Effort Estimation for Software Development using Predictive Models

P25

Akash Gupta, Arun Srinivasan Parthasarathy, Vishwesh Sangarya

What are we doing?

Given the complexity, system constraints, required reliability, developer experience, etc., the project goal is to estimate the effort in the development of a software product.

Project link: https://tinyurl.com/mmtxwpv4

Why? - Importance

- In software development, effort estimation is a parameter that determines the realistic effort required by a team to develop, maintain or scale a software application.
- It is represented in hours or months, and is useful in planning a project and allocating budgets in the early stages of the projects.
- The lack of having a good effort estimate can result in issues such as wrong budget allocation, unrealistic expectations, rejection of bid for project ownership, substandard services, poor team morale, and continuous delays in delivering satisfactory updates to the customer.

Background

- The very first breakthrough started with **Albrecht et al., 1983**, where the effort is determined by using Function Points. Non-algorithmic approach for non-technical personnel, uses requirements specification.
- **Srinivasan et al., 1995** on the other hand, uses Neural Networks and Decision Trees to estimate the effort.
- **Shepperd et al., 1997** estimates effort using analogies, i.e., to find the effort of a new project by finding its similarities with a stored set of completed projects. Similarity is calculated using Euclidean Distance.
- **Kocaguneli et al., 2012** replicates the effort estimation by combining multiple simpler estimation models to further boost the accuracy.

How? - Datasets

- **Cocomo81**: Available through Promise repository, this dataset contains 63 instances of software projects, spread across 17 attributes which are all numeric. In these 17 attributes, 15 are effort multipliers, 1 is the number of lines of code and the last one is the Effort, which is computed in person-months.
- **Desharnais** : Available through Promise repository, this dataset is a collection of 81 software projects, which contain 12 attributes that could have direct influence on the effort required to complete the project, in person-hours.

How? - Performance measures

RMSE (Root mean square error)

$$\mathrm{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

MAE (Mean absolute error)

$$\mathrm{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \left| y_i - \hat{y}_i \right|.$$

MMRE (Mean of magnitude of relative error)

$$\text{Relative error} = \frac{|\ \text{Actual} - \text{Predicted}\ |}{\text{Actual}}$$

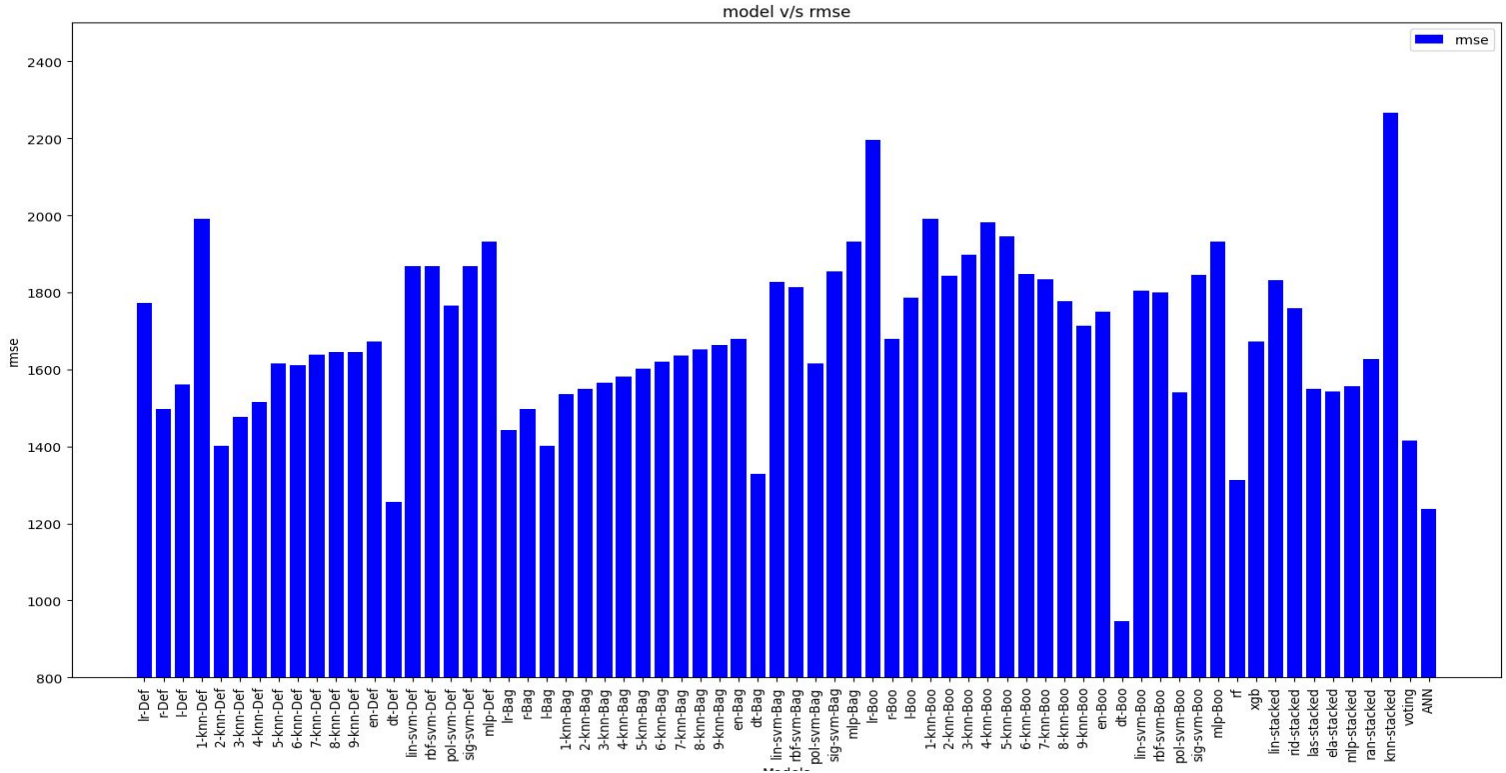
R^2

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

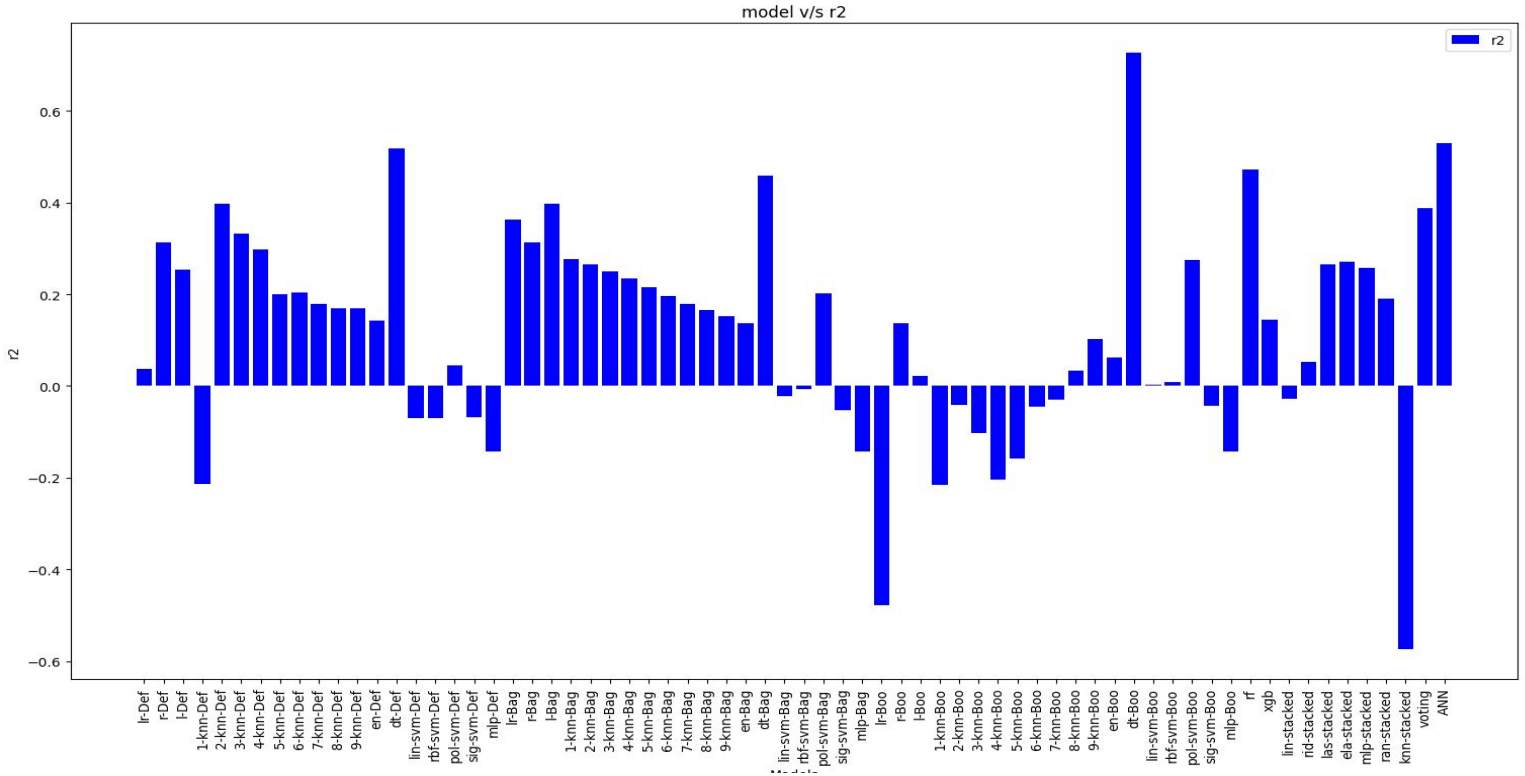
How? - Method and Experiment Setup

- Load datasets using pandas. Parse and remove non-numeric values. Perform **Exploratory Data Analysis (EDA)** and analyze box, scatter, histogram plots and correlation matrix.
- Perform preprocessing steps like **normalization, PCA,** and splitting the datasets into folds to apply cross validation. This experiment uses a **10-fold** approach, where testing is (identifier mod 10 = i - 1, i ϵ {1, 10}).
- Applied individual models including **LinearRegression, Ridge, Lasso, KNN, SVM, Decision Trees, Artificial Neural Networks, etc**.
- Used **bagging** and **boosting (Adaboost)** techniques to form **ensemble**.
- Applied **RandomForest**, **XGBoost**, **stacking** and **voting** approach for ensemble of models.
- Compared the performance using aforementioned performance measures.

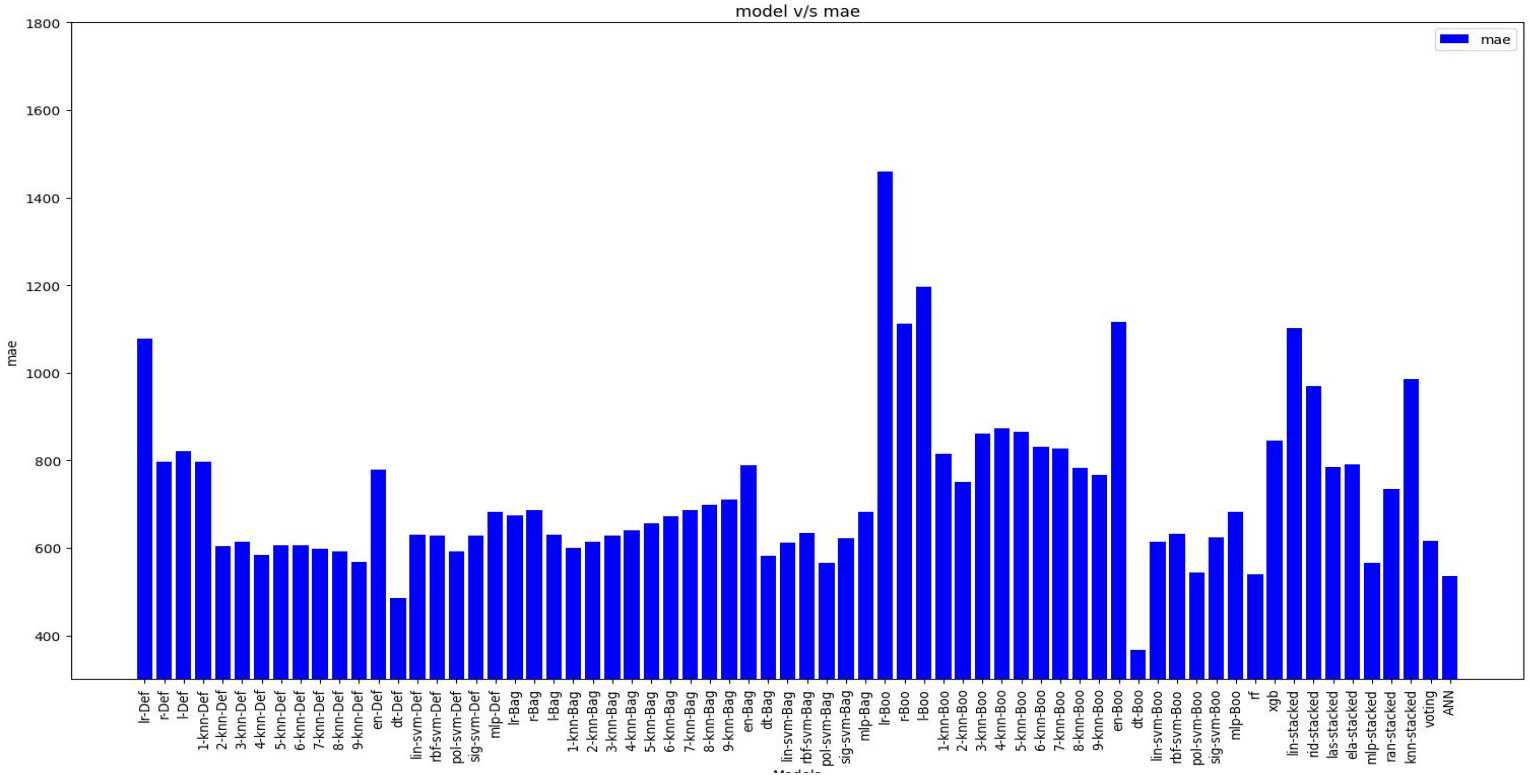
Experiment results - Cocomo81 - RMSE



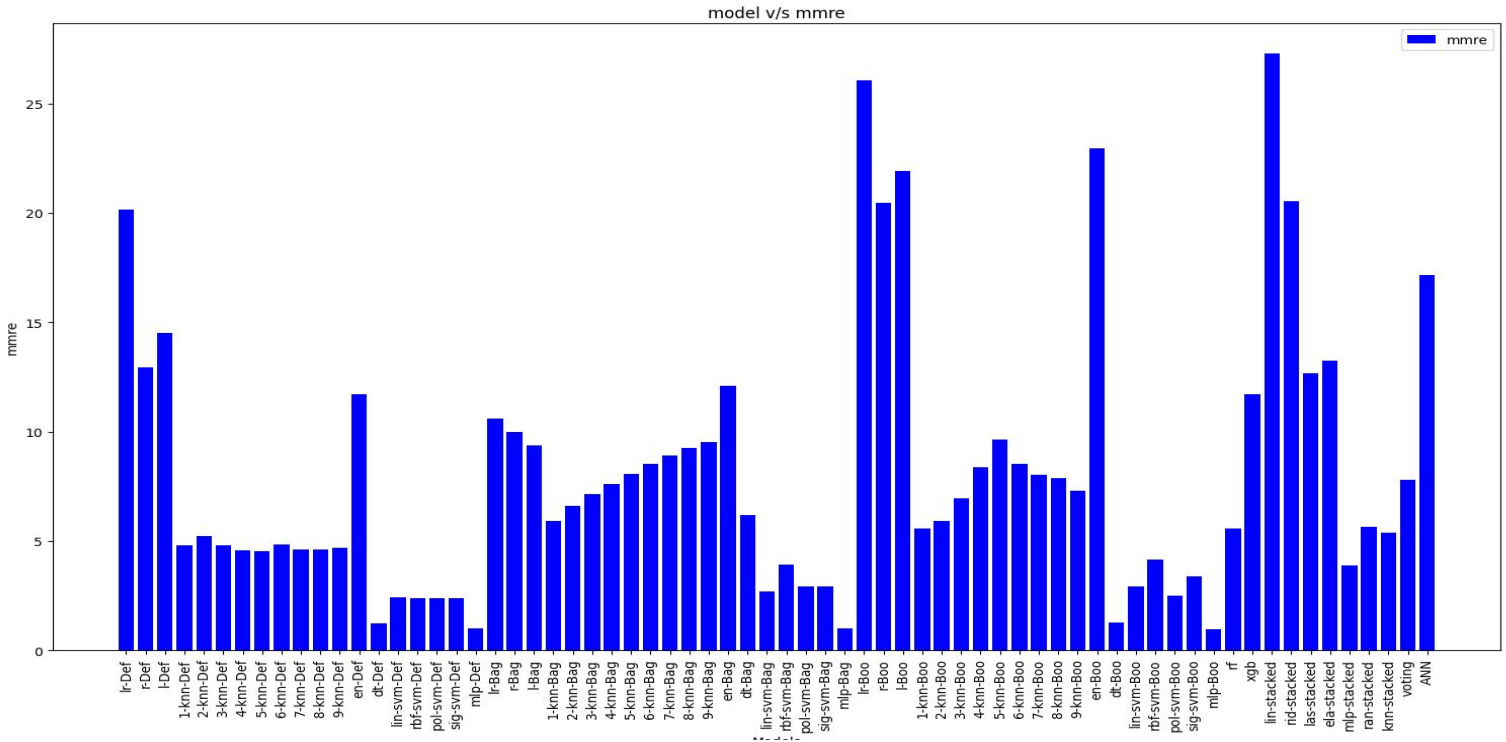
Experiment results - Cocomo81 - R^2



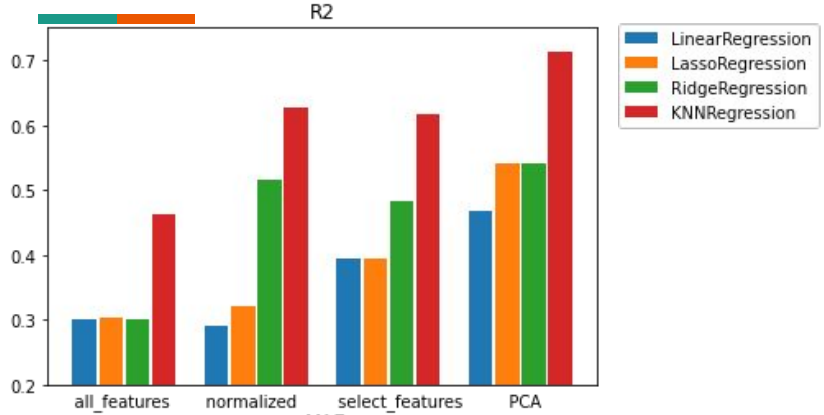
Experiment results - Cocomo81 - MAE

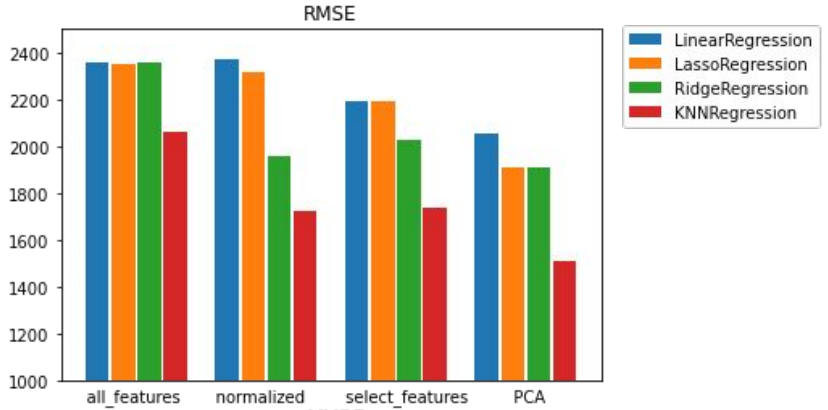


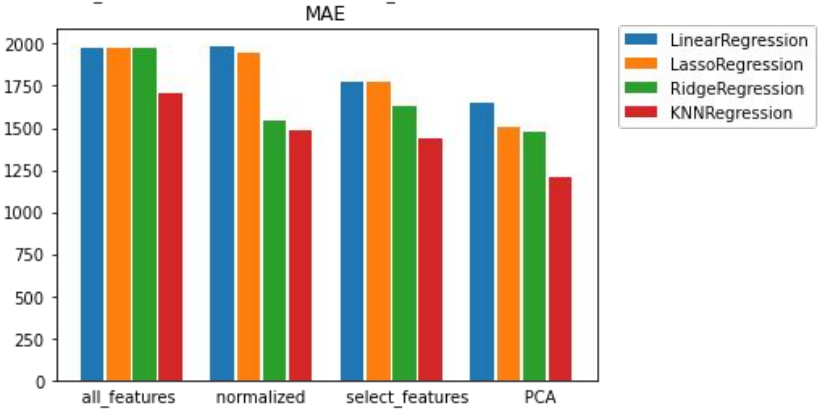
Experiment results - Cocomo81 - MAE & MMRE

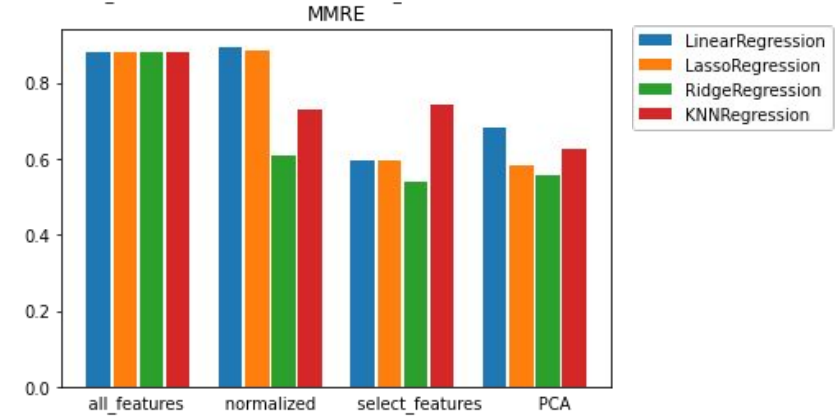


Experiment results - Desharnais









Conclusions

Desharnais	Cocomo81
• **No** non-numeric data observed. • **Five** of the eleven features have **high correlation** with the target. • Independent variables have varying distributions, target variable has a right skewed normal distribution. • **KNN regression using 5 principal components** and 6 nearest neighbours gives the best results - low error measures and highest R^2 score. • Using 6 and 7 principal components gives good consistent results on linear regularized models such as Lasso and Ridge regression.	• **No** non-numeric data observed. • **Only one feature** has high correlation with target. • Independent variables **not** normally distributed. • **Decision tree with boosting (ensemble)** gives the best results across all performance measures. Has the lowest error measures and highest R^2 score. Unlike other models, which perform better on some measures, DT performs better on all. • Applying PCA leads to worse results except when applied with ANN.

Future Scope

- Currently the main drawback is in the number of datasets that are present with regards to effort estimation.
- Even in the datasets that are present, the number of samples are quite low to actually innovate an effective Neural Network approach.
- Since the data samples are very less, we need to run an extremely high amount of epochs to actually the fit the model, which results in overfitting.
- Thus, more real world data needs to be captured to perform a more effective and relevant analysis.
- Also, here the process of effort estimation is ending with the initial product development, and can be further extended to include maintenance.

Questions?

Shoot us an email at:

- agupta57@ncsu.edu
- apartha4@ncsu.edu
- lsangar@ncus.edu