessel and an orthogonal connection model. Example 2: Miao Ling [2] gives a calculation example of the connection area of a container. Calculation example 3: Chen Xingliang [3] gives a calculation example of the connection between the nozzle and the cylinder.

Parameter Type	Parameter name	example 1	example 2	example 3
Geometric parameters (mm)	Cylinder inner diameter R0	300	394	396
	Cylinder Thickness T	8	6	8
	Cylinder length L	1000	600	2000
	Nozzles inner diameter r0	90	38.96	112
	Nozzles thickness t	8	5.49	8
	Nozzles length h	400	100	250
Load parameter	Radial force along z axis (N)	none	3160	4200
	Horizontal force along the x-axis (N)	none	none	4000
	Horizontal force along the y axis (N)	none	none	4000
	Torque around z axis (N*mm)	none	5.48e5	3.0e6
	Bending moment around x-axis (N*mm)	none	none	3.4e6
	Bending moment around y-axis (N*mm)	none	none	3.4e6
	Internal pressure (MPa)	1.8	0.5	0.6
Material parameters	Mass density (mm^3)	none	none	7.85e-9
	Elastic modulus (MPa)	2.0e5	1.95e5	2.0e5
	Poisson's ratio	0.3	0.3	0.3
	Yield stress (MPa)	294	117	181

The stress distribution of the plug-in operation results is consistent with the original results. The maximum stress value has a small difference, and the relative error is less than 3%. This error is since the details in the original text are not explained, so there are some differences in preprocessing. It can be considered that the calculation results of the plug-in introduced in this paper are accurate.

	example 2	example 3	example 4
Plug-in calculation result (MPa)	204.94	106.50	164.35
Original result (MPa)	211.10	105.09	163.80
Relative error	2.92%	1.34%	0.34%

[1] Yang Y Y. Research on Partial Mechanical Characteristics and Stress Concentration Factors of Pressure Vessels with Special Opening Nozzle [D]. Northeastern University,2012. (In Chinese).
[2] Miao L. Finite Element Analysis of Local Stress on Vessel Nozzle Based on ABAQUS [J]. Mechanical Engineer,2015(06):83-84. (In Chinese).
[2] Chen X L. Benchmarking Analysis of Stress Engineering Algorithm for Pressure Vessel Openings [D]. Wuhan Institute of Technology,2017. (In Chinese)

6. Troubleshooting

If you have any questions, please leave a message, or email Fanhangchao@163.com.

7. Disclosures

Refer to license.txt for details.