5D. PIPELINES FOR REGRESSION WITH JUPYTER NOTEBOOK

DR. ALVIN ANG

CONTENTS

INTRODUCTION

- This article is a continuation of Linear Regression (LR) / Multiple Regression (MR) / Polynomial Regression (PR) with Jupyter Notebook.
- LR / MR / PR already been described here
    - LR:
        i. https://www.alvinang.sg/s/How-to-Perform-Simple-Linear-Regression-using-Excel-Dr-Alvin-Ang-watermarked.pdf
        ii. https://www.alvinang.sg/s/Simple-Linear-Regression-with-Jupyter-Notebook-by-Dr-Alvin-Ang.pdf
    - MR:
        i. https://www.alvinang.sg/s/Multiple-Regression-MR-by-Dr-Alvin-Ang.pdf
        ii. https://www.alvinang.sg/s/Multiple-Regression-with-Jupyter-Notebook-by-Dr-Alvin-Ang.pdf
    - PR:
        i. https://www.alvinang.sg/s/Polynomial-Regression-with-Jupyter-Notebook-by-Dr-Alvin-Ang.pdf
- The purpose of LR / MR / PR is to find the best fit curve that runs through the dataset.
- The dataset is here:
    - https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DA0101EN/automobileEDA.csv
- Data Pipelines simplify the steps of processing the data.
- In simple terms, it means that we don't need to go through the long process just to predict the price.
- We will see in this article how to shortcut the process of LR / MR / PR to quickly obtain the best fit and immediately jump to a prediction.

***This applies to all Simple Linear Regression (LR) / Multiple Linear Regression (MR) / Polynomial Regression (PR)**	
Long Process **(Without Pipeline)**	**Shortcut** **(With Pipeline)**
Step 1: Load and Glance at the Dataset	Step 1: Load the Pipeline module together with the Standard Scaler module (the Standard Scaler module is to normalize the data for better fitting purposes).
Step 2: Visualize / Plot the Regression Model (Fitting and Training the Model)	Step 2: Choose which Regression model you wish to use (LR/ MR / PR).
Step 3: Generate a Regression Equation	Step 3: Fit the model using the Pipeline.
Step 4: Use a Residual Plot or Distribution Plot to inspect whether the Regression Model fits	Step 4: Predict the results!
Step 5: Use R2 or MSE as indicators to determine the accuracy of the fit.	

PART II: LEARN HOW TO USE PIPELINE TO AUTOMATICALLY FIND THE BEST FIT CURVE…AND MAKE SOME PREDICTIONS…

PRESUME WE CHOSE TO USE POLYNOMIAL REGRESSION (PR)…

STEP 1: DEFINE Z

- Code:
    - Z = df[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']]
- Comments:
    - Z has been defined to be the "price" affected by 4 independent variables, namely…'horsepower', 'curb-weight', 'engine-size' and 'highway-mpg'

STEP 2: LOAD THE PIPELINE + STANDARD SCALER MODULES

- We use the module Pipeline to create a pipeline.
- We also use StandardScaler as a step in our pipeline.
- Code:
    - from sklearn.pipeline import Pipeline
    - from sklearn.preprocessing import StandardScaler

STEP 3: CREATING THE PIPELINE

- We create the pipeline, by creating a list of tuples including the name of the model or estimator and its corresponding constructor.
- Code:
    - Input=[('scale',StandardScaler()), ('polynomial', PolynomialFeatures(include_bias=False)), ('model',LinearRegression())]

STEP 4: INPUT THE LIST AS AN ARGUMENT TO THE PIPELINE CONSTRUCTOR

- Code:

- pipe=Pipeline(Input)
- pipe

- Output:
  - Pipeline(memory=None, steps=[('scale', StandardScaler(copy=True, with_mean=True, with_std=True)), ('polynomial', PolynomialFeatures(degree=2, include_bias=False, interaction_only=False)), ('model', LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False))])

STEP 5: NORMALIZE THE DATA, TRANSFORM FOR FITTING, AND PREDICTION

- Code:
  - pipe.fit(Z,y)
  - ypipe=pipe.predict(Z)
  - ypipe[0:4]
- Output:
  - array([13102.74784201, 13102.74784201, 18225.54572197, 10390.29636555])
- Comments:
  - In short, what the array presented above is a prediction of price given that
    - We used the first 4 rows of data from the dataset.
    - We used only the variables contained within Z→ 'horsepower', 'curb-weight', 'engine-size' and 'highway-mpg' to predict it.
    - The regression technique used here was Polynomial Regression (PR).

PRESUME WE CHOSE TO USE MULTIPLE REGRESSION (MR)…

- Since the steps are similar to the above section, we will skip explaining the steps and jump straight to the code…

- Code:

    - Input=[('scale',StandardScaler()),('model',LinearRegression())]

    - pipe=Pipeline(Input)

    - pipe.fit(Z,y)

    - ypipe=pipe.predict(Z)

    - ypipe[0:10]

- Output:

    - array([13699.11161184, 13699.11161184, 19051.65470233, 10620.36193015, 15521.31420211, 13869.66673213, 15456.16196732, 15974.00907672, 17612.35917161, 10722.32509097])

- Comments:

    - In short, what the array presented above is a prediction of price given that

        - We used the first 10 rows of data from the dataset.

        - We used only the variables within Z → 'horsepower', 'curb-weight', 'engine-size' and 'highway-mpg' to predict it.

        - The regression technique used here was Multiple Regression (MR).

CONCLUSION

- In this article, we showed how Pipeline is a shortcut to perform data fitting and prediction.
- Previously, we used Linear Regression (LR)[1], Multiple Regression (MR)[2] and Polynomial Regression (PR)[3] on a dataset containing 200 car models…. So as to do price prediction.
- Specifically, we wanted to find out whether LR / MR / PR was a good fit or not.
- However, they were time consuming and tedious as they comprised of many steps.
- With the Pipeline module, it significantly reduced the time and effort (of data fitting, visualizing etc…) … and we could jump straight to price predictions.

[1] https://www.alvinang.sg/s/Simple-Linear-Regression-with-Jupyter-Notebook-by-Dr-Alvin-Ang.pdf

[2] https://www.alvinang.sg/s/Multiple-Regression-with-Jupyter-Notebook-by-Dr-Alvin-Ang.pdf

[3] https://www.alvinang.sg/s/Polynomial-Regression-with-Jupyter-Notebook-by-Dr-Alvin-Ang.pdf

ABOUT DR. ALVIN ANG

Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.