DS203 : Exercise 3

Nirav Bhattad

Last updated August 30, 2024

Step 1

Step 1

Review the Jupyter Notebook `E3.ipynb` and:
(a) Create a summary of the code therein.
(b) Are there any learnings from this code that you wish to highlight?

This Python Notebook `E3.ipynb` presents an analysis of various regression models applied to a dataset using polynomial features of different degrees. The models analyzed include Linear Regression, Support Vector Machine (SVM) Regression, Random Forest, Gradient Boosting, K-Nearest Neighbors (KNN), and Neural Networks. The performance of each model is evaluated based on metrics such as R-squared, Mean Squared Error (MSE), Durbin-Watson, and Jarque-Bera statistics.

We use the dataset given in `E3-MLR3.xlsx`. The dataset contains 548 samples of (y, x_1) to train the model and 152 samples of (y, x_1) to test the model.

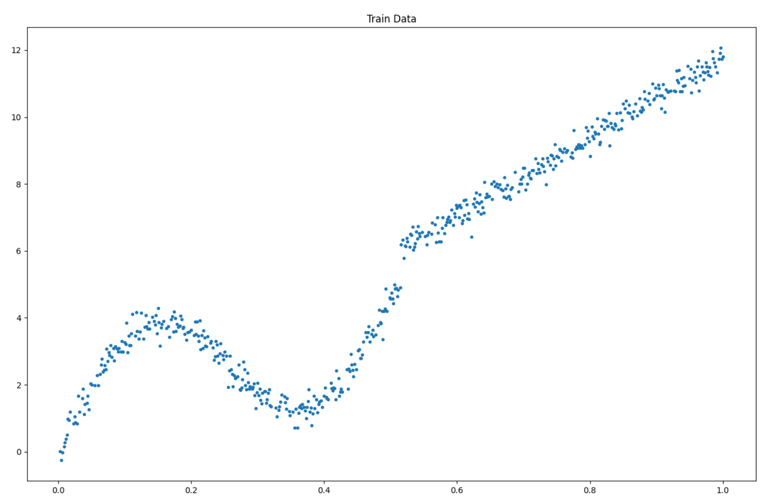

Figure 1: Training Data used in the analysis

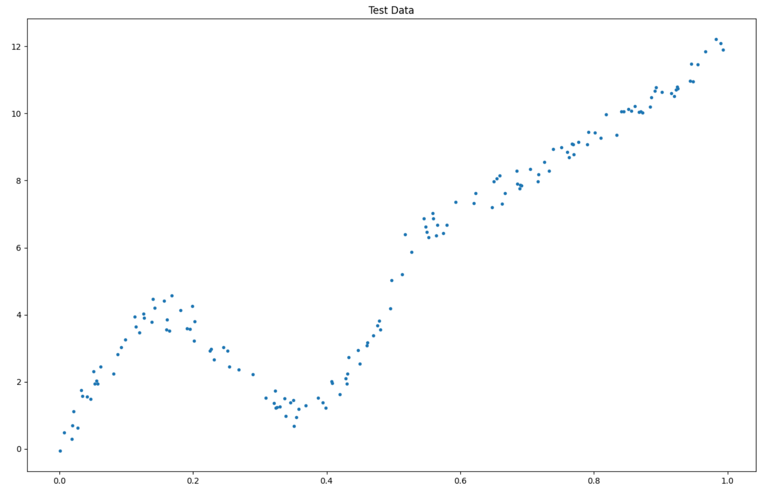

Figure 2: Testing Data used in the analysis

Step 2

Step 2

Review the **Sklearn** documentation for each Sklearn function used in the Notebook (eg. `PolynomialFeatures`, `LinearRegression`, `mean_squared_error`, etc.) and create a description of each to explain, to yourself, the functionality, the input parameters, and the outputs generated. Present this in the form of a two-column - Table (Function name | Description).

`PolynomialFeatures`	This function is used to generate polynomial and interaction features. It generates a new feature matrix consisting of all polynomial combinations of the features with a degree less than or equal to the specified degree. The input to this function is the degree of the polynomial features to be generated. The output generated is a new feature matrix consisting of all polynomial combinations of the features with a degree less than or equal to the specified degree. Input Parameters: • **degree**: `int` or `tuple (min_degree, max_degree)`, `default=2` • **interaction_only**: `bool`, `default=False` • **include_bias**: `bool`, `default=True` • **order**: `str` in {`'C'`, `'F'`}, `default='C'` Attributes: • **powers_**: `ndarray` of shape (`n_output_features_`, `n_input_features_`) • **n_output_features_**: `int` • **n_features_in_**: `int` • **feature_names_in_**: `ndarray` of shape (`n_input_features_`,) Output: • **ndarray** of shape (`n_samples`, `n_output_features_`)
`LinearRegression`	This function is used to fit a linear model. It fits a linear model with coefficients $w = (w_1, ..., w_p)$ to minimize the residual sum of squares between the observed targets in the dataset and the targets predicted by the linear approximation. Input Parameters: • **fit_intercept**: `bool`, `default=True` • **copy_X**: `bool`, `default=True` • **n_jobs**: `int`, `default=None` • **positive**: `bool`, `default=False` Attributes: • **coef_**: `ndarray` of shape (`n_targets`, `n_features`) • **intercept_**: `ndarray` of shape (`n_targets`,) • **rank_**: `int` • **singular_**: `ndarray` of shape (`min(X, y)`,) • **n_features_in_**: `ndarray` of shape (`n_targets`,) Output: • **self**: returns an instance of self

`SVR`	This function is used to fit the Support Vector Regression model. Input Parameters: • **kernel**: `str`, `default='rbf'` • **degree**: `int`, `default=3` • **gamma**: `float`, `default='scale'` • **coef0**: `float`, `default=0.0` • **tol**: `float`, `default=1e-3` • **C**: `float`, `default=1.0` • **epsilon**: `float`, `default=0.1` • **shrinking**: `bool`, `default=True` • **cache_size**: `float`, `default=200` • **verbose**: `bool`, `default=False` • **max_iter**: `int`, `default=-1` Attributes: • **coef_**: `ndarray` of shape (1, `n_features`) • **dual_coef_**: `ndarray` of shape (1, `n_SV`) • **fit_status_**: `int` • **intercept_**: `ndarray` of shape (1,) • **n_features_in_**: `int` • **feature_names_in_**: `ndarray` of shape (`n_features`,) • **n_iter_**: `int` • **n_support_**: `ndarray` of shape (1,) • **shape_fit_**: `tuple` of `int` of shape (`n_dimensions_of_X`,) • **support_**: `ndarray` of shape (`n_SV`,) • **support_vectors_**: `ndarray` of shape (`n_SV`, `n_features`) Output: • **ndarray** of shape (`n_samples`,)

Step 3

Step 3

Generate outputs by setting **degree = 1**, **degree = 3**, **degree = 6**, **degree = 10**, in the `PolynomialFeatures` function used in `E3.ipynb` and analyze them as follows:

(a) Review the `augmented_data.csv` file generated in each case and document your observations.
(b) Create an overall qualitative summary based on a review and analysis of the Figures generated.
(c) Summarize and explain the variations in the metrics **across regression methods for a given degree** (ie. a given set of polynomial features). Cover both, train and test, metrics, and compare them.
(d) Summarize and explain the variations in the metrics **across degrees for a given regression method**. Cover both, train, and test metrics, and compare them.
(e) When **degree = 1** which method(s) result in acceptable regression models? Why?
(f) When **degree = 6** which method(s) result in acceptable regression models? Why?
(g) As the value of degree is increased to 10 which regression methods show the most impact? Why?
(h) Why do non-parametric methods like `KNN` and `Decision Tree` based methods generate good results even without feature engineering?
(i) What are the limitations of the non-parametric methods?
(j) Given the results, should LinearRegression be used at all? Why, when? Justify your answer.

Step 4

Step 4

In step 2 you have already reviewed the important parameters and outputs related to the regression methods. Select 2-3 methods, vary the important parameters, and observe how the outputs change (eg. see the function calls for `SVR` and `MLPRegressor`). Document the outcomes of your experiments.

Step 5

Step 5

Review **`Sklearn`** documentation to understand and experiment with a few more (2-3) regression methods, in addition to the ones listed above, and document the outcomes of your experiments.

Main Learnings