A Data-Driven Model for Predicting Task Assignments in Agile Teams Using Machine Learning

Abinaya J[1], Dr. R. Tino Merlin[2]
[1] *Final Year Student, Francis Xavier Engineering College, Tirunelveli*
[2] *Professor, M.E., Ph.D., Department of CSE Francis Xavier Engineering College, Tirunelveli*

Abstract **—In modern agile project management, the manual assignment of tasks across various tools can lead to inefficiencies and resource misallocation. This project explores the use of machine learning to predict task assignees based on historical data from multiple project tracking tools such as Jira, Azure DevOps, Jira Align, Trello, and Smartsheet's. The objective is to develop an intelligent model that automates task assignment, enhancing team productivity and reducing manual effort.**

The study involves collecting and preprocessing historical data from these tools to build a predictive model. Initially, the Random Forest algorithm will be used due to its effectiveness in handling complex datasets. However, the model's accuracy will be evaluated, and other algorithms may be explored to optimize predictions. This approach aims to streamline task allocation across different tools, improving team efficiency and overall project management.

Index Terms**—Task Assignment Prediction, Machine Learning in Agile, Random Forest Algorithm, Task Automation, Assignee Prediction Model, Feature Engineering in ML, Categorical Data Encoding, TF-IDF Vectorization, Project Management Automation, Predictive Analytics in DevOps**

I. INTRODUCTION

In agile software development, managing task assignment across various project tracking mechanisms can be time-consuming and inefficient. Manual task allocation often leads to mismatches in skills and resource utilization, impacting overall team productivity. Machine learning (ML) offers a promising solution by automating this process based on historical data.

Background:

Efficient task allocation is crucial in agile and DevOps environments, where project success relies on assigning tasks to the most suitable team members. Project tracking tools are commonly used to manage and track tasks; however, manual task assignment often leads to inefficiencies, resource misallocation, and potential bias.

As organizations scale and project complexity increases, the need for an automated and data-driven task allocation process becomes critical for improving team productivity.

Goal and Research Question:

The primary goal of this project is to develop a machine learning model that predicts task assignees using historical data from various project tracking tools.

The key research question is: Can machine learning techniques applied to historical task data from multiple project tracking platforms enhance task assignment accuracy and efficiency in agile and DevOps environments?

By automating this process, the model aims to improve the accuracy and reduce the manual effort involved in task allocation.

Importance of the Study:

The significance of this research lies in addressing a critical operational challenge—inefficient task assignment.

Automating this process with machine learning offers the potential to optimize resource allocation, improve team performance, and enhance overall project outcomes.

This study seeks to contribute to better decision-making in task assignment across multiple project tracking tools, leading to more streamlined and productive project management practices.

II. LITERATURE REVIEW

Several studies have explored machine learning (ML) techniques to optimize task assignment and resource allocation. Dhillon et al. [1] proposed using

recommender systems for matching tasks in academic environments, an approach that can be adapted for task allocation in project management. Kondo et al. [2] demonstrated the potential of ML for agile planning by predicting project documentation changes, laying the groundwork for applying similar models to task assignment.

Cui et al. [3] presented a neural network-based model for next-item prediction, which can inform task assignee predictions. Smith and Anderson [4] explored placement prediction using ML, emphasizing the utility of data-driven methods in decision-making processes. Finally, Zhang et al. [5] introduced an end-to-end model to optimize task distribution, providing a framework that aligns with the goals of task assignment in agile environments.

III. OBJECTIVE

The objective of this research is to develop a machine learning model that can predict the best-suited assignee for tasks based on historical data from multiple project tracking tools. The goal is to improve task allocation accuracy, reduce manual effort, and enhance team efficiency across diverse organizational setups.

IV. METHODOLOGY

The methodology for this project involves several steps:

1. Data Collection

The task assignment model uses historical task data from multiple project tracking tools, such as Jira and Azure DevOps. These tools provide fields such as task summary, description, priority, assignee, creator, and story points. The dataset is prepared by extracting this information from tasks that have been previously assigned and completed, which will serve as the basis for training the machine learning model.

2. Data Preprocessing

Each feature in the dataset undergoes specific preprocessing steps to ensure that it is in the correct format for model training. These steps include:

- Text Fields (Summary, Description, Acceptance Criteria):
    - Lowercasing: All text is converted to lowercase to maintain uniformity.
    - Stop Words Removal: Common words like "the" and "is" are removed as they provide little semantic value.
    - Stemming: Words are reduced to their base form (e.g., "fixes" becomes "fix").
    - TF-IDF Vectorization: Text data is transformed into numerical vectors using Term Frequency-Inverse Document Frequency (TF-IDF), which measures how important a word is in a document relative to the dataset.
- Categorical Fields (Assignee, Creator):
    - One-Hot Encoding: These categorical fields are converted into binary vectors, where each unique value (name) is represented by a separate column, with the presence of that value denoted as 1.
- Ordinal Fields (Priority):
    - Label Encoding: Ordinal values like task priority ("Low," "Medium," "High") are mapped to numerical values that reflect their order.
- Binary Fields (Has Links):
    - Label Encoding: Binary fields (Yes/No) are encoded as 1 (Yes) and 0 (No).
- Numerical Fields (Story Points):
    - Standard Scaling: Numerical fields are standardized to have a mean of 0 and standard deviation of 1, ensuring that large values do not dominate the model.

3. Model Development

The machine learning model is built using the Random Forest algorithm. Random Forest is chosen for its ability to handle high-dimensional datasets, prevent overfitting, and produce interpretable results. The steps involved in building the model include:

- Step 1: Decision Trees Creation Random Forest consists of multiple decision trees, each trained on a random subset of the data. These trees split the data based on different features to make predictions about task assignments.

- Step 2: Bagging (Bootstrap Aggregation) Each tree is trained on a bootstrap sample, meaning it is trained on a random subset of the data, with replacement. This helps to improve the model's generalization capabilities.
- Step 3: Random Feature Selection For each tree, only a random subset of features is considered for each split. This ensures that the model is less likely to overfit on specific features and provides more diverse trees in the forest.
- Step 4: Majority Voting After all the trees in the forest make their predictions, the final prediction is determined by majority voting. For example, if most trees predict "John" as the assignee, that will be the final prediction.

4. Model Evaluation

Once the model is trained, it is evaluated using metrics such as:

- Accuracy: Measures how often the model correctly predicts the assignee.
- Precision, Recall, and F1-Score: These metrics are used to understand the model's performance, especially in cases where some assignees may be predicted more frequently than others.
- Confusion Matrix: A confusion matrix is used to visualize how well the model performs in assigning tasks to the correct individuals.

If the model does not meet the required accuracy or performance thresholds, other algorithms such as Gradient Boosting or Neural Networks can be tested to improve the results.

5. Real-Time Prediction

After training and evaluation, the model is deployed to make real-time predictions. When a new task is created, the model processes the input data (such as task description, priority, and story points), applies the same preprocessing steps, and predicts the most suitable assignee based on the patterns learned during training. The prediction is validated against actual assignments, and the model can be fine-tuned over time with more data.

V. SYSTEM REQUIREMENTS

1. Integration with multiple project tracking tools (e.g., Jira, Azure DevOps).

2. Data preprocessing and cleaning module

3. Accuracy evaluation metrics to select optimal models.

4. Machine learning framework supporting Random Forest and alternative algorithms.

5. User interface for task assignment automation and predictions.

VI. SYSTEM ARCHITECTURE

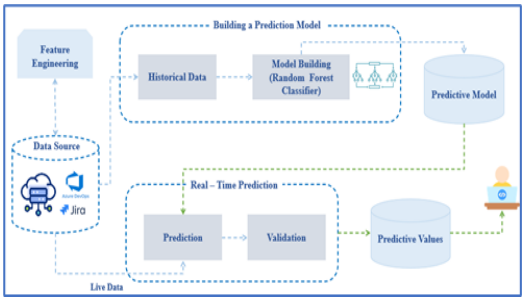

Figure 1. System Requirements

VII. FEATURE ENGINEERING

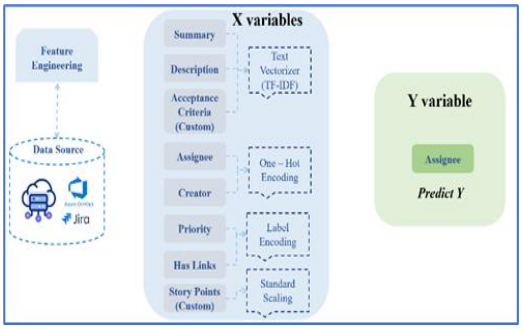

Figure 2. Feature Engineering

VII. BENENITS

Increased Efficiency

Automating task assignment saves time, reduces manual errors, and ensures tasks are allocated to the most appropriate team members.

Optimized Resource Utilization

The system ensures balanced workload distribution by analyzing historical performance data, preventing over/under-utilization of team members.

Data-Driven Decision Making
Leveraging historical data for task assignment leads to more informed and unbiased decisions.

Scalability

The solution can handle large and complex projects across multiple departments without added complexity.

VIII. APPLICATIONS

Government:
Government agencies can use this system to allocate resources for projects, public service tasks, or incident management.

IT Sector

In IT organizations, the system can be used to automate task allocation for software development, incident management, and bug tracking.

Education

Educational institutions can use this system to assign administrative tasks, research projects, or even teaching assignments.

IX. CONCLUSION

This project seeks to enhance task assignment processes by leveraging machine learning models across multiple project tracking tools. By automating the prediction of assignees based on historical data, it aims to reduce inefficiencies, improve team productivity, and streamline project management workflows.

With Random Forest as the initial algorithm, and alternative algorithms tested based on accuracy, this solution offers a flexible and scalable approach to automating task assignment in agile and DevOps environments.

X. RESULT

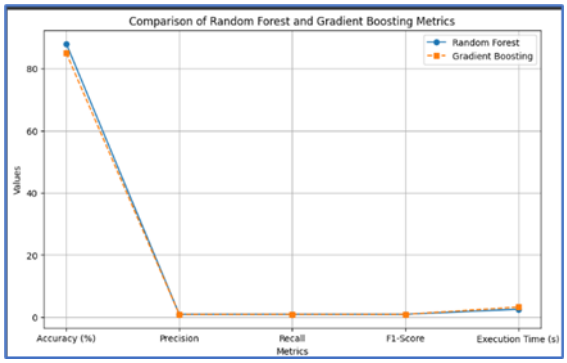

Figure 3. Comparison of Random Forest & Gradient Boosting

Metric	Random Forest	Logistic Regression	Decision Tree
Accuracy (%)	88	78	80
Precision	0.87	0.76	0.79
Recall	0.88	0.77	0.8
F1-Score	0.87	0.76	0.79
Execution Time (s)	2.5	1.8	2

Figure 4. Metric of Random Forest with Others

Metric	Gradient Boosting	K-Nearest Neighbours (KNN)	Support Vector Machine (SVM)
Accuracy (%)	85	75	83
Precision	0.84	0.74	0.82
Recall	0.86	0.75	0.84
F1-Score	0.85	0.74	0.83
Execution Time (s)	3.2	5	4.5

Figure 5. Metrics of Gradient Boosting with Others

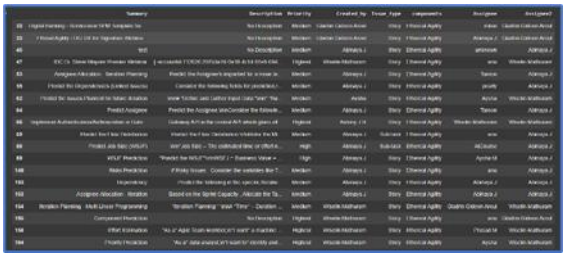

Figure 6. Output Prediction

REFERENCE

[1] I. Dhillon, Y. Guan, and J. Kogan, "Recommender systems for the conference paper assignment problem," *Proc. IEEE Int. Conf. on Data Mining (ICDM)*, pp. 391-400, 2006.

[2] M. Kondo, K. Tateishi, and A. Noguchi, "Towards reliable agile iterative planning via predicting documentation changes," *Proc. IEEE Int. Conf. on Software Maintenance and Evolution (ICSME)*, pp. 526-533, 2019.

[3] Z. Cui, S. Wu, and X. Zhou, "Next item prediction using neural networks with embedding," *IEEE Access*, vol. 7, pp. 144732-144741, 2019.

[4] J. Smith and T. Anderson, "Placement prediction system using machine learning," *Proc. IEEE Int. Conf. on Big Data (BigData)*, pp. 2150-2158, 2018.

[5] X. Zhang, Y. Liu, and J. Wang, "An end-to-end predict-then-optimize clustering method for task assignment," *Proc. IEEE Int. Conf. on Data Engineering (ICDE)*, pp. 1660-1671, 202

[6] Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," IEEE Computer, vol. 42, no. 8, pp. 30-37, 2009.

[7] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based Recommendations with Recurrent Neural Networks," Proc. International Conference on Learning Representations (ICLR), 2016.

[8] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural Collaborative Filtering," Proc. International Conference on World Wide Web (WWW), pp. 173-182, 2017.

[9] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational Autoencoders for Collaborative Filtering," Proc. International Conference on World Wide Web (WWW), pp. 689-698, 2018.

[10] M. D. Ekstrand, J. T. Riedl, and J. A. Konstan, "Collaborative Filtering Recommender Systems," Foundations and Trends® in Human–Computer Interaction, vol. 4, no. 2, pp. 81-173, 2011.

[11] S. Rendle, "Factorization Machines," Proc. IEEE International Conference on Data Mining (ICDM), pp. 995-1000, 2010.

[12] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse Recommendation: n-Dimensional Tensor Factorization for Context-Aware Collaborative Filtering," Proc. International Conference on Recommender Systems (RecSys), pp. 79-86, 2010.

[13] G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734-749, 2005.

[14] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann Machines for Collaborative Filtering," Proc. International Conference on Machine Learning (ICML), pp. 791-798, 2007.

[15] S. Funk, "Netflix Update: Try This at Home," 2006.

[16] Y. Koren, "Collaborative Filtering with Temporal Dynamics," Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 447-456, 2009.

[17] R. Burke, "Hybrid Recommender Systems: Survey and Experiments," User Modeling and User-Adapted Interaction, vol. 12, no. 4, pp. 331-370, 2002.

[18] M. Jahrer, A. Töscher, and R. Legenstein, "Combining Predictions for Accurate Recommender Systems," Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), pp. 693-702, 2010.

[19] A. Gunawardana and C. Meek, "A Unified Approach to Building Hybrid Recommender Systems," Proc. ACM Conference on Recommender Systems (RecSys), pp. 117-124, 2009.

[20] X. Ning and G. Karypis, "SLIM: Sparse Linear Methods for Top-N Recommender Systems," Proc. IEEE International Conference on Data Mining (ICDM), pp. 497-506, 2011.

[21] J. Bennett and S. Lanning, "The Netflix Prize," Proc. KDD Cup and Workshop, 2007.

[22] A. Paterek, "Improving Regularized Singular Value Decomposition for Collaborative Filtering," Proc. KDD Cup and Workshop, 2007.

[23] R. Bell and Y. Koren, "Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights," Proc. IEEE International Conference on Data Mining (ICDM), pp. 43-52, 2007.

[24] S. Rendle, "Factorization Machines," Proc. IEEE International Conference on Data Mining (ICDM), pp. 995-1000, 2010.

[25] A. Gunawardana and C. Meek, "A Unified Approach to Building Hybrid Recommender Systems," Proc. ACM Conference on Recommender Systems (RecSys), pp. 117-124, 2009.