Code Flow and Short Report of E3.ipynb

This Python notebook explores various machine learning algorithms for regression on a dataset with features x1 and y. Here's a breakdown of the code flow and key findings:

Data Preparation:

* Imports: Necessary libraries like pandas, numpy, and sklearn are imported.
* Data Reading: Data is loaded from an Excel file with separate sheets for training and testing (E3-MLR3.xlsx).
* Feature Separation: Features and target variable (y) are separated in both training and testing sets.
* Polynomial Features: Features are augmented using PolynomialFeatures with degree 10. This creates new features based on combinations of existing features.
* Augmented Data Save: The augmented test data and target variable are saved to a CSV file for further analysis.

Algorithm Evaluation:

* Algorithm Selection: Six algorithms are chosen: Linear Regression, SVM Regression, Random Forest, XGBoost, KNN, and Neural Network.
* Metric Tables: DataFrames are created to store training and testing metrics for each algorithm.
* Looping through Algorithms: Each algorithm is trained on the augmented training data (X\_train\_poly) and evaluated on both training and test sets.
* Training and Testing: Predictions are made using the trained models for both training and testing data.

Metrics Calculation: Performance metrics like R-squared, MSE, Durbin-Watson statistic, and Jarque-Bera test are calculated for each algorithm on both datasets.

Residuals Analysis: Additional metrics are calculated using stats models for normality and autocorrelation checks on residuals (differences between predicted and actual values).

Visualisation:

Subplots: A grid of subplots is created to visualise predictions and residuals for each algorithm.

Scatter Plots: Four plots are created for each algorithm:

Training data vs. predictions

Training data residuals

Testing data vs. predictions

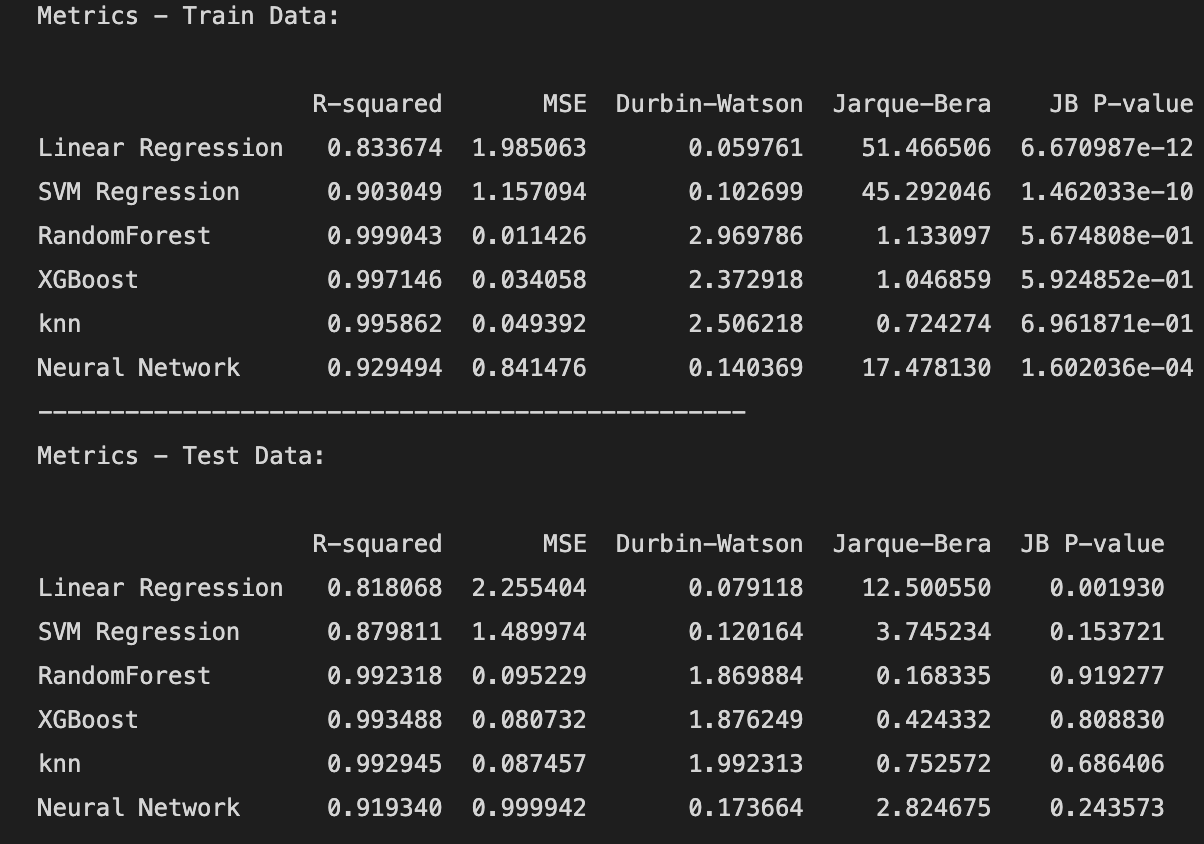
Testing data residuals

Now let’s Take a look at the results / outputs of the above code for the variable parameter degree = 1, 3, 6, 10 .

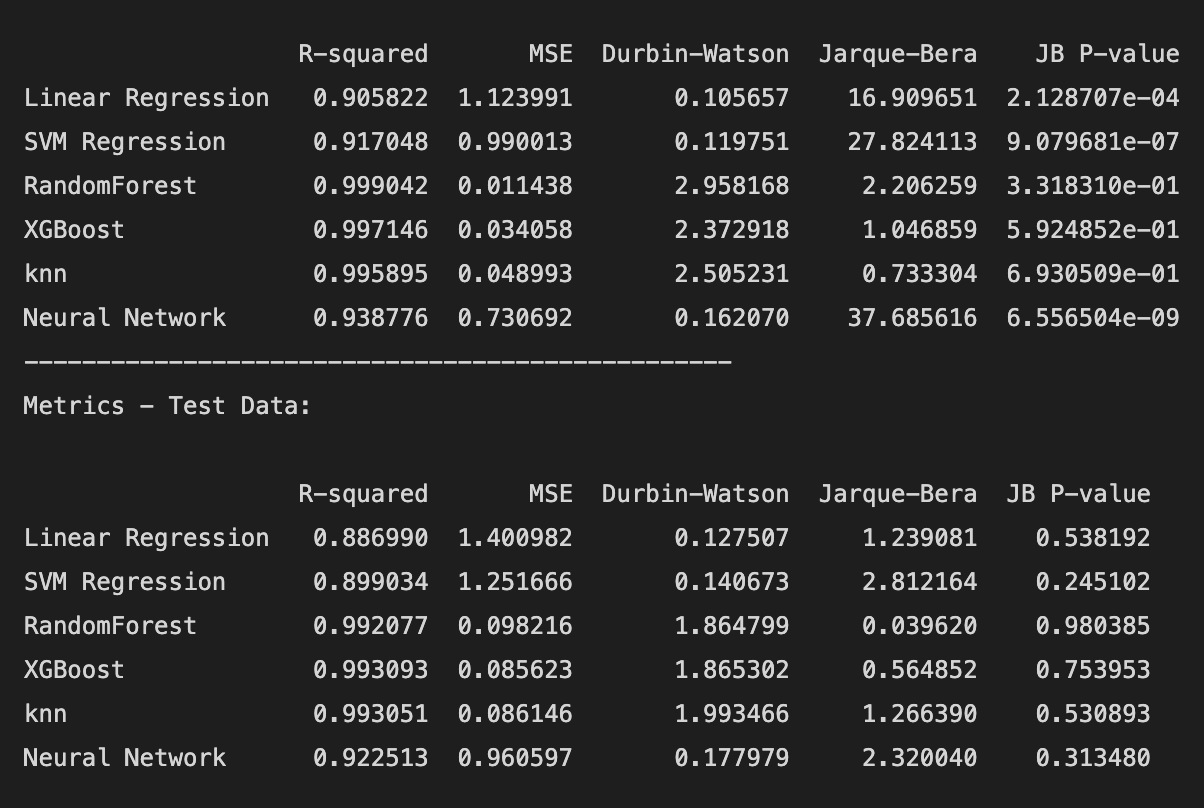
| Function | Description | Module |
| --- | --- | --- |
| train\_test\_split | Splits data into training and testing sets. | model\_selection |
| PolynomialFeatures | Creates polynomial features from existing features. | preprocessing |
| Model Fitting: |  |  |
| - LinearRegression | Fits a linear model to the data. | linear\_model |
| - SVR | Fits a support vector regression model with a specified kernel. | svm |
| - RandomForestRegressor | Fits a random forest regression model. | ensemble |
| - GradientBoostingRegressor | Fits a gradient boosting regression model. | ensemble |
| - KNeighborsRegressor | Fits a k-nearest neighbors regression model. | neighbors |
| - MLPRegressor | Fits a multi-layer perceptron regressor. | neural\_network |
| Evaluation Metrics: |  |  |
| - mean\_squared\_error | Calculates the mean squared error between predictions and target values. | metrics |
| - r2\_score | Calculates the R-squared score (coefficient of determination). | metrics |
| Additional Analysis: |  |  |
| - GridSearchCV | Performs grid search for hyperparameter tuning (not used in this code). | model\_selection |
| - durbin\_watson | Performs Durbin-Watson test for autocorrelation in residuals. | statsmodels.stats |
| - jarque\_bera | Performs Jarque-Bera test for normality of residuals. | statsmodels.stats |

Sklearn Function v/s Description

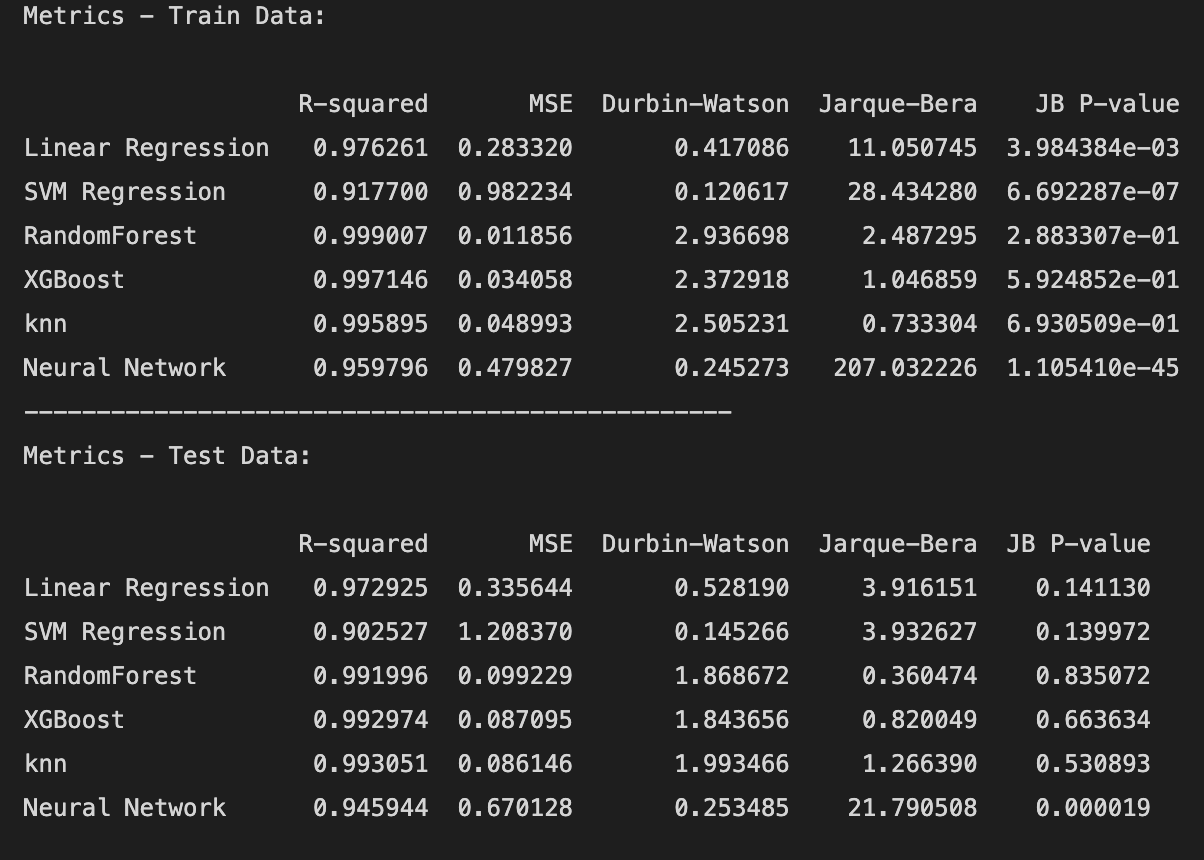
# Degree Wise Plots and Performance analysis



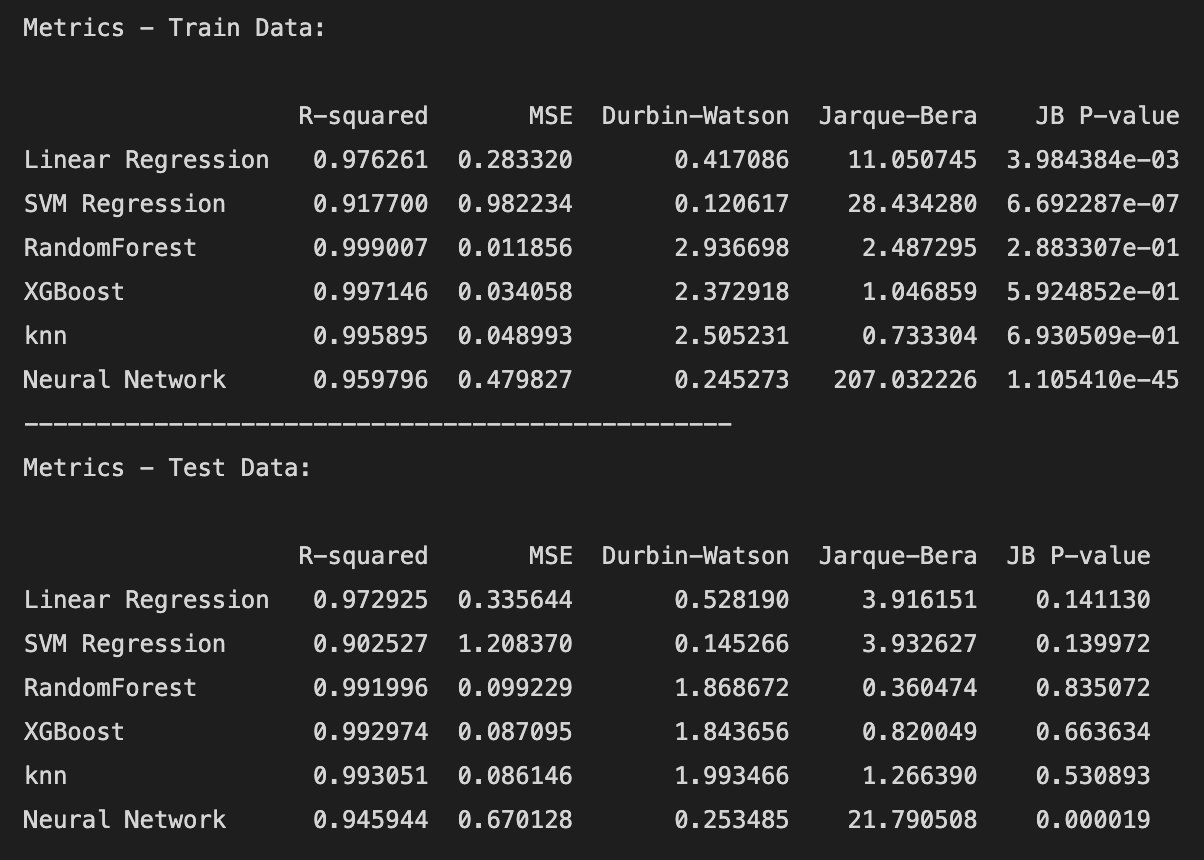
***Fig 1.1 - Performance Metrics degree = 1***



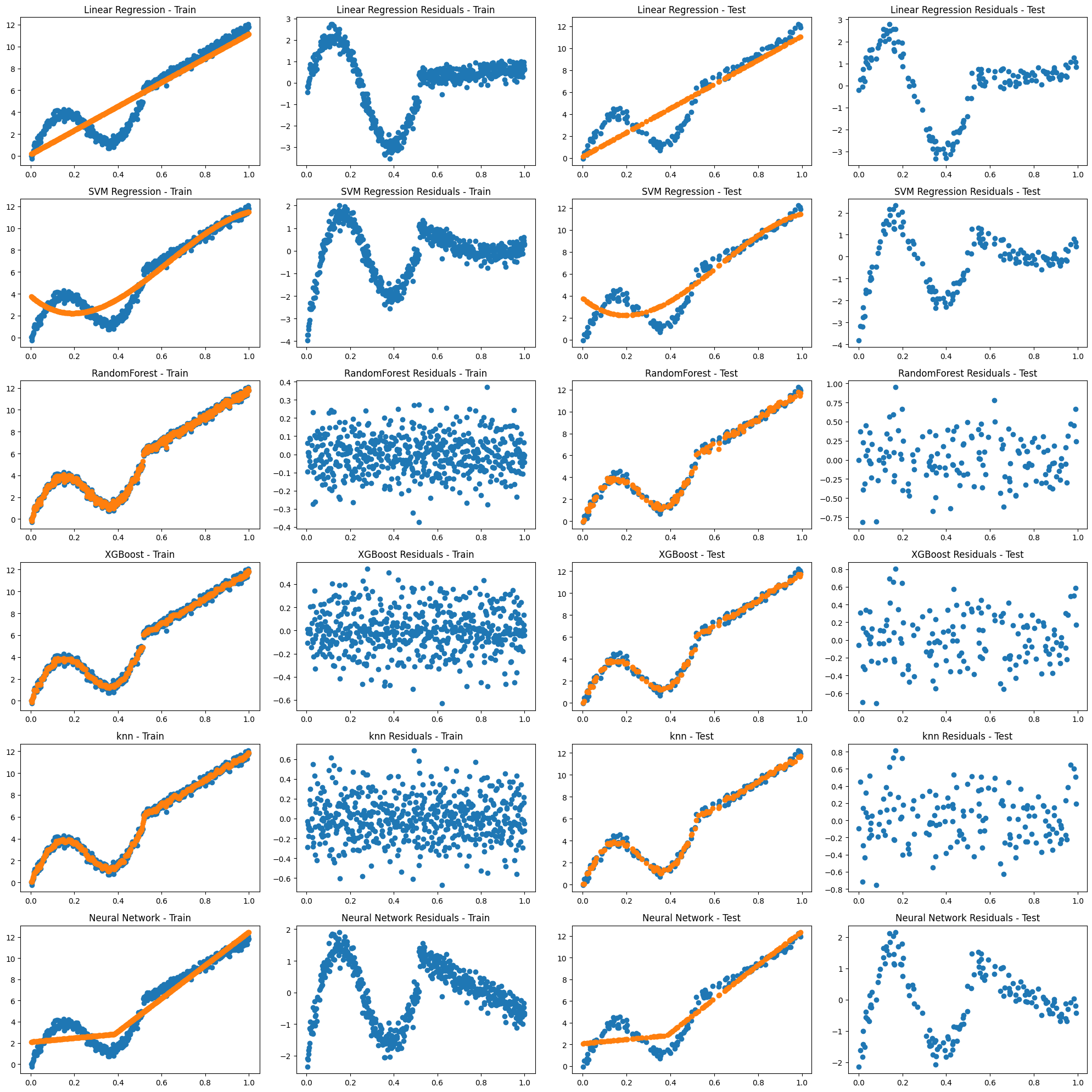
***Fig 1.2 - Performance Metrics degree = 3***



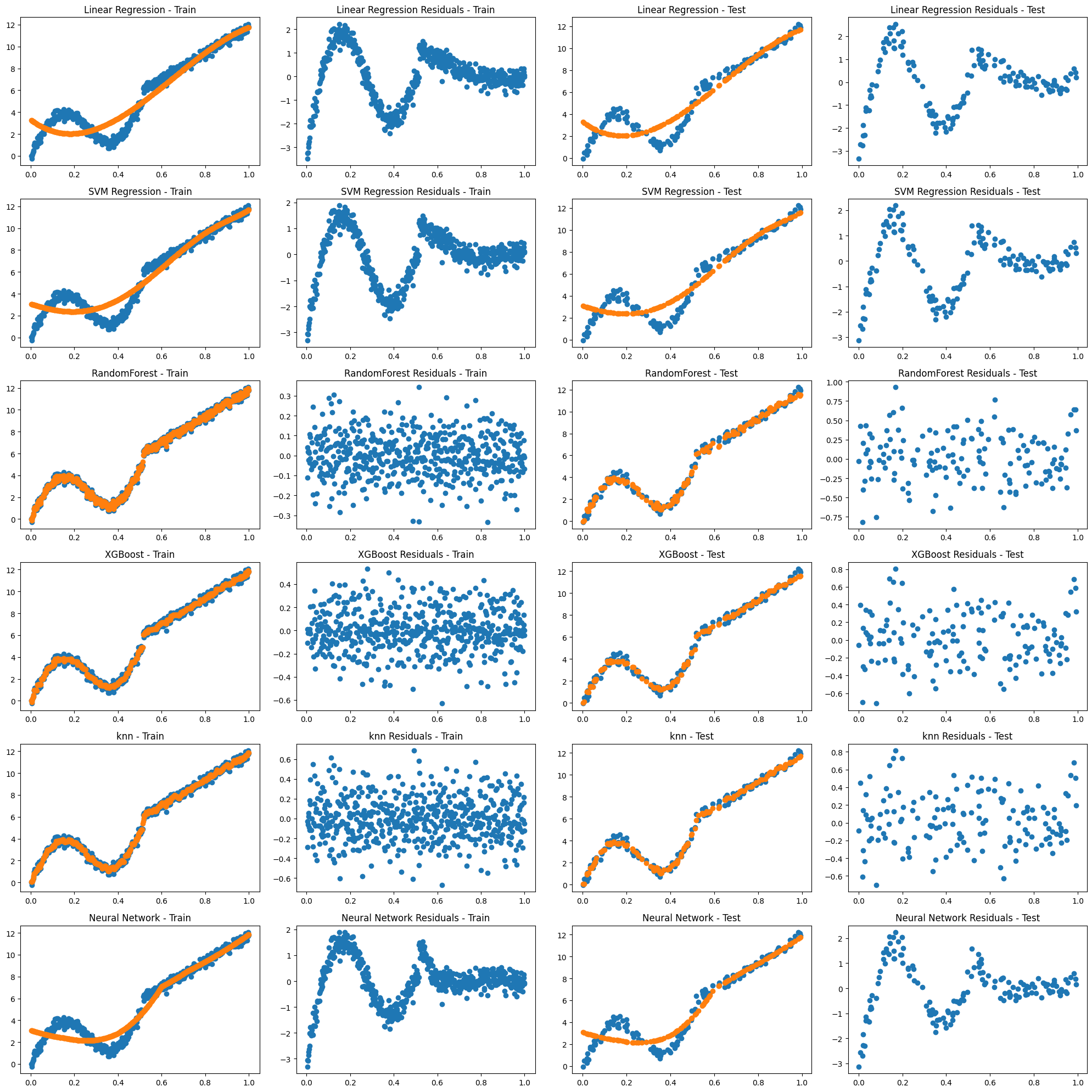
***Fig 1.3 - Performance Metrics degree = 6***



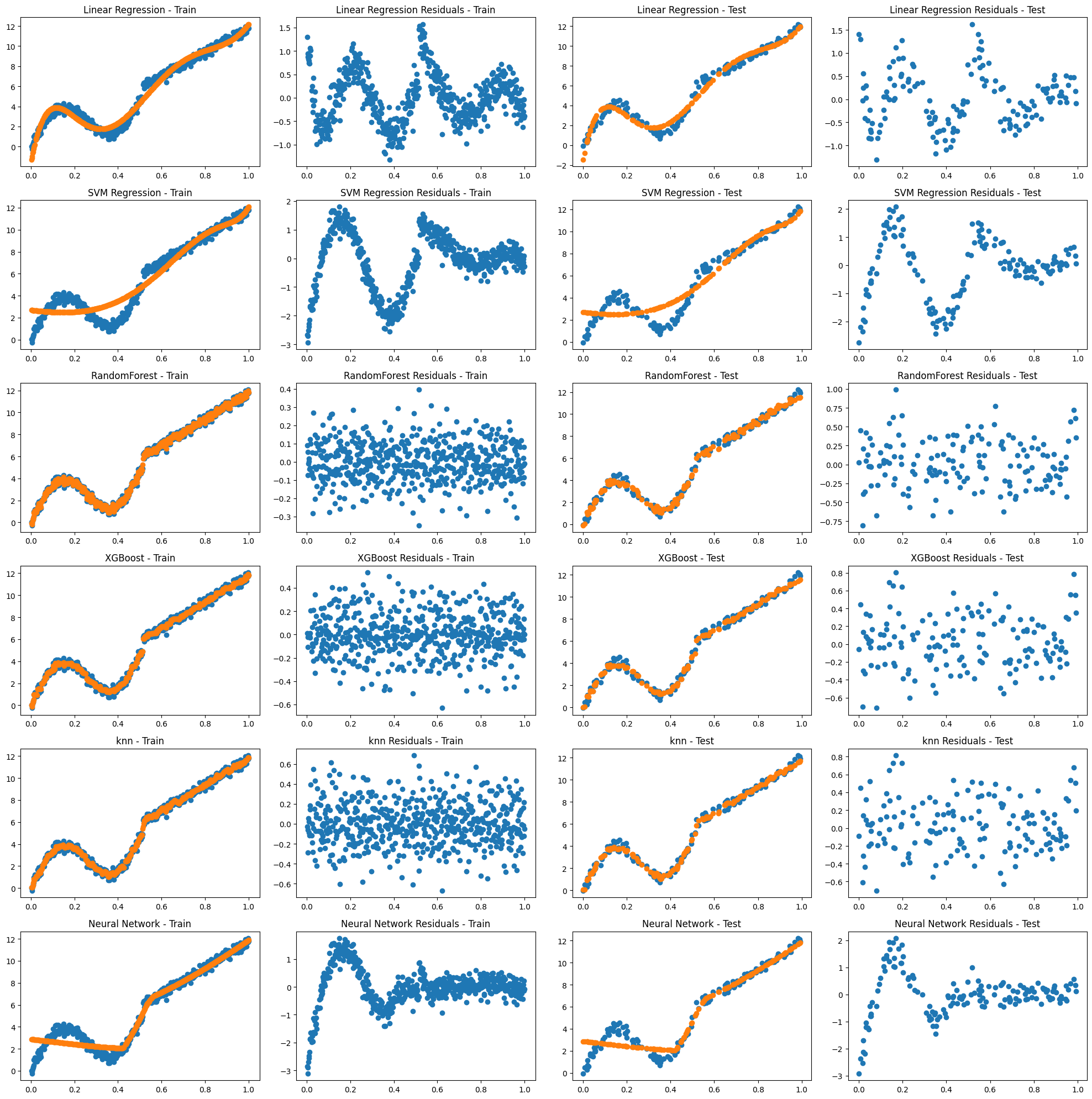
***Fig 1.4 - Performance Metrics degree = 10***



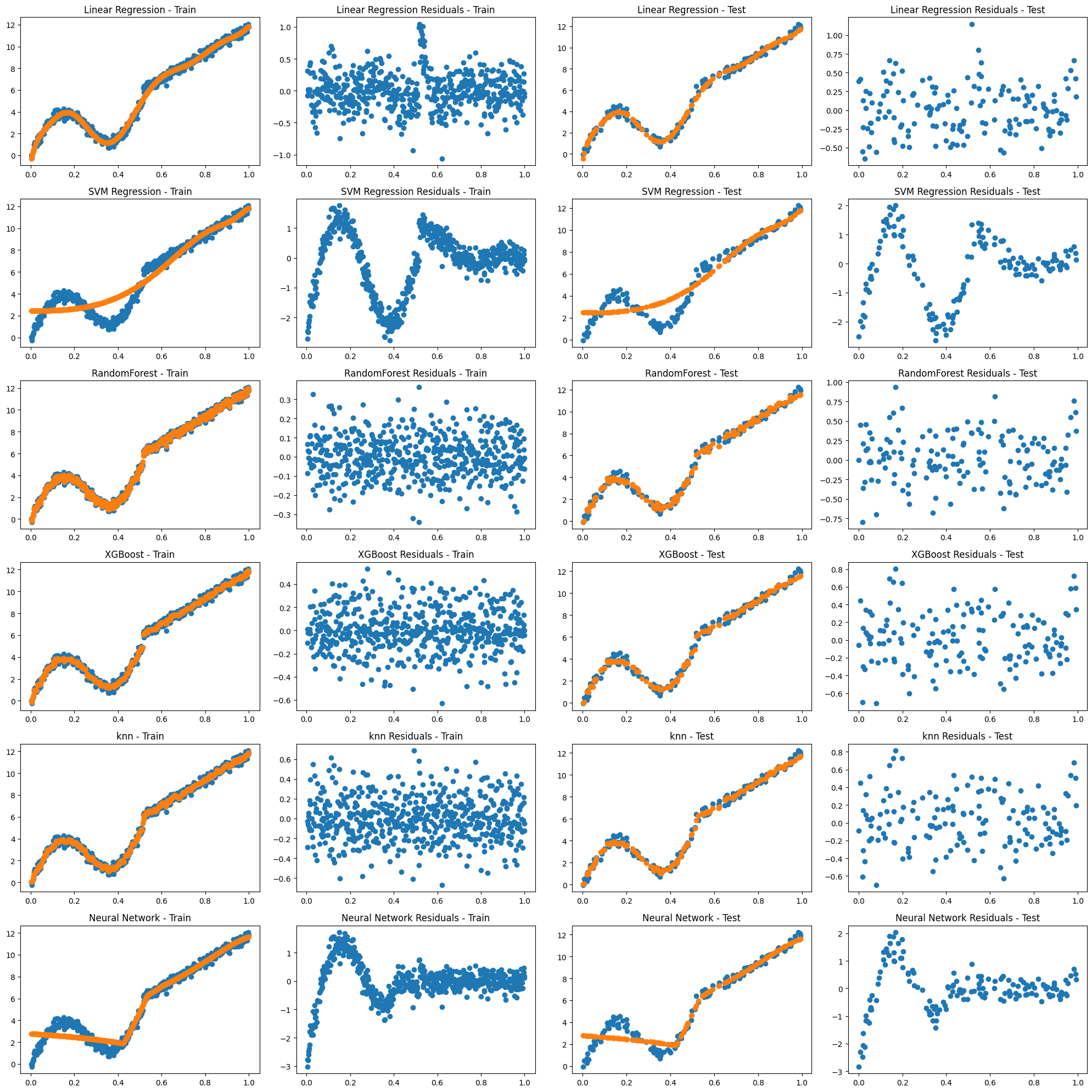
***Fig 2.1 - Performance Metrics degree = 1***



***Fig 2.2 - Performance Metrics degree = 3***



***Fig 2.3 - Performance Metrics degree = 6***



***Fig 2.3 - Performance Metrics degree = 10***

Analysis of Performance metrics of Degree = 1 :

Based on the Fig 1.1 , 2.1:

* On training data: Random Forest and XGBoost achieve the highest R-squared values (close to 1), indicating a very good fit to the training data. However, this might lead to overfitting.
* On testing data: XGBoost still has the highest R-squared (0.993488), followed by Random Forest (0.992328) and KNN (0.992945). These models seem to generalize well to unseen data.
* In terms of MSE: XGBoost also has the lowest MSE on both training and test data, again suggesting good performance.
* Looking at Durbin-Watson: Most models have values close to 2, indicating no significant autocorrelation concerns.
* Jarque-Bera test: While some models have p-values lower than 0.05, particularly on training data, it's important to consider sample size and other factors before drawing strong conclusions about normality.

Augmented Data output :

The Output shows features

**Key Takeaways:**

* XGBoost appears to be the best performing model overall, based on both training and testing metrics.
* Random Forest and KNN are also strong contenders, especially considering their interpretability compared to XGBoost.

Visual Analysis of Plots Fig 2.1

Linear Regression , SVM Regression , Neural Network shows patterns in their residual distribution indicating a sign of bias which is true for both test and train sets meanwhile the distributions for XGboost , Random Forest and KNN regression are well distributed showing no obvious signs of bias and overfitting/underfitting which is made true by the performance metrics analysis above.

U-shaped or inverted U-shaped patterns: This suggests the model is under-fitting the data and cannot capture the non-linear relationship between x and y. Thus, Tree based models and higher order regressions perform better in these scenarios.

Straight line patterns:

* Horizontal bands: This suggests equal underestimation or overestimation across different values of x. Possible causes include missing non-linearity, incorrect data transformation, or outliers.
* Diagonal lines: This indicates a proportional error, meaning the model consistently underestimates/overestimates by a certain amount as x increases. Potential reasons include missing interaction terms between features, incorrect model selection, or inadequate feature engineering.

Analysis of Performance metrics of Degree = 3 :

Based on the Fig 1.2 , 2.2:

* R-squared: Both training and test sets show high R-squared values for most models, indicating good fit. Random Forest and XGBoost achieve the highest values (>0.99) across both datasets.
* MSE: Lower MSE values suggest better performance. XGBoost consistently has the lowest MSE on both training and test data, followed closely by Random Forest and KNN.
* Durbin-Watson: Most models have values close to 2, indicating no significant autocorrelation concerns in residuals.
* Jarque-Bera test: While some models have p-values below 0.05 (suggesting non-normality), consider sample size and other factors before drawing strong conclusions.

Model Comparisons:

* Training Data:
  + XGBoost and Random Forest outperform other models in terms of R-squared and MSE.
  + SVM Regression and Neural Network also achieve good performance.
* Test Data:
  + XGBoost maintains its lead with the highest R-squared and lowest MSE.
  + Random Forest, KNN, and SVM Regression follow closely, suggesting good generalization potential.
  + Neural Network shows slightly lower performance on the test set compared to training.

Augmented Data output :

The Output shows features

Key Takeaways:

* XGBoost appears to be the best performing model overall, based on both training and test metrics.
* Random Forest and KNN are strong contenders, especially considering their interpretability compared to XGBoost.
* It's crucial to compare performance on unseen data (test set) to avoid overfitting.
* While the Jarque-Bera test results hint at non-normality, further analysis or transformation might be needed depending on your specific assumptions and requirements.

Visual Analysis of Plots Fig 2.2

We see the same problems that persisted in the Fig2.1 for Plots - Linear Regression is staring to adapt as degree = 3 introduces more flexibility for the model similar things can be said for SVM and NN but tree based models again consistently perform way better . A good hypothesis would be the feature engineering applied may be unsuitable for this type of non-linear relationship between y v/s x especially for non-tree based methods.

Pattern based analysis of Previous sections holds true for this as well.

Overall Trend as degree = 3 to degree = 6 :

* Both tables 1.2 , 1.3 seem to show similar trends, with XGBoost, Random Forest, and KNN consistently performing well across metrics.

Specific Changes:

* R-squared:
  + Training data: All models show slight decreases except Linear Regression and Neural Network. The change is marginal for most models.
  + Test data: Similar pattern with slight decreases for most models, except KNN which maintains its R-squared.
* MSE:
  + Training data: Minimal changes for most models except Neural Network which shows a slight improvement.
  + Test data: XGBoost shows a slight increase, while KNN and Random Forest remain almost unchanged. SVM and Neural Network have moderate increases.
* Durbin-Watson:
  + Minimal changes across both datasets and models.
* Jarque-Bera:
  + Training data: Neural Network shows a significant increase in p-value, suggesting potential normality of residuals. Other models exhibit minor changes.
  + Test data: Similar changes with Neural Network's p-value increasing further, and other models showing minor variations.

Key Takeaways:

* The overall performance rankings of the models remain largely consistent across both tables.
* XGBoost continues to be the leader in terms of R-squared and MSE, especially on test data.
* KNN maintains its strong performance with slight variations.
* Random Forest shows minimal fluctuations in results.
* Linear Regression and SVM Regression see minor changes, suggesting they might be sensitive to specific parameter settings or data variations.
* Neural Network exhibits some improvement in normality of residuals for training data but slightly underperforms on test data compared to the previous run.
* Choice between degree = 3 and 6 boils down to computational costs and preferability .

Visual Analysis of Plots Fig 2.3