MULTI-STAGE CONTINUOUS-FLOW MANUFACTURING PROCESS

https://www.kaggle.com/datasets/supergus/multistage-continuousflow-manufacturing-process

Arpit Srivastava	GitHub	arpitusrivastava9@gmail.com

1. About the Dataset:

Context

This data was taken from an actual production line near Detroit, Michigan. The goal is to predict certain properties of the line's output from the various input data. The line is a high-speed, continuous manufacturing process with parallel and series stages. Here we have an example of polymer extrusion setup.

Content

The data comes from one production run spanning several hours. Liveline Technologies has a large quantity of this type of data from multiple production lines in various locations.

Challenge

The data comes from a multi-stage continuous flow manufacturing process. In the first stage, Machines 1, 2, and 3 operate in parallel, and feed their outputs into a step that combines the flows. Output from the combiner is measured in 15 locations surrounding the outer surface of the material exiting the combiner.
Next, the output flows into a second stage, where Machines 4 and 5 process in series. After Machine 5, measurements are made again in the same 15 locations surrounding the outer surface of the material exiting Machine 5.

In this model we are trying to measure how accurately we are able to predict the values in the 15 locations using machine learning models for any given set of inputs.

The above problem is a supervised learning problem. Here we are trying to find correlation between the variables of inputs here we are implementing a **multi-stage machine learning pipeline.**
In the **first stage** feature selection is done using **backward elimination**. Then, it splits the data into training and testing datasets and applies a **Support Vector**

Machine (SVM) with a polynomial kernel for prediction. Finally, it evaluates the model by calculating its score on the test data-set.

The second stage of the code goes through a similar process of fitting an **OLS regression model, performing backward elimination to select significant features, and then training a SVM-Poly model on the selected features**. The score of the model on the test data is then printed.

Following the above stages, for each column in the output (target) data, we are performing backward elimination to select the most significant features from the input data. This is done by **fitting an OLS model, then eliminating the feature with the highest p-value if it's greater than 0.05.** Once all insignificant features are removed, we train an SVM with a Polynomial kernel using the selected features, and print the score of the model.

2. Description about the business scenario:

In the context of multi-stage continuous flow manufacturing processes, we are perpetually in search of superior predictive modeling techniques that can be integrated into real-time production settings. These models are utilized in numerous ways:

a) **Real-time Process Control Development:**

Utilizing these models within simulated environments enables us to design real-time process controllers. These controllers can constantly monitor and adjust the parameters of a manufacturing process across multiple stages, based on the models' predictions. For example, in a multi-stage chemical manufacturing process, predictive models can help adjust temperature, pressure, or flow rate to optimize production output and quality, reducing waste and improving efficiency.

b) **Anomaly Detection:**

By comparing model predictions with actual outcomes in real time, we can detect anomalies or outliers in the manufacturing process. This can indicate equipment malfunction, sub-optimal process conditions, or quality issues in early stages of the production, allowing for early interventions and thereby avoiding costly downtime or substandard products.

In essence, these predictive models can improve operational efficiency, product quality, and process safety in multi-stage continuous flow manufacturing environments, making them a critical tool for businesses.

3. Analysis of the data: (Refer to Annexure 01)

Here we have used python libraries in Google Colaboratory to analyse the data-set due to its large size and limitation of the Weka software.

1st Stage

% of the variance in the dependent variable that is predictable from the independent variables:

1. 6.62
2. 20.84
3. 2.71
4. 10.18
5. 16.34
6. 46.05
7. 2.21
8. 53.46
9. 64.22

The output is a series of scores for the Support Vector Regression (SVR) models for each output column from the first stage. These scores represent the percentage of the variance in the dependent variable that is predictable from the independent variables, also known as the R^2 score.

Here's a breakdown of the results:

1. For the 1st output column, the model achieved a score of 6.62%. This means that the model was able to explain about 6.62% of the variance in the dependent variable that is predictable from the independent variables.
2. For the 2nd output column, the model achieved a score of 20.84%, meaning it was able to explain about 20.84% of the predictable variance.
3. The process continues in the same way for all output columns.

The scores range from a low of 2.21% (very poor prediction) to a high of 64.22% (relatively good prediction). This means the predictive power of your models varies significantly across different output columns.

These scores provide an insight into the model's performance for each output column. Models with lower R^2 scores may need further optimization or a different approach, while models with higher scores show a better fit between the model and the data.

2nd Stage

% of the variance in the dependent variable that is predictable from the independent variables:
1. 41.13
2. 19.10
3. 63.57
4. 34.44
5. 1.04
6. 36.66
7. 50.13
8. 68.11
9. 64.80
10. 82.37
11. 84.72
12. 15.21
13. 71.91
14. 42.43

The output generated is a series of scores for the Support Vector Regression (SVR) models for each output column from the second stage. These scores represent the performance of your models in predicting the output on the test data. Specifically, these scores are the coefficient of determination R^2 of the prediction.

Here's a breakdown of the results:

1. For the 1st output column, the model achieved a score of 41.13%. This means that the model was able to explain about 41.13% of the variance in the dependent variable that is predictable from the independent variables.
2. For the 2nd output column, the model achieved a score of 19.10%, meaning it was able to explain about 19.10% of the predictable variance.
3. The process continues in the same way for all output columns.

The scores range from a low of 1.04% (very poor prediction) to a high of 84.72% (good prediction). This means the predictive power of your models varies significantly across different output columns.

These scores provide an insight into the model's performance for each output column. Models with lower R^2 scores may need further optimization or a different approach, while models with higher scores show a better fit between the model and the data.

These scores can help you assess which models are performing well and which ones need further tuning or perhaps a different approach.

4. Managerial recommendations:

Based on the analysis, here are several managerial recommendations:

1. Model Improvement: The predictive models vary significantly in terms of their ability to predict the output of both stages. Some models, particularly those with low R^2 scores (such as the 5th model in the second stage with a score of 1.04% and the 7th model in the first stage with a score of 2.21%), need further refinement or a different modeling approach to enhance their predictive capability.

2. Focus on Input Variables: It might be useful to revisit the input features used for predictions, especially for those models with lower scores. Understanding the relationship between these input variables and the outputs can help in feature selection and engineering, which in turn could improve the model performance.

3. Real-time Implementation: Implement the predictive models that demonstrate good predictive power into real-time manufacturing processes as soon as possible. This can

immediately assist in enhancing real-time process control and detecting anomalies, thereby improving production efficiency and quality, and minimizing downtime.

4. Staff Training: Ensure that the technical team understands the implications of these models and how to interpret the results. Training programs or workshops can be arranged for this purpose.

5. Continuous Monitoring and Optimization: Continuously monitor the models' performance over time and optimize as needed. Data and conditions change over time, and your models should adapt to these changes.

6. Further Research: Consider further research into other modeling techniques that might yield better predictive power. This could include exploring different machine learning algorithms, hyper-parameter tuning, or incorporating domain expertise into the model building process.

The goal is to increase the percentage of variance in the dependent variable that can be predicted from the independent variables. This would lead to more accurate predictions, more efficient manufacturing processes, and ultimately, a more profitable operation.

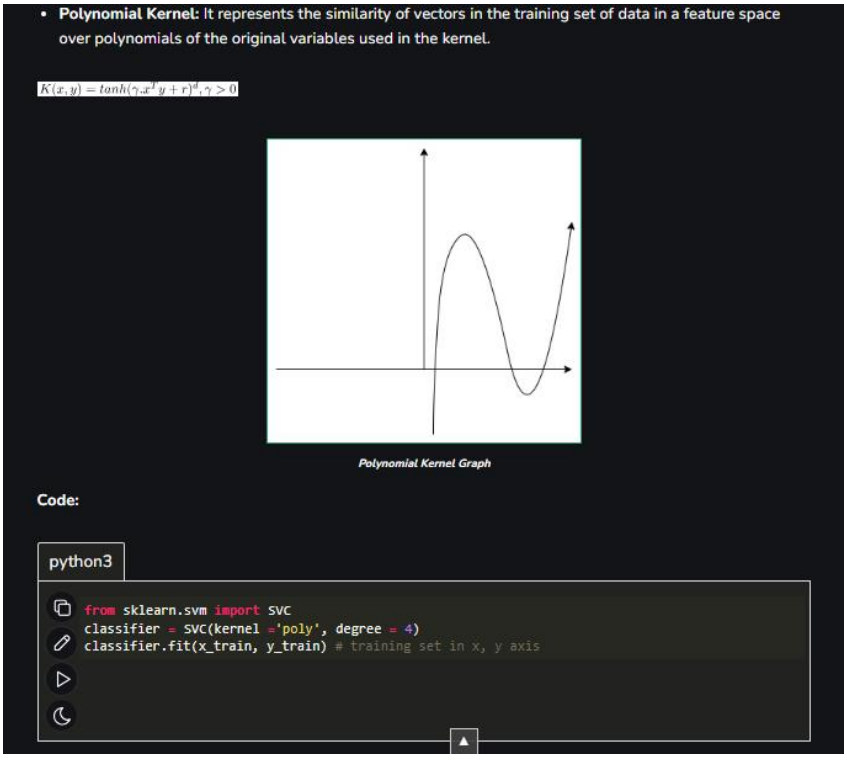
Annexure 01

The working is done in iPython notebook in the Google Colaboratory environment.

The steps involved are:

1. Install and import the required libraries
2. Upload the data to Google Colab server for processing
3. Exploratory Data Analysis
4. Check for null values
5. Splitting data stepwise
6. Anaylse each sub data-set
7. Check for correlation matrix within each data set
8. Output Prediction for Stage 01 and Stage 02

a) For Stage 01 : The code is implementing a machine learning pipeline where feature selection is done using backward elimination. Then, it splits the data into training and testing datasets and applies a Support Vector Machine (SVM) with a polynomial kernel for prediction. Finally, it evaluates the model by calculating its score on the test data-set. (ref: https://www.geeksforgeeks.org/major-kernel-functions-in-support-vector-machine-svm/)



b) For Stage 02 : This script is essentially a duplication of the first stage but using the second stage data X2 and Y2. The code goes through a similar process of fitting an OLS regression model, performing backward elimination to select significant

features, and then training a SVM-Poly model on the selected features. The score of the model on the test data is then printed.

c) Following the above stages, for each column in the output (target) data, we are performing backward elimination to select the most significant features from the input data. This is done by fitting an OLS model, then eliminating the feature with the highest p-value if it's greater than 0.05. Once all insignificant features are removed, we train an SVM with a Polynomial kernel using the selected features, and print the score of the model. (ref: https://www.middleprofessor.com/files/applied-biostatistics_bookdown/_book/model-fitting-and-model-fit-ols.html)