UIET CSJM UNIVERSITY KANPUR

B.Tech PROJECT

Software Project Estimation Using Artificial Intelligence

Bachelor of Technology

Information Technology

Proposed By:

VANSHIKA SRIVASTAVA
CSJMA20001390191

SIDDHARTH SINGH
CSJMA20001390187

Project Guide:

Er. PRATEEK SRIVASTAVA
Assistant Professor

CONTENT

CERTIFICATE

This is to certify that the project entitled "**Software Project Estimation Using Artificial Intelligence**" has been Submitted by Siddharth Singh and Vanshika Srivastava under the guidance of Er. Prateek Srivastava in partial fulfillment of the degree of Bachelor of Technology in "**INFORMATION TECHNOLOGY"**, from **UNIVERSITY INSTITUTE OF ENGINEERING AND TECHNOLOGY KANPUR** during academic year **2023-24.**

Signature Of Candidate

Siddharth Singh

Vanshika Srivastava

This is to certify that the above statement made by the candidate is correct to the best of my Knowledge.

Head of Department
Dr. Alok Kumar

Project Guide
Er. Prateek Srivastava

ACKNOWLEDGEMENT

The successful completion of this project report would not have been possible without the support and assistance of Er. Prateek Srivastava . We feel immensely blessed to have gotten this during the project. We would like to take this opportunity to offer our earnest admiration to each and every one of them. We express our sentiment of gratitude to Er. Prateek Srivastava Sir , who has been a continuous source of inspiration as our Project Guide. Our gratitude for his trust and generosity goes beyond words. We are indebted and thankful to our learned and revered Dr. Alok Kumar Head of Department of Information technology, University Institute of Engineering Technology, for his encouragement, support and providing a meticulous platform to learn. Finally, thank all our teachers,who prepared us for this endeavor. We owe you all our success.

ABSTRACT

This project involves Software Project Estimation Using Artificial Intelligence.Software estimation is a crucial part of software project development..This project builds a software estimation model using machine learning approach. Various stacking models considered and evaluated were stacking using a generalized linear model, stacking using decision tree, stacking using a Support Vector Machine, stacking using Random Forest, stacking using Linear Regression, and stacking using KNN(K-Nearest Neighbour). Different machine learning algorithms are applied to public datasets.Datasets considered for estimation are Cocomo81,china,DeSharnais,kemerer,Maxwell and Albrecht Dataset.. Evaluation measures used are mean absolute error, root mean squared error, mean absolute percentage error and r-squared. The results proved that the proposed stacking using XGBoost provides the best results compared with single model approaches using the machine learning algorithms.

INTRODUCTION

The "Software Project Estimation Using AI" initiative is a pioneering endeavor that converges cutting-edge technologies of Artificial Intelligence (AI) and Machine Learning (ML) algorithms to redefine software development project planning. By leveraging historical data and real-time adaptation, this project aims to revolutionize project management by offering precise estimations of effort..

SOFTWARE PROJECT:

A software project is a structured endeavor aimed at creating, developing, or enhancing software applications to meet specific requirements within defined constraints, encompassing planning, implementation, testing, and maintenance phases for successful delivery.

It can be divided into two types:

1. WATERFALL MODEL

The waterfall model is a sequential software development approach where tasks progress linearly through phases like requirements, design, implementation, testing, and maintenance, each stage dependent on the completion of the previous one.

2. AGILE MODEL

The Agile model is an iterative and flexible software development methodology emphasizing adaptability, collaboration, and customer involvement, breaking tasks into smaller increments called sprints, and allowing for continuous feedback and adaptation throughout the project lifecycle.

By having machine learning as the project effort estimation, money and time can be saved as it will need less human effort. This project will be useful for every software development company to estimate a project's effort. The investigations of software effort estimation have started since the 1960s.

OBJECTIVE

The software project management domain faces the ongoing issue of predicting project effort accurately. The gap between predictions and reality emphasizes the necessity for a dependable, data-driven approach to address these challenges.

This project focuses on the estimation of **Effort** required for fulfilling a software project. Below is a brief description of all three.

Effort in Software Projects:
Effort in software projects refers to the amount of human work, skills, and expertise required to complete a project. It encompasses the time and energy invested by developers, designers, testers, and other team members to conceptualize, design, code, test, and deliver the software product.

Following is the description of the technologies to be used for the practical implementation of the project:

Ensemble Machine Learning:
Ensemble machine learning involves combining predictions from multiple individual models to create a stronger, more accurate prediction. Standard ensemble methods include bagging, boosting, and stacking, each contributing to more robust and reliable predictions across various datasets and scenarios.

- *Stacking:* Stacking combines predictions from multiple models using a higher-level model to improve overall prediction accuracy.
- *Boosting:* Boosting iteratively enhances the predictive power of weak learners by adjusting their focus on challenging instances.
- *Bagging (Bootstrap Aggregating):* Bagging trains multiple instances of the same model on different data subsets to reduce variance and enhance stability in predictions.

Neural Networks and Metaheuristic Algorithms:
Neural networks are computational models inspired by the intricate interconnections between neurons in the human brain. They consist of layers of interconnected nodes (neurons) that process and transform input data to produce desired outputs.

Metaheuristic algorithms are optimization techniques used to solve complex problems that may lack a well-defined mathematical formulation.

LITERATURE REVIEW

Ref no.	Datasets Used	Algo Used	Evaluation Metrics Used	Result
[1]	CM1, KC1, COCOMO81, Desharnais, China, and Albrecht	Decision Tree, Principal Components Regression (PCR), Random Forest, NeurelNet, glmnet, XGBoost, Earth and Support Vector Machine	MAE, RMSE, MSE, PRED(25), R-SQUARED	In all cases, stacking using pcr produced the best results.
[2]	COCOMO81	Regression Analysis	MMRE, R^2, MSE	The linear regression model for the projects based on the use case point gave the best results.
[3]	Desharnais, ENB, EVP	MLR-DNN, PSO-DNN, GBR-DNN, RFR-DNN, XGB-DNN	RMSE, RMSLE, MAE, MMRE, MdMRE, Pred(m)	The verification results in Agile, Hybrid, and Waterfall projects indicated that the MLR-DNN model improved and significantly enhanced project effort performance and duration estimation.
[4]	ISBSG, PRIMARY DATASET (INDUSTRIAL CASE STUDIES)	SVR, Linear Regression, k-NN, XGBoost Regressor, and ANN	MMRE, MAE, MdMRE, MDAE, and PRED	To increase the efficiency and precision of software development initiatives, a heterogeneous ensemble model that combines artificial neural networks (ANN), use case points (UCP), and expert judgment (EJ) was used.
[5]	COCOMO81, China, Desharnais, Kemerer, Maxwell, and Albrecht	Linear Regression, Decision Tree Regression, Random Forest Regression, Support Vector Regression, and Neural Network Regression	MAE, MSE, RMSE	The ensembling of models outperformed other single models
[6]	ISBSG	Neural network, Support vector regression, Decision trees	RMSE, R-squared, MAE	Grid search (GS) tuning can improve the accuracy of both individual and stacking models by quickly and accurately selecting optimal parameter

				values.
[7]	Data were obtained from completed machine development projects that produced highly customized machines.	ANN	MSE, MMRE, PRED(X)	The performance of the best ANN architecture showed promising results.
[8]	China, Desharnais, Maxwell, Kitchenham, Albrecht, Kemerer, and Cocomo81	Random forest, SVM, decision tree, neural net, ridge, LASSO, elastic net, deep net algorithms, ensemble ML	RMSE, R-squared, MAE	Random Forest Based Stacked Ensemble Approach provides better results compared with single models.
[9]	China, Kemerer, Kitchenham, Maxwell, Nasa93	Regression tree (RT), neural network (NN), Support vector regression (SVR), k-nearest neighbor	RMSE, R-squared, MAE, PRED(X)	LSTM-grid search-based deep neural network outperforms all other baseline algorithms.
[10]	DESHARNAIS	Linear Regression, ANFIS	RMSE, MAE	ANFIS technique has been able to outperform the Linear Regression Model in terms of the various performance criteria
[11]	Albrecht, China, COCOMO81, Desharnais, Finnish, Kemerer, Kitchenham, Maxwell, Miyazaki, NASA18, NASA93 and Telecom	SVM (over all 4 kernels: Linear, Polynomial, RBF and Sigmoid), RF, SGB, DT, kNN, LR, NB and MLP	mean absolute error, mean square error, root mean square error, and r-squared(MMRE), (MMER), (MdMRE), and prediction accuracy	Random Forest technique followed by Decision Tree was found most stable.

METHODOLOGY

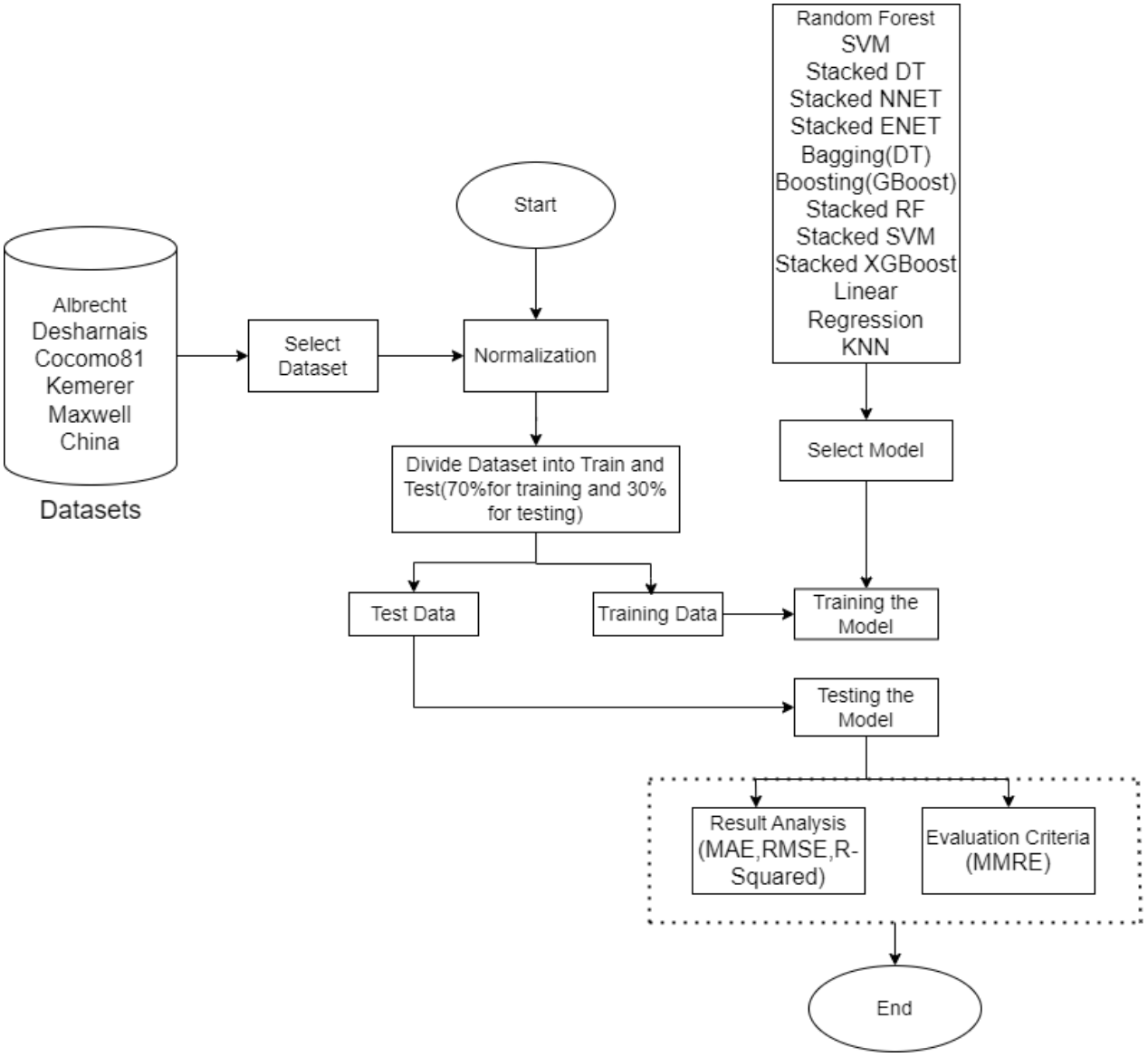


Fig. 1

5.0. Workflow:

Dataset:
A dataset is a collection of data that is used to train a machine learning model. It usually consists of input data (also known as features) and output data (also known as labels or targets) that the model is trying to predict. The input data could be in the form of text, images, audio, or any other type of data that the model can understand.

Training data:
Training data is a subset of the dataset used to train a machine learning model. It consists of input data and output data (also known as labels or targets) that the model uses to learn the underlying patterns and relationships in the data.

Test data:
Test data is a subset of the dataset that is used to evaluate the performance of a machine learning model after it has been trained on the training data. It consists of input data and output data (also known as labels or targets) that the model has not seen before.

Model building:
Model building is the process of creating a machine-learning model using a training dataset. This involves selecting an appropriate algorithm and optimizing its parameters to achieve the best performance on the training data.
Each sample has some input features(X) and one output variable (y). We use this training data to build a regression model that can predict the value of y for new input data.

Model evaluation:
Model evaluation is the process of testing a machine learning model on a separate test dataset to measure its performance. This involves computing various metrics such as accuracy, precision, recall, and F1-score to determine how well the model is performing.

Prediction result:
The prediction result is the output produced by a machine learning model when given input data. The model uses the patterns and relationships it has learned from the training data to make predictions on new, unseen data. The quality of the prediction result is evaluated by comparing it to the true output data.

Preprocessing:

Preprocessing improves the quality of the original dataset as it will enhance the performance of the predictions in the machine learning context. Dataset fed into standardization process as most of the machine learning algorithms are required to scale the numerical input variable into standard range. Scaling each input variable separately by subtracting the mean and dividing by the standard deviations to shift the distribution to have a mean of zero and a standard deviation of one is called standardization, removing null values, discarding duplicate records, and attributes were also carried out but has no null values and duplicate records or attributes in the above dataset effects were minimum.

5.1. Proposed ML Algorithm:

The paper proposed a variety of experiment considering 12 different machine learning based algorithms namely; SVM, RF, Stacked DT,Stacked Nnet,Stacked Enet,Bagging DT,Boosting GB,Stacked RF,Stacked XGB, LR and kNN over 6 different publicly available datasets; Albrecht, China, COCOMO81, Desharnais, Kemerer ,and Maxwell,widely used in SDEE community. The paper has compared the performance of these regression algorithms applied with these datasets over 4 parameters namely; Mean Absolute Error (MAE), Mean Magnitude of Relative Error (MMRE), Root Mean Squared Error (RMSE), R-Squared value.

THE PROPOSED METHODOLOGY IS AS FOLLOWS:

1. Base Models Initialization: Two XGBoost regressor models (xgb_model1 and xgb_model2) are created with 100 estimators and a random state set to 42. These models will be used as base models in the ensemble.
2. Training Base Models: Both base models are trained using the training data (X_train and y_train).
3. Making Predictions with Base Models: Predictions are made on the test data (X_test) using both base models (xgb_predictions1 and xgb_predictions2).
4. Meta-Model Initialization: Another XGBoost regressor model (meta_model) is created with 100 estimators and a random state of 42. This model will be used to combine predictions from the base models.
5. Creating Meta-Features: Meta-features are created by combining the predictions from the base models (xgb_predictions1 and xgb_predictions2) into a DataFrame (meta_features).
6. Training Meta-Model: The meta-model (meta_model) is trained using the meta-features (meta_features) and the actual test targets (y_test).
7. Making Predictions with Meta-Model: Predictions are made on the meta-features using the trained meta-model (meta_predictions).
8. Performance Evaluation: Several performance metrics are calculated to evaluate the ensemble model:

Finally, the calculated metrics (mae, rmse, r2, mmre) are printed to evaluate the performance of the ensemble model.

This approach leverages the strengths of multiple XGBoost models by combining their predictions using a meta-model, aiming to improve predictive accuracy and robustness compared to using a single model.

5.2. Datasets:

Dataset	Number of records	Number of attributes	Output Attribute-Effort (Unit)
cocomo81	63	17	Person-months
China	499	16	Person-hours
Albrecht	24	8	Person-months
Desharnais	81	12	Person-hours
Maxwell	62	27	Person-hours
Kemerer	15	7	Person-months

Datasets – Original attributes and attributes considered after Feature selection.

Datasets Name	Attributes and Description
Cocomo81	Rely Required software reliability Data Database size Time Time constraint Stor Storage constraint Acap Analyst capability Modp Modern programming practices Sced Development schedule Loc Lines of code Efforts actual Work carried in person-months
China	AFP Adjusted function points Output Function points of external output File Function points of internal logical Interface Function points of external interface Added Function points of added functions PDR_AFP Productivity delivery rate(adjusted function points) NPDR_AFP Normalised productivity delivery rate(adjusted function points) NPDU_UFP Productivity delivery rate(Unadjusted function points) N-Effort Normalised effort Effort Summary work report

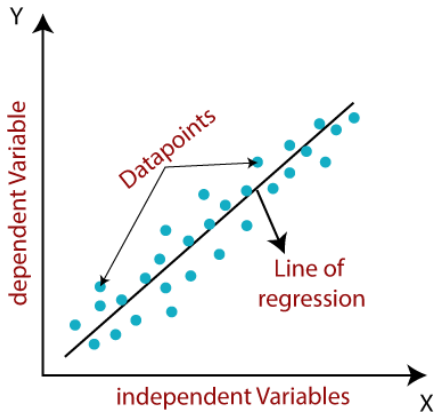
Albrecht	OutputNumeric Count of output functions InquiryNumeric Count of query functions RawFPcounsNumeric Raw function points AdjfpNumeric Adjusted function points Effort Effort in person-months
Desharnais	Project Project number TeamExp Team experience in years ManagerExp Project manager's experience in years YearEnd Year of completion Length Length of the project Transactions Number of transactions processed Entities PointsNonAdjust Unadjusted function points Adjustment factor PointsAdjust Adjusted function points Entities number of entities Adjustment adjustment factor Language Programming language Effort Measured in person-hours
Maxwell	Year Time App Application type Har Hardware platform Dba Database Ifc User interface Source Where developed Telonuse Telon use Nlan Number of development languages T01 Customer participation T02 Development environment adequacy T03 Staff availability T04 Standards use T05 Methods use T06 Tools use T07 Software logical complexity T08 Requirements volatility T09 Quality requirements T10 Efficiency requirements T11 Installation requirements T12 Staff analysis skills T13 Staff application knowledge T14 Staff tool skills T15 Staff team skills Duration Duration (months) Size Application size (FP) Time Time taken Effort Work carried out in person-hours

Kemerer	Language	Programming language Hardware
	Hardware	resources
	Duration	Duration of the project
	KSLOC	Kilo lines of code
	AdjFP	Adjusted function points Raw
	FP	Unadjusted function points
	Effort	Measured in person-months

5.3. Algorithms

5.3.0. Linear Regression:

Linear regression is a widely used machine learning algorithm for predictive analysis. It establishes a linear relationship between a dependent variable (e.g., sales, salary) and one or more independent variables. The goal is to find the best-fit line, minimizing the error between predicted and actual values. This is achieved by determining optimal weights or coefficients (a0, a1) through a cost function, ensuring the linear model accurately represents the data. The linear regression model provides a straight line representing the relationship between variables.
The linear regression model provides a sloped straight line representing the relationship between the variables. Consider the below image.

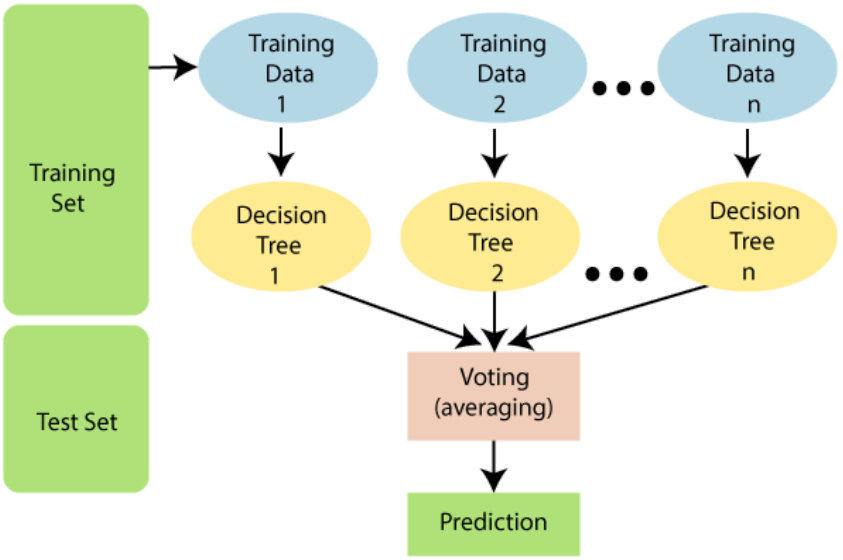


5.3.1. Random Forest Technique:

"Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, it predicts the final output.

Greater trees in the forest lead to higher accuracy & prevent the problem of overfitting.

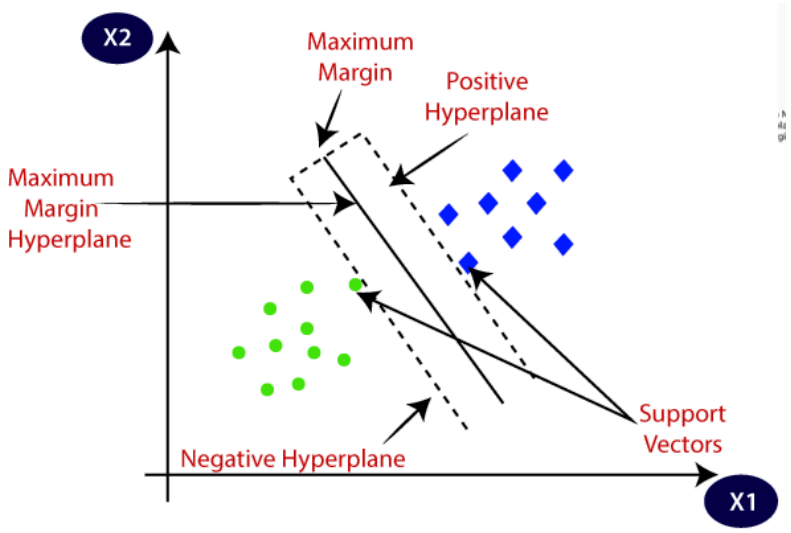
The below diagram explains the working of the Random Forest algorithm:



5.3.2. Support Vector Machine:

Support Vector Machine (SVM) is a widely-used Supervised Learning algorithm for both Classification and Regression tasks, although its primary application is in Classification in Machine Learning. SVM aims to create an optimal decision boundary or hyperplane in an n-dimensional space, effectively segregating the space into classes. This hyperplane is crucial for accurately categorizing new data points in the future.

SVM identifies the key points or vectors at the extremes that contribute to defining the hyperplane. These critical instances are known as support vectors, giving the algorithm its name. The goal is to select support vectors that assist in establishing the best decision boundary. The diagram below illustrates how SVM classifies two different categories using a decision boundary or hyperplane:



Hyperplane: There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the best decision boundary that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimensions of the hyperplane depend on the features present in the dataset, which means if there are 2 features (as shown in the image), then the hyperplane will be a straight line. And if there are 3 features, then the hyperplane will be a 2-dimension plane.

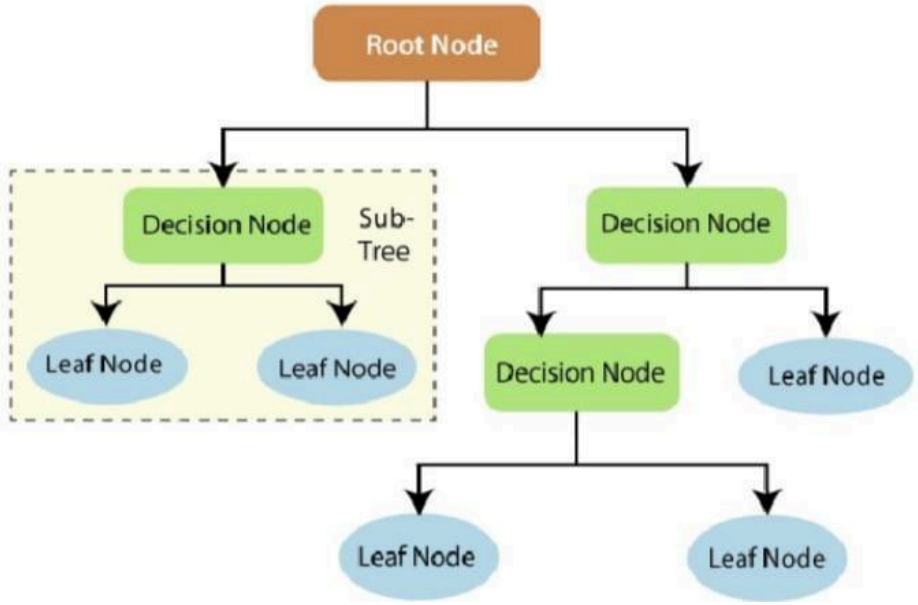
We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors:

The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vectors. These vectors support the hyperplane, hence called a Support vector.

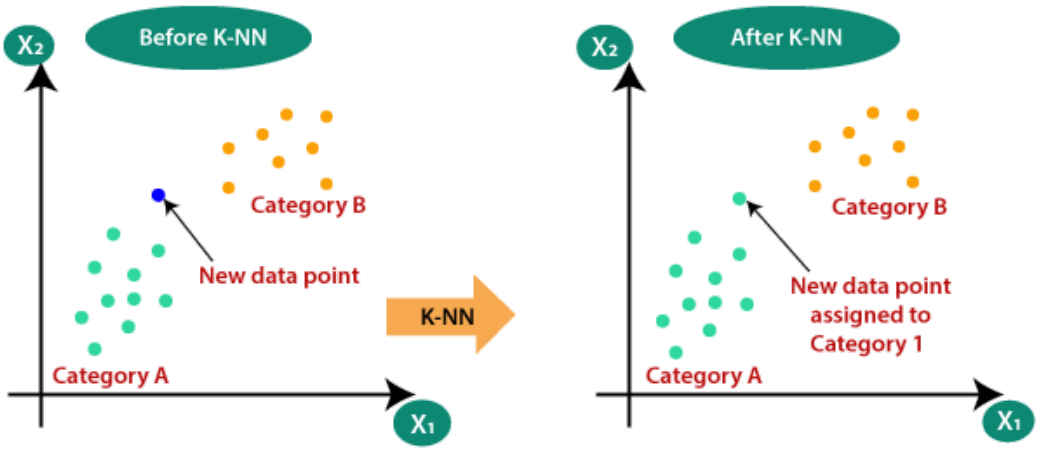
5.3.3. Decision Tree Regression:

A Decision Tree is a supervised machine learning algorithm used for classification and regression. It recursively splits the dataset based on the most informative features, creating a tree structure. The root node represents the entire dataset, and each internal node represents a decision based on a feature. The process continues until a stopping criterion is met. Leaf nodes provide the final output, whether a class (for classification) or a predicted value (for regression).



5.3.4. KNN(K-Nearest Neighbor):

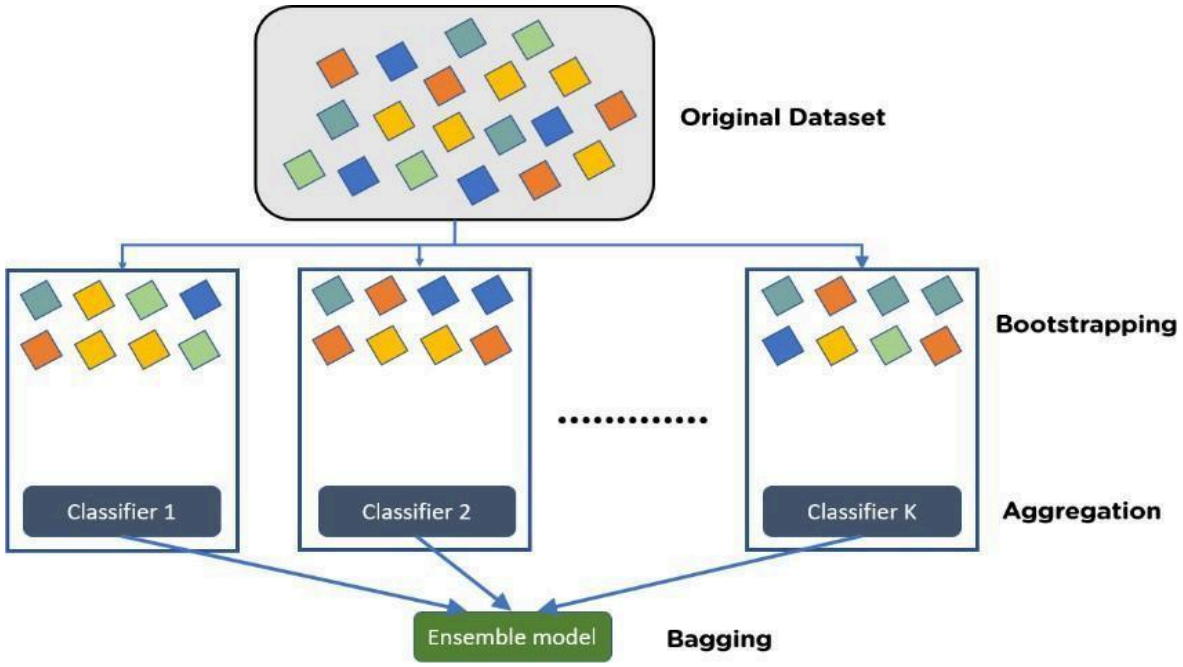
The K Nearest Neighbor algorithm falls under the Supervised Learning category and is used for classification (most commonly) and regression. It is a versatile algorithm also used for imputing missing values and resampling datasets. As the name (K Nearest Neighbor) suggests it considers K Nearest Neighbors (Data points) to predict the class or continuous value for the new Datapoint. Suppose there are two categories, i.e., Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset. Consider the below diagram:



5.3.5. Bagging

Bagging, also known as Bootstrap aggregating, is an ensemble learning technique that helps to

improve the performance and accuracy of machine learning algorithms. It is used to deal with bias-variance trade-offs and reduces the variance of a prediction model. Bagging avoids overfitting of data and is used for both regression and classification models, specifically for decision tree algorithms.

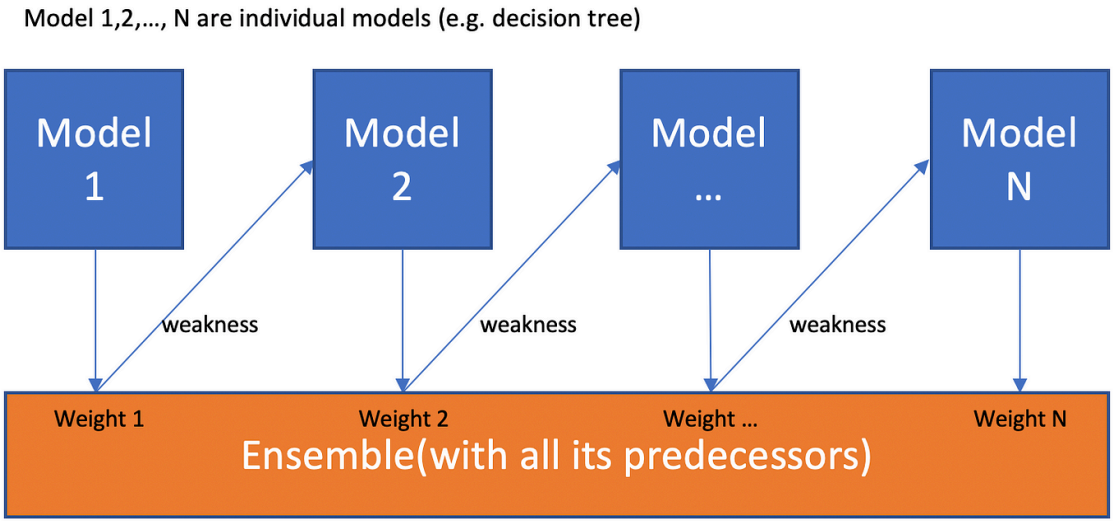


Steps to Perform Bagging

- Consider there are n observations and m features in the training set. You need to select a random sample from the training dataset without replacement
- A subset of m features is chosen randomly to create a model using sample observations
- The feature offering the best split out of the lot is used to split the nodes
- The tree is grown, so you have the best root nodes
- The above steps are repeated n times. It aggregates the output of individual decision trees to give the best prediction

5.3.6. Boosting

is an ensemble modeling technique that attempts to build a strong classifier from the number of weak classifiers. It is done by building a model by using weak models in series. Firstly, a model is built from the training data. Then the second model is built which tries to correct the errors present in the first model. This procedure is continued and models are added until either the complete training data set is predicted correctly or the maximum number of models are added.



The boosting techniques used in this project are as follows:

1.GradientBoost:

Gradient Boosting is a machine learning algorithm used for regression and classification tasks. It combines weak learners into strong learners, where each new model is trained to minimize the loss function of the previous model. It is a flexible algorithm that allows for the use of arbitrary differentiable loss functions, making it applicable to a wide range of problems. It is a stage-wise additive model that can be used for classification tasks and is implemented in several libraries, including XGBoost, LightGBM, CatBoost, and Scikit-learn. It is a powerful and versatile algorithm for tabular supervised learning tasks.

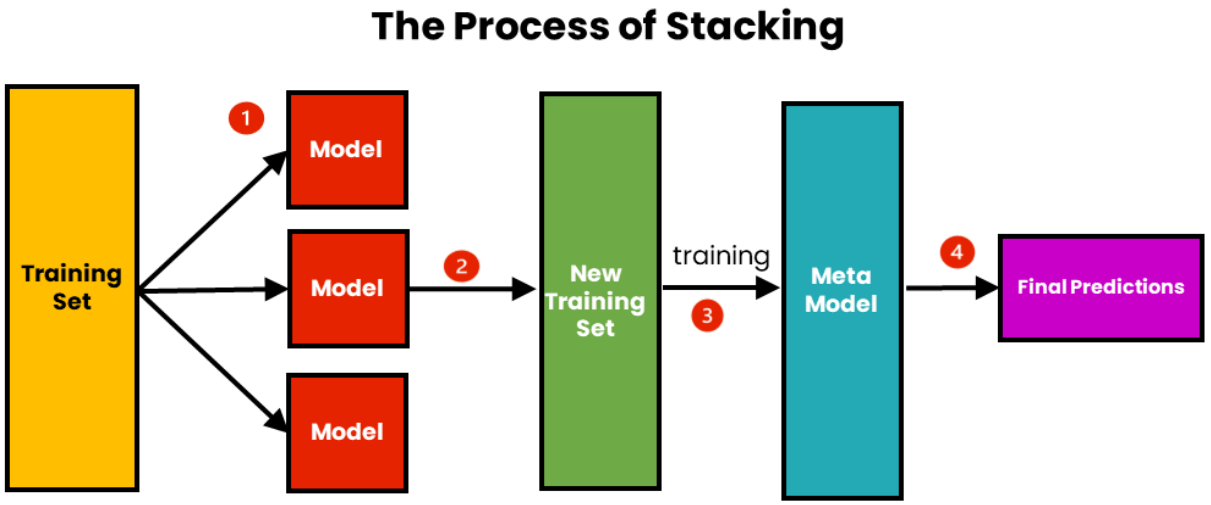
2.XGBoost:

XGBoost, short for Extreme Gradient Boosting, is a popular and powerful machine learning algorithm known for its speed and performance. It is an optimized implementation of the Gradient Boosting algorithm and is designed to be highly efficient and scalable. XGBoost uses a technique called gradient boosting to iteratively build a strong predictive model by combining multiple weak models. It is widely used in various machine learning competitions and real-world applications due to its ability to handle large datasets, its flexibility in defining custom optimization objectives, and its support for parallel processing. XGBoost is implemented in several programming languages, with the Python version being particularly popular due to its ease of use and integration with other data science libraries.

5.3.7. Stacking:

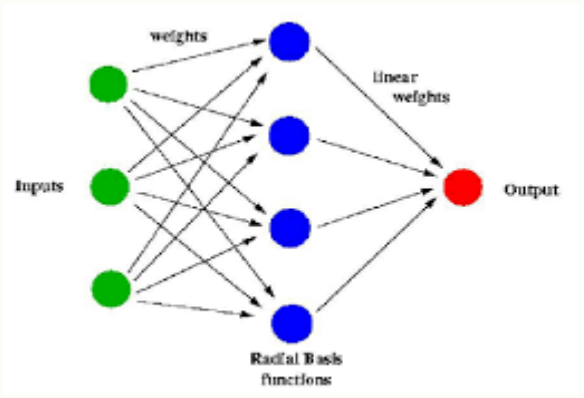
Stacking, or Stacked Generalization, is an ensemble learning technique in machine learning where multiple models (base models) are trained to make predictions, and then a meta-model is trained to make predictions based on the outputs of those base models. The idea is to combine the strengths of diverse models to improve overall predictive performance. Base models may include various algorithms, and the meta-model is trained on their predictions to provide a final output. Stacking aims to enhance model generalization and robustness by leveraging the

complementary strengths of different models.



5.3.8.Neural Networks(NNet):

Neural network algorithms are a type of machine learning model inspired by the structure and function of the human brain. A neural network consists of interconnected layers of nodes, or "neurons," which process and transmit information. During training, the algorithm adjusts the weights of these connections to minimize the difference between the network's predictions and the actual data. Neural networks have been successful in a wide range of applications, including image and speech recognition, natural language processing, and forecasting. However, they can be computationally intensive and require large amounts of data to train effectively. Despite these challenges, neural networks remain a powerful tool for solving complex problems in artificial intelligence.



5.3.9.Elastic Net(ENet):

The Elastic Net algorithm is a regularization technique used in linear regression that combines Lasso and Ridge regression to prevent overfitting and improve predictive power. It is implemented in the LARS-EN algorithm and can automatically select the appropriate penalty to add to the model. The algorithm is particularly useful when the majority of features are relevant or irrelevant, and can lead to dimensionality reduction and improved computational efficiency. It is implemented in various software packages and can be used in combination with strong rules and the KKT conditions to ensure the exact solution is obtained. Overall, the Elastic Net algorithm is a powerful tool for improving model performance and reducing computational complexity in linear regression.

6. Evaluation Criteria:

6.0. Mean Absolute Error: It is the measure of errors between paired observations expressing the same phenomenon and the way to measure the accuracy of a given model. *Take the difference between your model's predictions and the ground truth, square it, and average it out across the whole dataset.*The formula to it is as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |Y_i - \hat{Y}_i|$$

Where as ,

***pi* =** actual value,
***yi* =** predicted value,
n = number of observation

6.1. RMSE or Root Mean Squared Error: it is a general-purpose error estimation that is calculated by computing the square root of the summation of the square of the difference of the prediction of an experiment and its actual/expected value. RMSE is a good error estimation as it tells us how far a line is fit from its actual value. Fields such as Machine Learning and Numerical Analysis have extensive requirements if this measure of error calculation:

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - p_i)^2}$$

Where as ,
***di* =** predicted value,
***pi* =** actual value,
n = number of observation

6.2. **R-squared:** R-squared (R2) is an important statistical measure. A regression model represents the proportion of the difference or variance in statistical terms for a dependent variable that an independent variable or variables can explain. In short, it determines how well the data will fit the regression model.

r = n (∑xy) – ∑x ∑y / √ [n* (∑x2 – (∑x)2)] * [n* (∑y2 – (∑y)2)]

- r = The Correlation coefficient
- n = number in the given dataset
- x = first variable in the context
- y = second variable

6.3. **MMRE**: The Magnitude of Error Relative (MER) values are utilized to find the value of error relative which is determined by the following equation

$$MMER = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|ActualEffort - PredictedEffort|}{PredictedEffort}\right)_i \text{ or } MMER = \frac{1}{n}\sum_{i=1}^{n}(MER)_i$$

7. Results and Discussion:

For effort prediction, the datasets considered were Albrecht, China, Desharnais and Cocomo81. The evaluation measures considered were Mean Absolute Error (MAE), Mean Squared Error (MAE) Root Mean Square Error (RMSE), MMRE, and R-squared value. It was found that the lesser the values of MAE, MMRE and RMSE, the better the model; additionally, if the R-squared value was closer to 1, it was the better model. Various ensemble techniques available were bagging, boosting, and stacking. Single models considered for comparison were random forest, SVM, decision tree,

Below are the tables of MAE,MSE,RMSE and R-SQUARED containing the results obtained by applying different algorithms to all the datasets to make it easy to analyze which algorithm is giving the better results when compared to others.

1. Comparative analysis of performance of ML algorithms over ALBRECHT Dataset

Models	**MAE**		**RMSE**		**R2 SCORE**		**MMRE**	
	Our values	Base Paper Values	Our values	Base Paper Values	Our values	Base Paper Values	Our values	Base Paper Values
RF	0.0553	0.0425	0.0607	0.526	0.87	0.9202	0.609	0.2155
SVM	0.089	0.0461	0.1158	0.0631	0.1869	0.8852	0.8894	-1.5324
Stacked DT	0.075	0.1032	0.1035	0.1679	0.3509	0.1877	0.7594	0.278
Stacked Nnet	0.1795	-	0.2183	-	0.7908	-	1.5746	-
Stacked Enet	0.122	-	0.1409	-	0.667	-	2.2419	-
Bagging DT	0.0418	-	0.0519	-	0.667	-	1.0051	-
Boosting GB	0.0468	-	0.0593	-	0.9763	-	0.7264	-
Stacked RF	0.077	-	0.1258	-	0.8938	-	0.6858	-
Stacked SVM	0.0683	-	0.0899	-	0.9456	-	0.9136	-
Stacked XGB	0.0009	-	0.00109	-	0.9999	-	0.0001	-
LR	0.0415	0.0632	0.0445	0.0698	-0.677	0.8578	2.2829	1.4789
KNN	0.0622	0.0958	0.0845	0.1571	0.775	0.2892	1.0051	0.72

2. Comparative analysis of performance of ML algorithms over CHINA Dataset

Model	**MAE**		**RMSE**		**R2 SCORE**		**MMRE**	
	Our values	Base Paper values	Our Values	Base Paper values	Our values	Base Paper values	Our values	Base Paper values
RF	0.1195	0.0204	2.347	0.0679	0.8885	0.7453	1.6477	0.1033
SVM	0.823	0.0441	4.943	0.0748	-0.16	0.6914	1.1081	0.168
Stacked DT	0.086	0.0366	0.693	0.0807	0.98	0.5011	0.1831	0.5456
Stacked Nnet	0.705	-	5.101	-	-0.08	-	3.5149	-
Stacked Enet	0.8	-	4.992	-	-0.03	-	3.9103	-
Bagging DT	0.071	-	0.6378	-	0.9831	-	0.0949	-
Boosting GB	0.058	-	0.467	-	0.9907	-	0.1115	-
Stacked RF	0.072	-	0.626	-	0.9836	-	0.0949	-
Stacked SVM	0.051	-	0.5355	-	0.988	-	0.0649	-
Stacked XGB	0.0009	-	0.0046	-	0.98	-	0.0046	-
LR	0.887	0.0433	5.872	0.0099	0.9911	0.4531	0.1058	0.2739
KNN	0.084	0.0332	0.7504	0.0063	0.98	0.6541	0.0983	0.3264

3. Comparative analysis of performance of ML algorithms over CHINA Dataset

Models	MAE		RMSE		R2 SCORE		MMRE	
	Our Values	Base Paper values	Our Values	Base Paper values	Our values	Base Paper values	Our values	Base Paper values
RF	0.023	0.1085	0.0392	0.1539	0.456	0.5195	0.5443	0.4915
SVM	0.021	0.09	0.0302	0.1148	-0.04	0.7325	0.8967	0.6038
Stacked DT	0.025	0.1039	0.0354	0.1719	0.4032	0.4005	0.5146	0.6182
Stacked Nnet	0.028	-	0.0344	-	-0.323	-	0.5119	-
Stacked Enet	0.0316	-	0.036	-	0.11	-	0.388	-
Bagging DT	0.0213	-	0.0289	-	0.0685	-	0.5811	-
Boosting GB	0.0185	-	0.0239	-	0.3611	-	0.4737	-
Stacked RF	0.0218	-	0.0278	-	0.137	-	0.5152	-
Stacked SVM	0.0207	-	0.02609	-	0.242	-	0.5947	-
Stacked XGB	0.0011	-	0.0013	-	0.9999	-	0.000000474	-
LR	0.0173	0.0896	0.0236	0.1174	-0.52	0.7203	0.7512	0.5138
KNN	0.0286	0.0994	0.0462	0.1686	0.694	0.4232	0.7768	0.4118

4. Comparative analysis of performance of ML algorithms over MAXWELL Dataset

MODELS	MAE		RMSE		R2 SCORE		MMRE	
	Our values	Base Paper values	Our values	Base Paper values	Our values	Base Paper values	Our values	Base Paper values
RF	0.266	0.026	0.3632	0.0394	0.458	0.9058	1.252	1.4791
SVM	0.2305	0.0501	0.2807	0.0625	-0.116	0.7633	1.2079	0.3173
Stacked DT	0.206	0.0574	0.2239	0.1057	0.289	0.3239	0.8675	0.5614
Stacked Nnet	0.2921	-	0.4031	-	-1.303	-	0.9361	-
Stacked Enet	0.391	-	0.5096	-	-2.68	-	2.3775	-
Bagging DT	0.183	-	0.2524	-	0.096	-	0.9033	-
Boosting GB	0.3462	-	0.543	-	0.6751	-	0.454	-
Stacked RF	0.3877	-	0.5932	-	0.6123	-	0.8996	-
Stacked SVM	0.456	-	0.6781	-	0.4935	-	0.9887	-
Stacked XGB	0.0021	-	0.0036	-	0.9999	-	0.0000004307	-
LR	0.592	0.1469	0.933	0.2307	0.67	-2.2231	1.3819	1.8998
KNN	0.6074	0.0628	1.107	0.085	0.41	0.5626	0.6948	0.5894

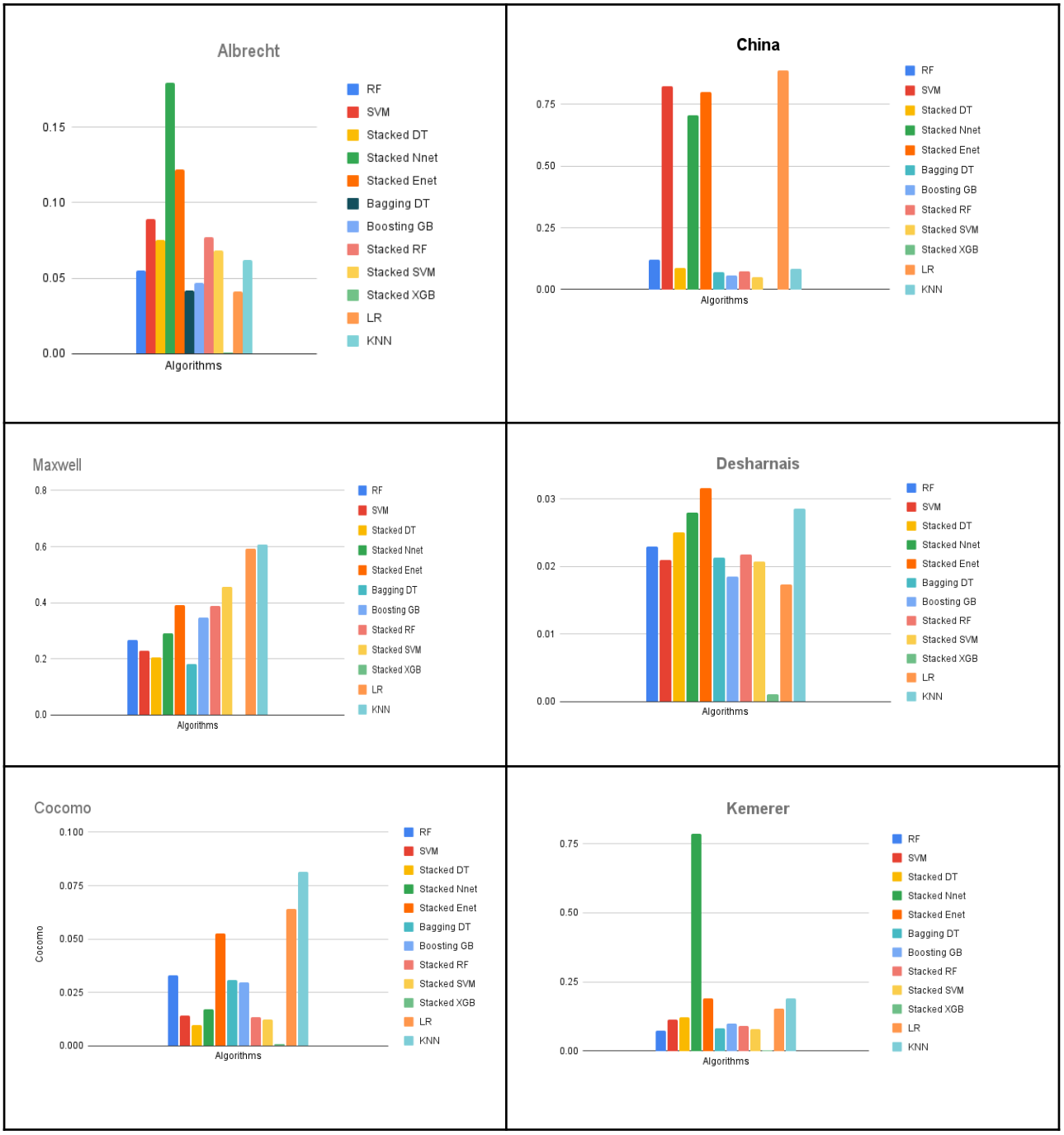
5. Comparative analysis of performance of ML algorithms over KEMERER Dataset

MODELS	MAE		RMSE		R2 SCORE		MMRE	
	Our values	Base Paper values	Our values	Base Paper values	Our values	Base Paper value	Our values	Base Paper values
RF	0.073	0.0487	0.07398	0.0567	0.4723	0.3109	1.874	0.3123
SVM	0.113	0.0668	0.1217	0.0778	-0.52	0.5635	3.123	0.9447
Stacked DT	0.121	0.0801	0.1329	0.0911	-8.14	-0.1736	1.2851	0.2071
Stacked Nnet	0.788	-	0.8569	-	-3.78	-	2.422	-
Stacked Enet	0.189	-	0.1959	-	-1.88	-	2.2385	-
Bagging DT	0.082	-	0.1036	-	0.05	-	1.4065	-
Boosting GB	0.099	-	0.11304	-	0.559	-	1.2894	-
Stacked RF	0.09	-	0.1037	-	0.16	-	1.5393	-
Stacked SVM	0.079	-	0.0851	-	0.03	-	1.4821	-
Stacked XGB	0.0009	-	0.00099	-	0.999	-	0.00002541	-
LR	0.1537	0.2723	0.1837	0.326	-0.52	0.6322	2.2461	1.1189
KNN	0.1909	0.0423	0.386	0.0436	0.01	0.5629	1.727	0.4214

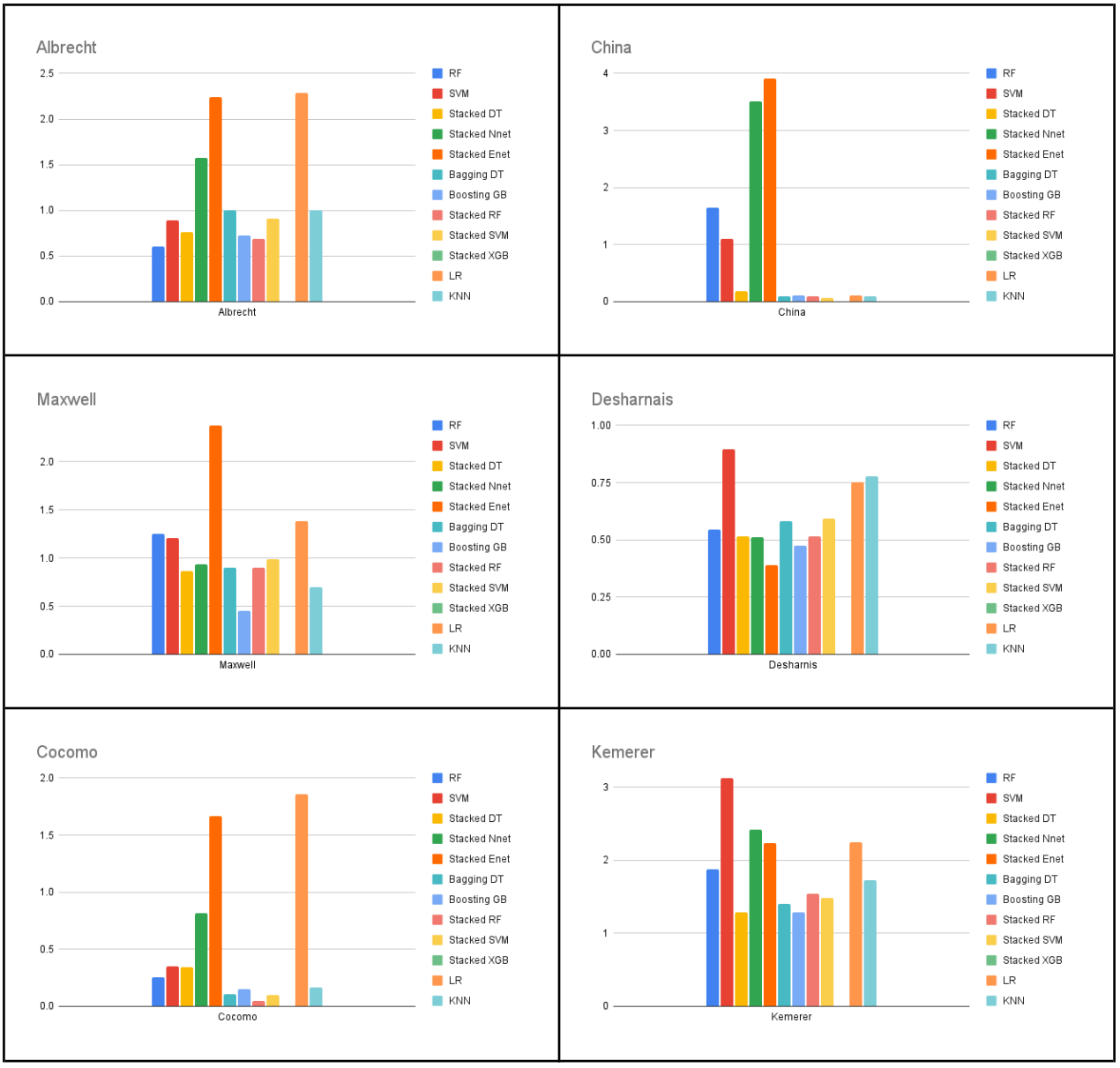
6. Comparative analysis of performance of ML algorithms over COCOMO81 Dataset

MODELS	MAE		RMSE		R2 SCORE		MMRE	
	Our values	Base Paper values	Our values	Base Paper values	Our values	Base Paper values	Our values	Base Paper values
RF	0.033	0.0247	0.0475	0.0553	0.304	-2.7263	0.2533	1.7131
SVM	0.0143	0.0478	0.0212	0.0605	-0.0011	-3.46	0.3493	14.8331
Stacked DT	0.0096	0.0168	0.0145	0.0228	0.532	0.3644	0.3399	1.6067
Stacked Nnet	0.017	-	0.026	-	0.78	-	0.8159	-
Stacked Enet	0.0526	-	0.0639	-	-0.342	-	1.6681	-
Bagging DT	0.031	-	0.0554	-	-0.1011	-	0.1077	-
Boosting GB	0.0299	-	0.0554	-	0.0059	-	0.1482	-
Stacked RF	0.0135	-	0.0242	-	0.8101	-	0.0455	-
Stacked SVM	0.0123	-	0.0193	-	0.8787	-	0.0986	-
Stacked XGB	0.00091	-	0.00112	-	0.999	-	0.00000386	-
LR	0.064	0.1002	0.0722	0.138	-2.68	-22.2084	1.8603	18.3929
KNN	0.0815	0.0154	0.2021	0.03	0.43	-0.0955	0.164	1.4125

MEAN ABSOLUTE ERROR (MAE)



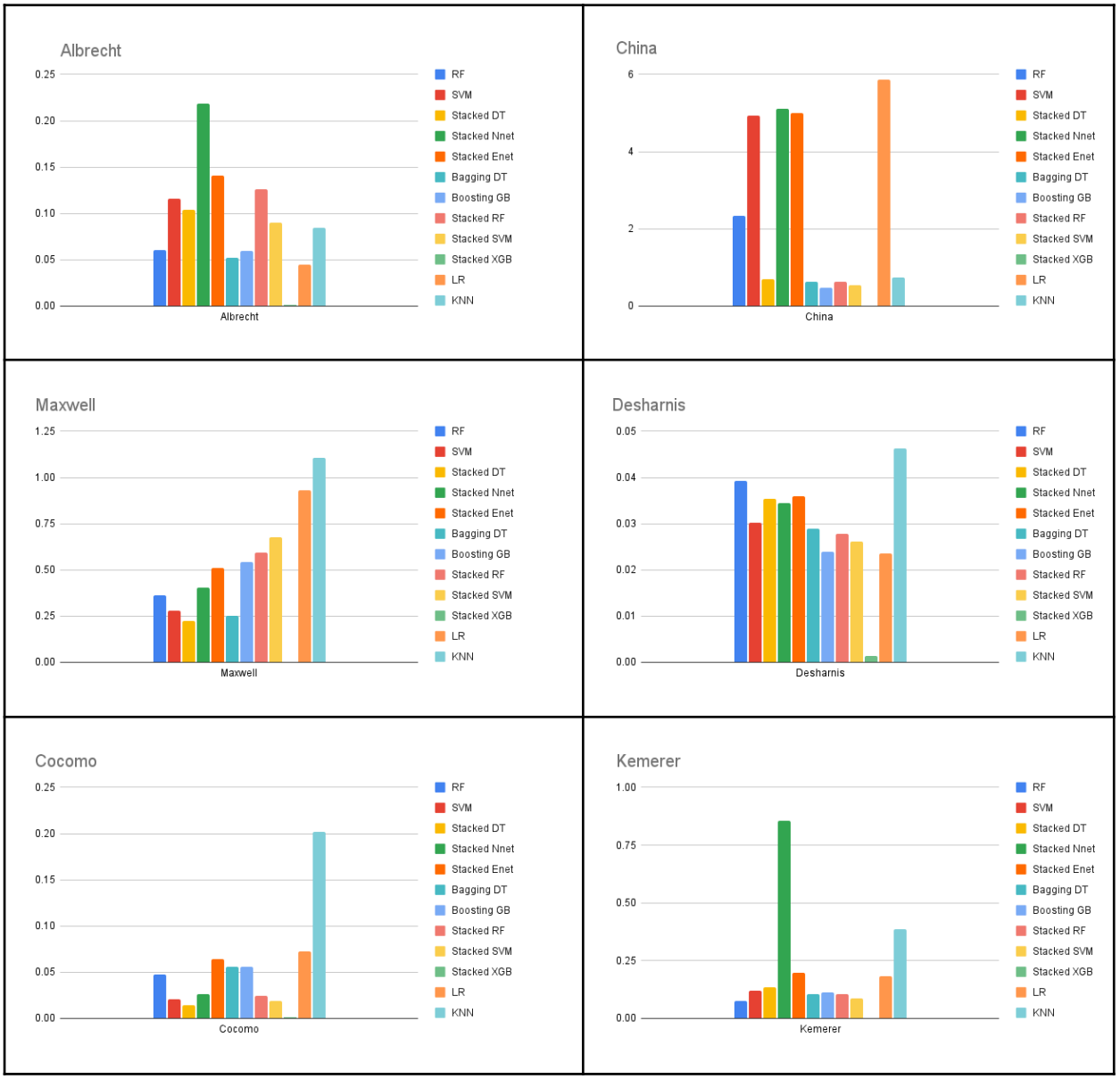
MEAN ABSOLUTE PERCENTAGE ERROR(MMRE)



R-SQUARED(R2)


Albrecht
1.0
0.5
0.0
-0.5
-1.0
Albrecht
RF
SVM
Stacked DT
Stacked Nnet
Stacked Enet
Bagging DT
Boosting GB
Stacked RF
Stacked SVM
Stacked XGB
LR
KNN
China
1.00
0.75
0.50
0.25
0.00
-0.25
China
Maxwell
1
0
-1
-2
-3
Maxwell
Desharnais
1.0
0.5
0.0
-0.5
-1.0
Desharnis
Cocomo
1
0
-1
-2
-3
Cocomo
Kemerer
2
0
-2
-4
-6
-8
-10
Kemerer

ROOT MEAN SQUARED ERROR(RMSE)



CONCLUSION AND FUTURE WORK:

Effort estimation is recognized as one of the most critical activities for the success of the overall solution delivery in software development process. Software Development Effort Estimation (SDEE) predicts the efforts to develop any software project in terms of persons-month or person-hours. The paper provides a comparative analysis to evaluate the impact of different machine learning based regression algorithms when used with a variety of datasets, over different parameters of SDEE.
As per the proposed,Stacked XGBoost model which significantly provided better results in terms of different metrics. The analysis specifies that the Stacked XGBoost algorithm performs better effort prediction in the majority of experiments. To conclude we believe that our analysis here highlighted the stability achieved by Stacked XGBoost in almost every case.

In the future we intend to perform further analysis, and will try to increase the performance of various estimation techniques, by implementing more robust and optimized models.Further we intend to perform some feature selection and feature extraction,along with optimization techniques like hyperparameter tuning. We also intend to apply some more complex ensemble learning techniques to improve the accuracy. Since the problem is regression type, we can also apply discretization to convert the regression type problem into classification type.

REFERENCES

[1] Priya Varshini A G, Anitha Kumari K," Software Effort Estimation using Base Ensembled Regression Techniques and Principal Components Regression as Super Learner".(2023)

[2] Amrita Sharma, Neha Chaudhary," Prediction of Software Effort by Using Non-Linear ' Power Regression for Heterogeneous Projects Based on Use case Points and Lines of code" .(2023)

[3] Der-Jiun Pang," Hybrid Machine Learning Model Performance in IT Project Cost and Duration Prediction".(2023)

[4] Syed Sarmad Ali, Jian Ren, Kui Zhang, Ji Wu, Chao Liu," Heterogeneous Ensemble Model to Optimize Software Effort Estimation Accuracy" (2023)

[5] Beesetti Kiran Kumar, Saurabh Bilgaiyan, Bhabani Shankar Prasad Mishra," Software Effort Estimation through Ensembling of Base Models in Machine Learning using a Voting Estimator" (2023)

[6] Taher Labidi, Zaineb Sakhrawi, "On the value of parameter tuning in stacking ensemble model for software regression test effort estimation" (2023)

[7] Burcu Sengüne¸ Nursel Öztürk "An Artificial Neural Network Model for Project Effort Estimation" (2023)

[8] Priya Varshini A G, Anitha Kumari K, Vijayakumar Varadarajan," Estimating Software Development Efforts Using a Random Forest-Based Stacked Ensemble Approach" (2021)

[9] Robert Marco, Sharifah Sakinah Syed Ahmad, Sabrina Ahmad, "An Improving Long Short Term Memory-Grid Search Based Deep Learning Neural Network for Software Effort Estimation" (2023)

[10] Dheeraj Kapoor, R. K. Gupta, "Software Cost Estimation using Artificial IntelligenceTechnique" (2016)

[11] Akshay Jadhav , Shishir Kumar Shandilya ,"Reliable machine learning models for estimating effective software development efforts: A comparative analysis"(2023)