Improving Software Effort Estimation Accuracy with a Kalman Filter-Driven Ensemble Model

Beatrice O. Akumba
Department of Mathematics/Computer Science
Benue State University
Makurdi, Benue State, Nigeria

Iorshase Agaji
Department of Computer Science
Joseph Sarwuan Tarka University
(Formerly University of Agriculture)
Makurdi, Benue State, Nigeria

Nachamada V. Blamah
Department of Computer Science
University of Jos
Jos, Plateau State

Emmanuel Ogala
Department of Computer Science
Joseph Sarwuan Tarka University
(Formerly University of Agriculture)
Makurdi, Benue State, Nigeria

Barnabas T. Akumba
HISP Nigeria, Abuja.

Samera U. Otor
Department of Mathematics/Computer Science
Benue State University
Makurdi, Benue State, Nigeria

ABSTRACT

Software effort estimation requires the determination of one or more of the following estimates; effort (usually in person-months), project duration (in calendar time) and cost (in money). The ability to accurately estimate software project effort is essential for successful project planning, budgeting, and execution. This paper focuses on the development of an ensemble stacking model to enhance software effort estimation accuracy. The integration of a Kalman Filter (KFA) with various machine learning techniques, the model offers an improvement over traditional single-model approaches. Datasets of Albrecht, China, Cocomo81, Desharnais, Kemerer, and Maxwell were used for the model training and evaluation. Performance metrics of Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and R-Squared values were employed to validate the model. The results demonstrated a notable improvement in estimation accuracy, particularly in larger datasets, as compared to established models like the ensemble voting model. We made recommendations on the incorporation of additional datasets and hyper-parameter optimization to further enhance the model's performance.

General Terms

Software Engineering, Machine Learning, Software Effort Estimation, Software Project Management.

Keywords

Ensemble Stacking, Effort Estimation, Kalman Filter, Software Project.

1. INTRODUCTION

Software effort estimation plays a critical role in project management by determining the resources, time, and budget required to complete the software projects. Accurate effort estimation helps mitigate risks associated with under-budgeting, resource misallocation, and missed deadlines, which can negatively impact project outcomes. Traditional effort estimation models of COCOMO and Function Point Analysis (FPA), have often been limited by inaccuracies due to their reliance on algorithmic approaches that cannot effectively capture the complexity and uncertainty inherent in modern software development.

Recent advancements in machine learning have introduced new opportunities for enhancing the accuracy of effort estimation models. Specifically, ensemble learning, which combines the strengths of multiple models, has shown promise in improving predictive performance. However, while techniques like ensemble voting have provided some benefits, there remains a need for models that can handle noise and uncertainty in data more effectively. This paper addresses this gap by developing an ensemble stacking model that integrates the Kalman Filter algorithm (KFA) with machine learning techniques for improved effort estimation. The objectives of the paper include the design, implementation, and evaluation of this model to assess its accuracy across some selected datasets.

2. RELATED WORKS

Software effort estimation has been a central focus of software engineering since the 1960s, when early models like the Constructive Cost Model (COCOMO) were introduced. These models aimed to estimate the time, effort, and cost required to complete software projects based on key variables such as the number of Source Lines of Code (SLOC) and project complexity. Although these traditional models, like COCOMO and Function Point Analysis (FPA), have been widely adopted, they face some limitations as they inaccurately estimate software effort due to the dynamic and non-linear nature of modern software projects. These models tend to struggle with over fitting to historical data or under fitting when they fail to capture all the sensitivities of new software projects [1].

According to [2], as software projects have grown in complexity, non-algorithmic and learning-based models have been introduced to improve the accuracy of effort estimations. The non-algorithmic models, also known as non-parametric models, estimate effort by leveraging on analogy or expert judgment. For instance, expert judgment relies on insights from experienced project managers, while analogy-based estimation involves comparing a current project to similar past projects. However, these approaches are limited by subjective biases and can result in inconsistent estimates across different experts or organizations. Similarly, analogy-based models face challenges when no closely analogous projects exist in historical data, or the quality of past records is poor [3].

Machine learning-based methods have become increasingly popular for their ability to handle large datasets and uncover

patterns that traditional models might overlook. Among these are Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), and Regression Trees, which use historical project data to train predictive models. These models are particularly good at discovering complex relationships between project attributes (such as team size, technical requirements, and project deadlines) and the associated effort. [4] used a combination of ANN, SVM, and regression models to improve effort estimation accuracy across a large dataset of software projects. Similarly, [5] explored machine learning techniques like Naïve Bayes, Logistic Regression, and Random Forests, noting that, these approaches provided more reliable predictions than traditional models.

However, these machine learning methods also come with some challenges. According to [6], one key issue is the tendency to over fit the training data, leading to poor generalization to new, unseen projects. [7] added that most machine learning algorithms are sensitive to noise in the data, which led to inaccurate predictions. This is where ensemble learning; a technique that combines multiple predictive models comes to play. Ensemble methods, such as bagging and boosting, seek to combine the strengths of individual models while reducing their weaknesses. Boosting focuses on correcting the errors of weak models by training successive models to focus on misclassified instances, while bagging reduces variance by averaging the predictions of several base models trained on different data subsets [8].

Stacking, a more advanced form of ensemble learning, takes this concept further by combining the predictions of multiple base models using a meta-model. [9] demonstrated that, stacking significantly improved the accuracy of effort estimation compared to the single-model approaches. In their study, stacking was used to combine multiple regression models, outperforming single models across diverse datasets. The advantage of stacking lies in its ability to mitigate the biases of individual models by learning from their combined predictions.

Despite the success of stacking, challenges still remain. Specifically, noise in datasets and the uncertainty associated with software project attributes; such as changing requirements and unpredictable risks could still lead to inaccurate estimates. Kalman Filter Algorithms (KFA) offers a promising solution to this problem. The Kalman Filter is a recursive algorithm that reduces noise by adjusting predictions based on real-time data, which makes it an excellent tool for refining effort estimates in dynamic environments. [10] applied Kalman Filter to software effort estimation and demonstrated that it could improve prediction accuracy by smoothing noisy datasets and accounting for unforeseen project changes. By integrating the Kalman Filter into an ensemble stacking model, this paper builds on the strengths of previous work while addressing key limitations related to noise and uncertainty in software project data. The developed model combined multiple machine learning algorithms with Kalman Filter Algorithm to improve predictive accuracy, particularly in datasets with high variability or incomplete information.

3. METHODOLOGY

3.1 Mixed-Mode Methodology

The mixed-mode methodology was employed in this work which comprised; the interactive waterfall model, mathematical model and machine learning models. The interactive waterfall model of the verification and validation (V&V) method was used for the user interface development as seen in Figure 1. The mathematical model was formulated along with the architecture as well as the flowchart of the ensemble stacking model in Figures 2 and 3 respectively. The model development and its implementation are shown in Figure 4 with the datasets used, the machine learning algorithms employed, model evaluation and the performance metrics explained in the sub-sections of the paper.

The interactive waterfall model phases employed in the user interface design are;

(i) Phase 1 is the Requirements Gathering Phase. Here, the problem was defined to be the development of a hybrid model for software project effort estimation. The objective was to predict the actual and predicted effort, completion time of the software projects. The scope of the project combined existing machine learning algorithms to achieve improved software effort prediction accuracy. The data collection and understanding was online (Zenodo) data sources for training the machine learning model. The six datasets used were; Albrecht, China, Cocomo, Desharnais, Kemerer and Maxwell datasets. These were preprocessed using MinMaxScaler and Recursive Feature Elimination (RFE). The metrics used to evaluate the models performance were MAE, MSE, RMSE, R-Squared, Friedman's Test and Wilcoxon's Rank Sum Tests.

(ii) Phase 2 is the design of the system which comprised; the algorithm selection, feature engineering and training of the model. For the Algorithm Selection, we used regression- based machine learning algorithms of Random Forest Regressor, Gradient Boosting Regressor, SVR (Support Vector Regressor), DecisionTree Regressor, MLPRegressor (Multi-layer Perceptron Regressor) and Kalman Filter. A meta-model for stacking using Linear Regression was developed afterwards.

The Feature Engineering was done to identify relevant features from the dataset that were used as inputs to the machine learning model. Data preprocessing with MinMaxScaler and Pearsons Correlation Coefficient such as normalization, encoding categorical variables, handling missing values were performed.

In the model training, the dataset was split into the training and validation sets using the ratio of 70:30 respectively. The selected machine learning algorithms were trained using the training data and validated the developed model using the validation sets data and tuned the hyper-parameters as necessary.

(iii) Phase 3 is the Implementation of the model, its evaluation and integration. For the Model Evaluation, the performance of the trained model was evaluated using the metrics MAE, MSE, RMSE, R-Squared, Friedman's Test and Wilcoxon's Rank Sum Tests. Different models were compared and the best-performing one was selected based on the evaluation results. The performance metrics values were compared with an existing ensemble voting model of [11]. The user interfaces for the interaction with the effort prediction and completion modules were developed as the system Integration.

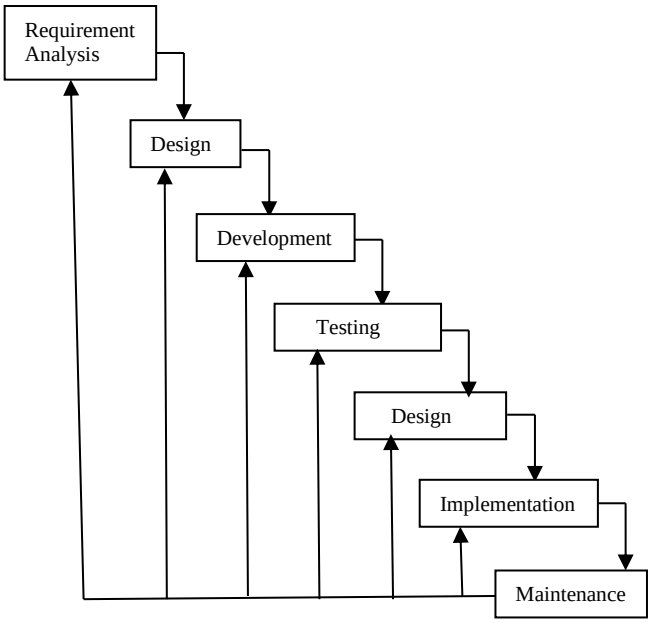

Figure 1: The Interactive V&V Waterfall Model

(iv) Phase 4 is testing the model and system through unit and system testing. Unit tests were conducted to verify the functionality of individual components of the Hybrid Ensemble Stacking Software Effort Prediction (HENSSEP) Model. This was done to ensure that each module performed as expected. Furthermore, end-to-end testing of the entire HENSSEP Model was done as well as validation of the system against different algorithms and datasets scenarios as system testing.

(v) Phase 5 is Deployment. This process was done by deploying the HENSSEP Model into the environments to include timelines and resources required. The systems performance and behavior were monitored after post-deployment.

(vi) Phase 6 is Maintenance and Monitoring. Maintenance is done to provide ongoing support and maintenance for the deployed system, to tackle bugs, issues, and updates as need arises. Monitoring mechanisms were implemented to track the performance of the system. This is to monitor the model drift and retrain the model periodically with new datasets.

3.2 The Ensemble Stacking Mathematical Model

The variables and components of the mathematical model are denoted as follows:

Let

X: The feature matrix representing the input features of software development projects

y**:** The vector representing the actual effort estimation values (target variable).

M: The number of base regression models.

H: The meta-model (the final regression model that combines base model predictions)

$h_m(X)$: The prediction made by base model m on the feature matrix X

Z**:** The matrix of meta-features, where each row i contains predictions from all base models plus Kalman Filter for data sample i.

Assumption: X and y are loaded and preprocessed

For each base model m, the model was trained on the training data to learn a mapping

$$i.e. \qquad h_m(X) \tag{1}$$

Kalman Filter was applied to the target variable y to generate smoothed predictions using $kf_{meta-features}$

For each data sample i in the validation set, generated a meta-feature vector $\boldsymbol{z_i}$ that contains predictions from all base models plus the Kalman Filter prediction.

$$\boldsymbol{z_i} = \left[h_1(X_i), h_2(X_i), \dots, h_M(X_i), kf_{meta-features_i}\right] \tag{2}$$

We trained the meta-model H (Random Forest Regression, Linear Regression, etc) on the entire training set, where the input is the matrix of meta-features Z and the target is the scaled target variable y.

For each data sample i in the test set, generated a meta-feature vector $\boldsymbol{z_i}$ using the same structures as for the validation set.

The trained meta-model H was used to make predictions on the test set, which resulted in predicted effort estimation values $\hat{y}$.

$\therefore$

$$\hat{y} = H(X) \tag{3}$$

$$\hat{y} = h_H(z) \tag{4}$$

$$\boldsymbol{\hat{y}} = \boldsymbol{h_H}\left(\left[\boldsymbol{h_1}(\boldsymbol{X_i}), \boldsymbol{h_2}(\boldsymbol{X_i}), \dots, \boldsymbol{h_M}(\boldsymbol{X_i}), \boldsymbol{kf_{meta-features}}\right]\right) \tag{5}$$

Where,

h_H represents the prediction function of the meta-model H

The developed ensemble stacking mathematical model is as presented in equation (5).

3.3 Ensemble Stacking Architecture

The model architecture consists of two main components: the conventional software effort estimation flow design and the developed ensemble stacking effort estimation flow design as seen on Figure 2. The latter comprises three key modules: feature selection, ensemble stacking model building, and the Kalman filter duration estimation module. Different methodologies were applied to each component, including a mathematical model for regression algorithms, an interactive waterfall methodology for interface development, and machine learning techniques for estimating software effort through ensemble stacking to achieve accurate results. Datasets of Albrecht, China, COCOMO, Desharnais, Kemerer, and Maxwell were preprocessed to reduce noise and outliers removed. The data processing technique employed was MinMaxScaler. This was used to normalize and scale the input features and target variable. Also used was Recursive Feature Elimination (RFE) for feature selection and Imputation for handling missing values. MinMax Scaling is also known as Min-Max normalization. The MinMax Scaling was applied in the model through loading and extracting the input data from the CSV file using the Pandas library. The input features X (software attributes) and the target variable Y (effort) were extracted from the dataset. Regression algorithms of Random Forest, Gradient Boosting, Support Vector, Decision Tree, and Multilayer Perceptron Regressors, were employed to build ensemble models. These models estimated software project efforts, and their outputs were combined in a meta-model using Linear Regression for more accurate predictions. The model's performance was compared with a previous hybrid ensemble

voting estimation by [9] using metrics like MAE, MSE, and RMSE, showing improvement in accuracy.

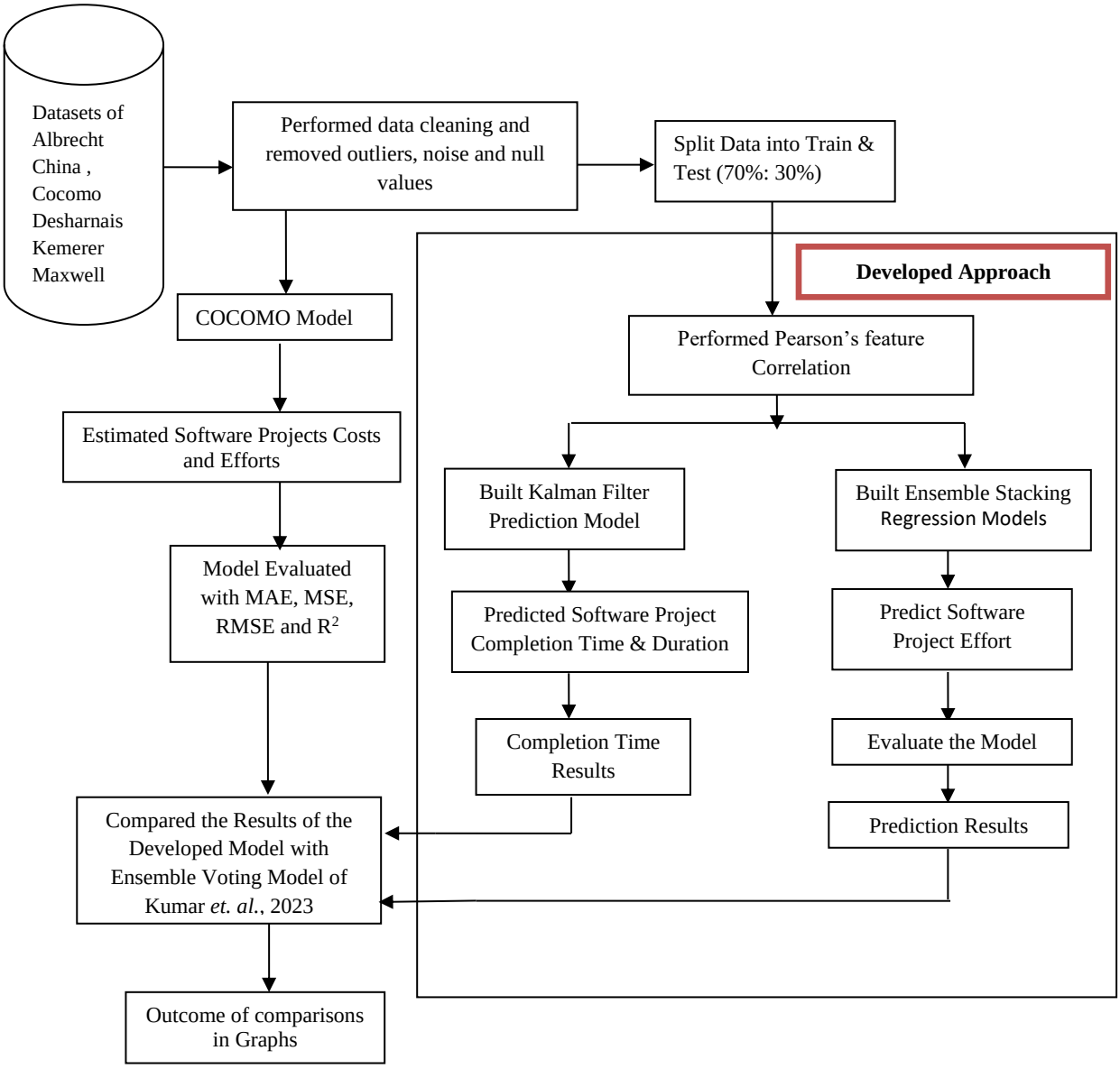

Figure 2: The Architecture of the Ensemble Stacking Prediction Model

3.4 The Model Development

The ensemble stacking model was developed as contained in Figure 3 using six base model algorithms of, Random Forest, Gradient Boosting, Support Vector Machines (SVR), Decision Tree, MLPRegressor (Neural Network), and Kalman Filter (KFA). These algorithms were selected based on their previous success in effort estimation and their ability to handle non-linear relationships [12]. Recursive Feature Elimination (RFE) was applied to select the most relevant features, and MinMax scaling was used to normalize the data. The ensemble stacking model leveraged on the strengths of the above machine learning models and potentially a traditional model (COCOMO) to arrive at more accurate software effort estimation, by combining the estimation through ensemble stacking, to improve the overall accuracy and generalizability of the model compared to using a single model alone.

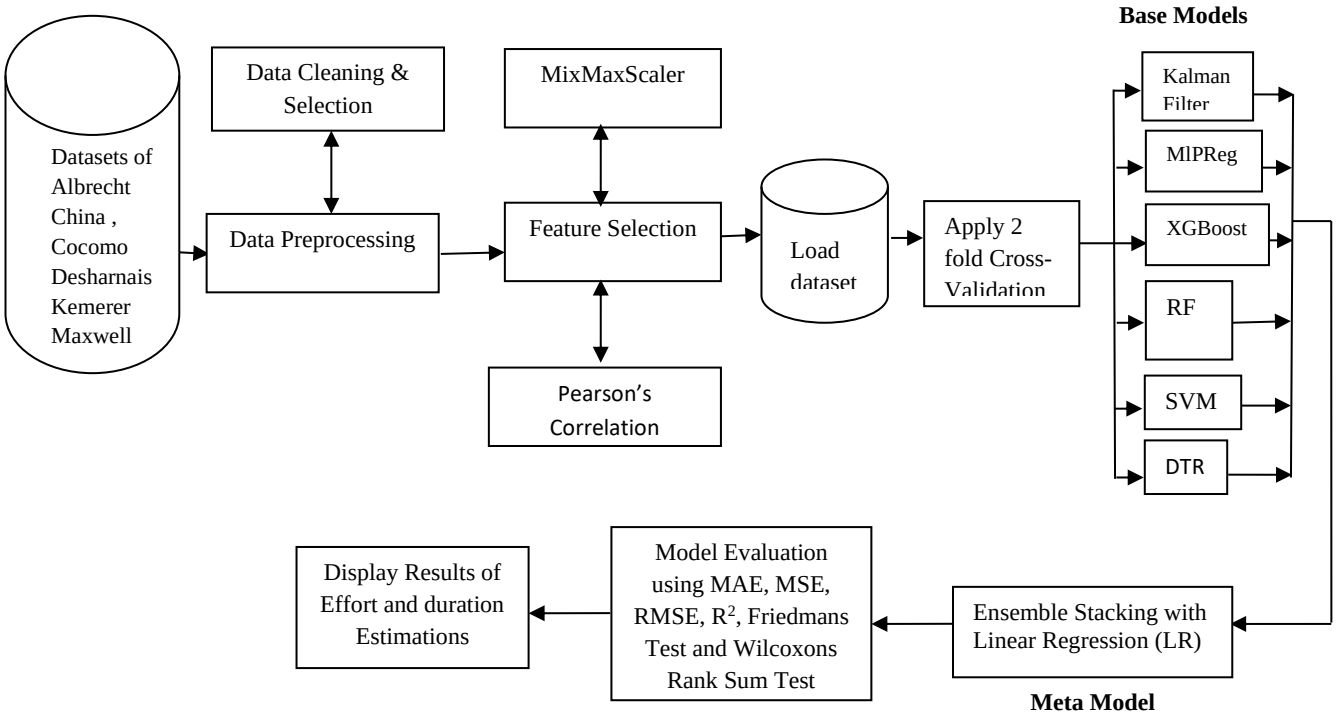

Fig 3: The Developed Model of the System

3.5 Data Sources

This study used six publicly available datasets of Albrecht, China, Cocomo81, Desharnais, Kemerer, and Maxwell, sourced online from the Zenodo repository. These datasets cover a wide range of project sizes and complexity, offering diverse test cases for evaluating the developed model as they were seen to be mostly employed in software effort estimation [12]. The number of records and the respective attributes are:

i. **Albrecht Dataset:** Contains 24 records with attributes such as effort, size (in function points), and cost drivers. The effort is measured in person-hours.

ii. **China Dataset:** The largest of the six, containing 499 records with multiple attributes and function points as the size metric. Effort is measured in person-hours.

iii. **Cocomo81 Dataset:** Comprising 63 records with attributes like Lines of Code (LOC), this dataset uses person-months as the effort unit.

iv. **Desharnais Dataset:** Contains 81 records, each representing software projects with effort recorded in person-hours.

v. **Kemerer Dataset:** The smallest dataset, with only 15 records, measuring size in KSLOC (thousands of source lines of code) and effort in person-months.

vi. **Maxwell Dataset:** Comprising 62 records, this dataset focuses on project effort in person-hours and project attributes.

3.6 Performance Metrics

The model was validated using the most widely employed performance metrics in effort estimation by researchers [12], to assess its accuracy and reliability. They are the Mean Absolute Error (MAE), that measures the average absolute difference between predicted and actual values, the Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) extend this by penalizing larger errors more heavily. MAE provides a straightforward measure of prediction accuracy, MSE gives insight into error magnitude with a focus on larger discrepancies, and RMSE allows interpretation in the same units as the target variable. Additionally, the R-Squared (R^2) score indicates how well the model captures the variance in the data, with a score closer to one signifying a better goodness of fit.

Furthermore, to compare multiple models and their performance across different datasets, Friedman's Test was conducted. This non-parametric test ranks algorithms and checks for significant differences, with a small p-value indicating that at least one algorithm outperform the others.

Finally, the Wilcoxon's Rank Sum Test was employed to compare two independent algorithms, assessing to know whether one significantly outperforms the other or not. Together, these statistical tests and performance metrics provided a comprehensive evaluation of the model's effectiveness across the various scenarios.

4. RESULTS AND DISCUSSION

4.1 Model Experiments

The first model experiment conducted was to predict the actual and predicted effort for the software projects across the six datasets. The experiment accepted the datasets as inputs which were used to train the base models, generated meta-features, and trained the meta-model in the ensemble stacking process for software development effort estimation. The expected outputs of the experiment were line graphs that showed the predicted and actual efforts for all the 6 datasets used. Figures 4(a) to 4(f) shows the results of the predicted effort versus actual effort for Albrecht, China, Cocomo81, Desharnais, Kemerer and Maxwell datasets used by the ensemble stacking model. The model was able to estimate the software effort across the datasets with minimum deviations. The x-axis represented the number of records in the datasets while the y-axis represented the effort of the respective records.

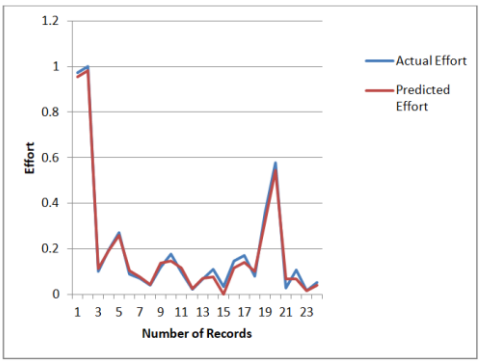

Fig 4a: Albrecht dataset

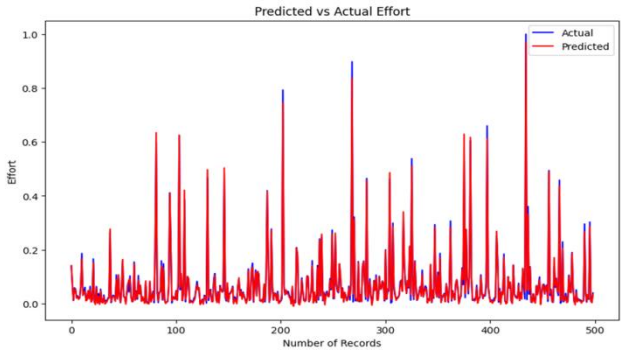

Fig 4b: China Dataset

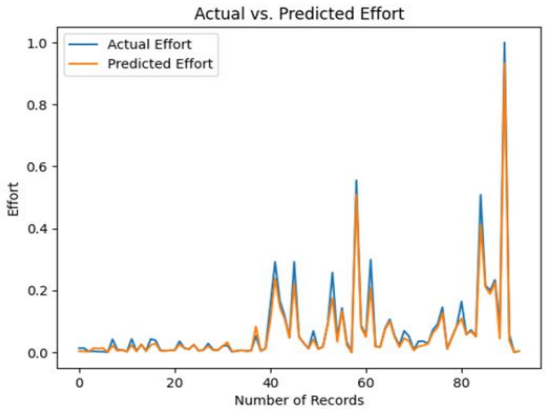

Fig 4c: Cocomo Dataset

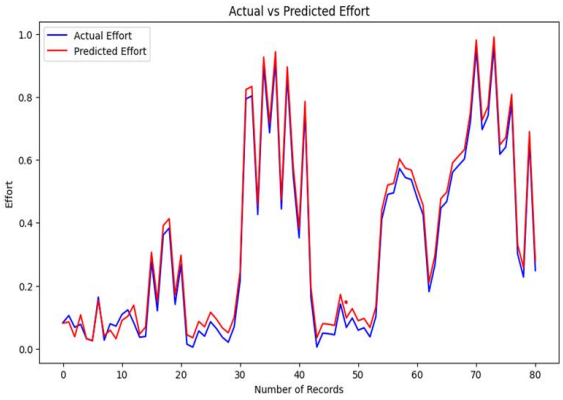

Fig 4d: Desharnais Dataset

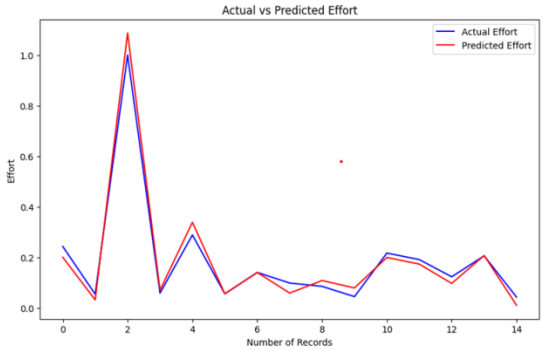

Fig 4e: kemerer Dataset

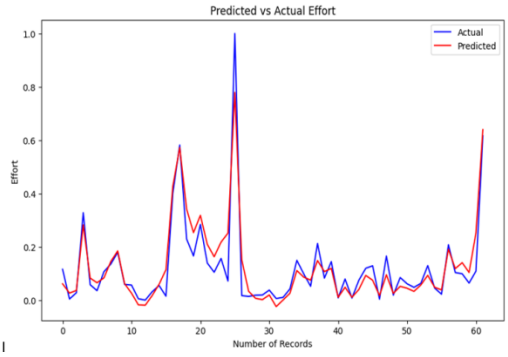

Fig 4f: Maxwell Dataset

The second model experiment was performed to investigate the Mean Absolute Error (MAE), Mean Squared Error (MSE) Root Mean Squared Error (RMSE) and R-Squared across the 6 datasets. The inputs to the experiment were the six datasets and the outputs are the tabulated MAE, MSE, RMSE and R-Squared values contained in Tables 1 – 4, corresponding to each algorithm and dataset combination. For the Albrecht dataset, the ensemble stacking model demonstrated a 95% R-Squared value, indicating that it captured most of the variance in the dataset. The MAE, MSE and RMSE values were lower compared to both the single-model approaches and the ensemble voting model. This suggests that the Kalman Filter effectively reduced noise and allowed for more accurate predictions, particularly in the presence of small variations in effort estimates. The improvements are especially notable in the areas where traditional models, such as COCOMO, would have under or overestimated effort due to the linear assumptions inherent in their methodologies. The China dataset yielded the best results, with an R-Squared value of 99%, making it the highest performing dataset in the study. The size of this dataset, along with the diversity of its attributes, allowed the model to generalize well and make highly accurate predictions. The MAE was minimal, indicating that the stacking model correctly predicted the majority of the projects' efforts. In the Cocomo81 dataset, the model performed similarly well, achieving an R-Squared of 93%. However, the RMSE was slightly higher compared to the Albrecht and China datasets, which could be attributed to the use of Lines of Code (LOC) as a key predictor. LOC is known to be a less reliable indicator of project effort due to variations in developer productivity and code quality. Nevertheless, the ensemble stacking model still outperformed the traditional COCOMO model by better accounting for these non-linear relationships.

The results for the Desharnais and Kemerer datasets were less impressive compared to the larger datasets. For the Desharnais

dataset, the model's R-Squared value was only 78%, while Kemerer had an R-Squared of 80%. The primary reason for this under-performance was the small size and outlier presence in these datasets. Both datasets were relatively small, with only 15 and 81 records, respectively, and contained several outliers that disproportionately impacted the prediction accuracy. While the Kalman Filter was effective in reducing noise, the small sample size limited the model's ability to generalize effectively. Further refinement, such as the application of outlier detection methods or hyper-parameter tuning, could potentially improve the results for these smaller datasets.

The Maxwell dataset also showed strong performance, with an R-squared of 91%. The model's accuracy in this dataset can be attributed to the well-structured nature of the data, which made it easier for the base models to generalize. The Kalman Filter played a key role in further refining the predictions by accounting for potential deviations in the data. Despite the challenges posed by smaller datasets, the Kalman Filter's inclusion still provided improvements by smoothing predictions and accounting for real-time data variations. Friedman's Test and Wilcoxon Rank Sum Test confirmed that the performance improvements of the ensemble stacking model were statistically significant in the larger datasets, particularly for Albrecht, China, and Cocomo81. The tests showed that the model's ability to reduce variance and improve prediction accuracy was consistent across multiple performance metrics, including MAE and RMSE.

Table 1: MAE of the Individual Algorithms and Hybrid Ensemble Model using the 6 Datasets

Mean Absolute Error (MAE)						
Algorithms	**Dataset 1 Albrecht**	**Dataset 2 China**	**Dataset 3 Cocomo81**	**Dataset 4 Desharnais**	**Dataset 5 Kemerer**	**Dataset 6 Maxwell**
Linear Regression	0.1026	0.0067	0.0596	0.2480	0.1812	0.1047
Random Forest Regressor	0.0843	0.0079	0.0500	0.2403	0.12380	0.0544
Decision Tree Regressor	0.1418	0.0102	0.0771	0.2161	0.1491	0.0654
Support Vector Regressor	0.1179	0.0689	0.0958	0.2449	0.1507	0.1093
Gradient Boosting Regressor	0.0866	0.0078	0.0611	0.2145	0.1367	0.0608
MLP Regressor	0.0751	0.0239	0.0736	0.2519	0.1396	0.2314
Kalman Filter	0.0579	0.0476	0.0434	0.1557	0.1107	0.0618
Kumar *et al*., [9] [Voting]	0.0775	0.0077	0.1466	0.0627	0.0925	0.1221
HENSSEP (Stacking)	**0.0333**	**0.0072**	**0.0528**	**0.1192**	**0.1166**	**0.0383**

Table 2: MSE of the Individual Algorithms and Hybrid Ensemble Model versus the 6 Datasets used in the Experiment

Mean Square Error (MSE)						
Algorithms	**Dataset 1 Albrecht**	**Dataset 2 China**	**Dataset 3 Cocomo81**	**Dataset 4 Desharnais**	**Dataset 5 Kemerer**	**Dataset 6 Maxwell**
Linear Regression	0.0302	0.0003	0.0114	0.0901	0.0608	0.0213
Random Forest Regressor	0.0304	0.0006	0.0126	0.0843	0.0496	0.0105
Decision Tree Regressor	0.0732	0.0009	0.0388	0.1327	0.0540	0.0136
Support Vector Regressor	0.0474	0.0073	0.0150	0.0949	0.0630	0.0212
Gradient Boosting Regressor	0.0227	0.0007	0.0200	0.0828	0.0509	0.0161
MLP Regressor	0.0181	0.0015	0.0149	0.0923	0.0545	0.0724
Kalman Filter	0.0077	0.0067	0.0076	0.0475	0.0331	0.0105
Kumar et al., [9] [Voting]	0.0099	0.0007	0.0527	0.0061	0.0160	0.0555
HENSSEP (Stacking)	**0.0028**	**0.0001**	**0.0060**	**0.0239**	**0.0237**	**0.0033**

Table 3: RMSE of the Individual Algorithms and Hybrid Ensemble Model versus the 6 Datasets used in the Experiment

Root Mean Square Error (RMSE)						
Algorithms	**Dataset 1 Albrecht**	**Dataset 2 China**	**Dataset 3 Cocomo81**	**Dataset 4 Desharnais**	**Dataset 5 Kemerer**	**Dataset 6 Maxwell**
Linear Regression	0.1492	0.0189	0.1039	0.2997	0.2262	0.1448
Random Forest Regressor	0.1345	0.0245	0.1119	0.2899	0.1770	0.0920

Decision Tree Regressor	0.2105	0.0304	0.1870	0.3606	0.2031	0.1110
Support Vector Regressor	0.1696	0.0858	0.1201	0.3074	0.2128	0.1386
Gradient Boosting Regressor	0.1271	0.0263	0.1399	0.2864	0.1887	0.1204
MLP Regressor	0.1081	0.0380	0.1145	0.3032	0.1996	0.2673
Kalman Filter	0.0809	0.0815	0.0873	0.2178	0.1819	0.1026
Kumar et al., [9] [Voting]	0.0996	0.0270	0.2297	0.0783	0.1031	0.2356
HENSSEP (Stacking)	**0.0531**	**0.0110**	**0.0776**	**0.0547**	**0.1541**	**0.0571**

Table 4: R Squared Values across the datasets using the Model

Dataset	**R-squared Value**
China	0.9919
Albrecht	0.9595
Cocomo81	0.6822
Desharnais	0.8061
Kemerer	0.5681
Maxwell	0.8801

The third experiment was performed to determine the Friedman's test on the models. The Friedman's test was conducted to assess the statistical significance of differences among the algorithms used in the study. The inputs to the experiment were the MAE values gotten from all the experiments using the six datasets as tabulated in Table 1 by the developed model. The output was the Friedman's test statistics and the p-value is as presented:

Friedman's test statistic: 22.0

P-value: 0.004915867265928975 ≈ 0.005

There is a significant difference among the algorithms with the Friedman's test.

According to [11], a p-value ≤ 0.05 for a Friedman's test indicates statistical difference at a general level. This result suggested that at least one of the algorithms used in the study performed significantly different from the others. It implies that the choice of algorithm had a meaningful impact on the performance, and it's not merely due to random chance.

The Wilcoxon**'s** Rank Sum Test, also known as the Mann-Whitney U test, was another experiment that was investigated to assess the performance differences between pairs of algorithms in the study. The test produced p-values for each pair-wise comparison, and the p-value helped determine whether there was a statistically significant difference in the performance of these algorithms. The output was the significant algorithm pairs based on Mann-Whitney U test statistics and the p-value was presented in Table 5. It was observed from Table 5 that each 2 x 2 comparison produced a p-value < 0.05. This was an indication that the performance of each algorithm was statistically different from one another [11].

Table 5: Wilcoxon' Rank Sum (Mann-Whitney U) values of the Algorithms across the Datasets

	Kumar et al., [9] [Voting]	**HENSSEP (Stacking)**	**Kalman Filter**	**MLP Regressor**	**Gradient Boosting Regressor**	**Support Vector Regressor**	**Decision Tree Regressor**	**Random Forest Regressor**	**Linear Regression**
Kumar et al., [9] [Voting]	0	0.0011	0.0038	0.0047	0.0081	0.0318	0.0316	0.0035	0.0026
HENSSEP (Stacking)	0.0011	0	0.0027	0.0006	0.0004	0.0027	0.0031	0.0023	0.0016
Kalman Filter	0.0038	0.0027	0	0.0020	0.0063	0.0031	0.0015	0.0056	0.0093
MLP Regressor	0.0047	0.0006	0.0020	0	0.0341	0.0081	0.0068	0.0062	0.0084

Gradient Boosting Regressor	0.0081	0.0040	0.0063	0.0341	0	0.0217	0.0031	0.0056	s0.0015
Support Vector Regressor	0.0318	0.0027	0.0031	0.0081	0.0217	0	0.0156	0.0312	0.0031
Decision Tree Regressor	0.0316	0.0031	0.0015	0.0068	0.0031	0.0156	0	0.0156	0.0068
Random Forest Regressor	0.0035	0.0023	0.0056	0.0062	0.0056	0.0312	0.0156	0	0.0062
Linear Regression	0.0026	0.0016	0.0093	0.0084	0.0015	0.0031	0.0068	0.0062	0

The last experiment was performed to compare the developed model with the existing works of literature to ascertain whether there was a significant improvement or not. The ensemble voting study of [9] was used for benchmarking. The datasets were of the same type, dimension, attributes as well as the machine learning algorithms except for the introduction of three new algorithms to the model. The inputs to the experiment were the MSE values of [9] [Voting] and HENSSEP (Stacking) models across the datasets from Table 2. The output is a line graph of the Mean Square Errors (MSE) of [9] [Voting] and the developed ensemble stacking model (HENSSEP) as seen in Figure 5.

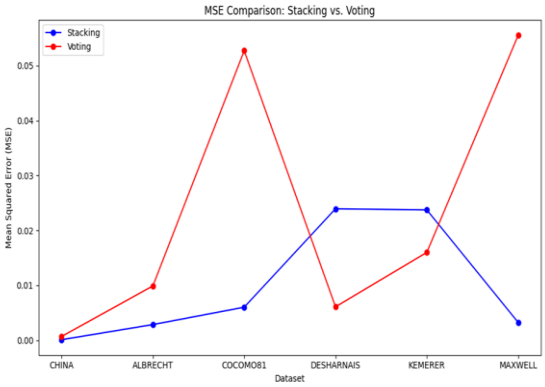

Fig 5: MSE of [9] [Voting] and the developed Ensemble Stacking Model

It was seen that the developed model performed better than that of the hybrid ensemble voting done by [9] across four of the datasets used except in two datasets of the Kemerer and Desharnais. The model was seen to have lower values of MSE, reflecting an average quantitative measure of the estimation accuracy of the model. That the MSE values were higher in the four datasets was a strong indication that the software project effort estimation accuracy challenge has been improved upon greatly with our developed model as it was platform compatible with the widely used software effort estimation datasets.

5. CONCLUSION

This paper developed an ensemble stacking software effort estimation model that addressed how the accuracy of software effort estimation could be improved upon by harnessing the capabilities of hybrid ensemble stacking techniques, while effectively addressing uncertainty and providing practical implementation guidance. This was achieved through the development of an ensemble stacking model for software effort estimation, implementation of the ensemble stacking model, developed a mathematical model and integrated Kalman Filter algorithm to the design and implementation of the ensemble stacking model and the mathematical model formulated. The performance of the model was also evaluated for accuracies and statistical significant differences amongst the algorithms used for its implementation. From the experiments conducted, it was seen that the developed ensemble stacking software effort estimation model was able to show the actual and estimated efforts respectively across the datasets of the software projects used with minimal deviations, gave better accuracies and had minimal errors. This was seen from the MAE, MSE, RMSE, R-Squared, Friedman's test and Wilcoxons rank sum test values recorded. The MAE, MSE and RMSE values of the models used were at the lowest across the datasets and the respective regression models with values of 0.0072, 0.0001 and 0.0110 respectively. The R-Squared values were above 80% in the four datasets while 57% and 68% for two datasets respectively. The Friedmans Test indicated that there was a statistical difference among the algorithms with a p-value of 0.0049 which according to Atsa'am and Wario (2021), a p-value of ≤ 0.05 for a Friedman's test indicated a statistical difference at a general level. The Wilcoxons Rank Sum test (Mann-Whitney U test) on Table 5 indicted that each 2 x 2 algorithm comparison produced a p-value < 0.05, which indicated that the performance of each algorithm was statistically different from one another [11]. The study equally compared the developed model with the hybrid ensemble voting study of [9] and it was seen to have a better improvement in terms of the estimation accuracies and the performance metrics.

This work is open to further research by incorporating new data and refining the system as more data becomes available. Other ensemble techniques should be explored to further improve on the study as well as other feature engineering techniques for data preprocessing. This is important so that the systems adaptability to evolving software development trends/technology can be met and to improve the accuracy of the model.

6. REFERNCES

[1] Jørgensen, M. (2014). "Challenges in Software Cost Estimation." *Journal of Systems and Software*, 101, 174-190.

[2] Awan, M. J., and Asif, M. (2021). "A Comprehensive Review on Software Effort Estimation: Machine

[3] Idri, A., and Abran, A. (2015). "Analogy-based Software Development Effort Estimation: A Systematic Mapping and Review." *Information and Software Technology*, 58, 206-230.

[4] Pospieszny, P., Czarnacka-Chrobot, B. and Kobyliński, A. (2017). An Effective Approach for Software Project Effort and Duration Estimation with Machine Learning Algorithms, *The Journal of Systems & Software* . doi: 10.1016/j.jss.2017.11.066.

[5] BaniMustafa, A. (2018). Predicting Software Effort Estimation Using Machine Learning Techniques. *2018 8th International Conference on Computer Science and Information Technology (CSIT).* doi:10.1109/csit.2018.8486222

[6] Zhang, X., Yang, F., and Jiang, S. (2020). "A Review on Ensemble Learning Algorithms in Machine Learning." *Journal of Physics: Conference Series*, 1693, 012129.

[7] Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2019). "Do We Need Hundreds of Classifiers to Solve Real World Classification Problems?" *Journal of Machine Learning Research*, 15(90), 3133-3181.

[8] Sharma, A., and Kaur, A. (2020). "A Review on Ensemble Techniques in Machine Learning." *International Journal of Advanced Science and Technology*, 29(3), 6874-6886.

[9] Kumar, B. K., Bilgaiyan, S. and Mishra, B. S. P. (2023). Software Effort Estimation through Ensembling of Base Models in Machine Learning using a Voting Estimator. *International Journal of Advanced Computer Science and Applications (IJACSA), 14,(2)*

[10] Fadhil, A. A., Alsarraj, R. G. and Altaie, A. M. (2020). Software Cost Estimation Based on Dolphin Algorithm. *IEEE Access 8,* Digital Object Identifier 10.1109/ACCESS.2020.2988867.

[11] Atsa'am, D. D. and Wario, R. (2021). Classifier Selection for the Prediction of Dominant Transmission Mode of Coronavirus within Localities: Predicting COVID-19 Transmission Mode. *International Journal of E-Health and Medical Communications (IJEHMC) 12(6)*

[12] Akumba, B. O., Blamah, N.V., Agaji, I. and Ogalla, E. (2023). *Quest Journals Journal of Software Engineering and Simulation 9(4). 01-07 ISSN(Online) :2321-3795 ISSN (Print):2321-3809* www.questjournals.org