UNIVERSITY OF ECONOMICS AND LAW

FACULTY OF INFORMATION SYSTEMS

MACHINE LEARNING IN BUSINESS ANALYTICS REPORT

MAJOR IN DIGITAL BUSINESS AND ARTIFICIAL INTELLIGENCE

TOPIC NAME

Building a Machine Learning app for factory management

Supervisor: **Tran Duy Thanh, Ph.D.**

Group: WabiSabi

Class: K21416C

Ho Chi Minh City, May 2024

Members of Group Wabisabi

NO.	**Full name**	**Student ID**	**Point / 10 (Individual Contribution)**	**Signature**
1	Nguyễn Hoàng Duy Thông	K214162156	100	
2	Nguyễn Thị Thu Thảo	K214162154	100	
3	Hoàng Trần Thế Phúc	K214162150	100	
4	Lục Minh Phú	K214161342	100	

Acknowledgements

First and foremost, we would like to express our heartfelt gratitude to all the professors in the Department of Management Information Systems for providing the knowledge, skills, and information necessary to carry out this project.

Furthermore, we want to extend our profound gratitude to our mentor, Ph.D Tran Duy Thanh, for providing the opportunity for research and offering invaluable guidance throughout this project.

Group WabiSabi

Commitment

We commit to declare that the results achieved in this thesis are our own work and are not copied from any other source. Throughout the entire content of the project, the information presented is either personal or synthesized from various sources. All references are properly cited and quoted according to regulations. We will take full responsibility and accept any form of disciplinary action if found in violation.

Ho Chi Minh City, March 2024

Group WabiSabi

Table of Contents

List of Tables

List of Figures

List of Abbreviation

Abbreviation	Meaning
ML	Machine Learning
AI	Artificial Intelligence
HR	Human Resources

Abstract

This project focuses on developing a machine learning-based application to enhance factory management by analyzing employee performance data. The primary objectives are to cluster employees into groups based on their efficacy and characteristics, predict future employee performance using machine learning models, and provide insightful statistics on various aspects of the workforce. The application leverages data spanning 18 weeks and encompassing 508 employees, including details on their daily efficacy, health indicators, roles, and significant events such as resignations or terminations.

By clustering employees into high, moderate, or low efficacy groups with relatively stable or highly variable performance, the application enables optimized labor allocation and targeted interventions. Moreover, the integrated machine learning models facilitate the prediction of employee efficacy based on factors like health, commitment, and skills, allowing for proactive management and resource optimization. Comprehensive statistical visualizations provide managers with an in-depth understanding of workforce dynamics, including gender and age distributions, resignation reasons, work group efficacy, and the impact of health and goodness on performance.

The developed application serves as a valuable tool for factory managers, enabling data-driven decision-making, improved workforce management, and enhanced operational efficiency. Through its machine learning capabilities and insightful analytics, the application supports the identification of high-performing employees, the development of targeted training programs, and the implementation of effective retention strategies.

Chapter 1. Project Overview

1.1. Introduction

This project focuses on managing the performance of 508 employees in a factory over 18 weeks (from 2021-01-01 to 2022-06-30). It includes working performance, health indicators, roles, etc., and aims to identify the relationships between these indicators and employees' efficacy and their likelihood of leaving the job. We will cluster groups of employees to assess the overall situation of the factory and predict the factory's productivity for the following week.

1.2. Objectives

- Help the company identify groups of employees with high efficacy to assign to important positions.
- Cluster employees based on different key factors.

1.3. Subjects and Scopes

- Research subject:
    - Efficacy and clustering of worker
- Reseach scope:
    - Time: 04/2024 – 05/2024
    - Space: Employees in the factory

1.4. Business challenges

1.4.1. Business problems

In a factory environment, effective management of worker performance is crucial for maintaining high productivity and operational efficiency. One of the key challenges is to understand how to classify the workforce based on their performance data to optimize labor allocation and improve overall productivity.

There is a need to develop a robust system that can cluster workers into high, medium, or low-performing groups, identifying those with relatively stable performance versus those with highly varying performance. This classification will enable management to implement targeted interventions, provide appropriate training

programs, and allocate tasks more efficiently, ultimately boosting the factory's operational output.

Additionally, accurately predicting worker performance would be a valuable tool for early detection of potential underperformers and timely application of interventions. Detailed performance statistics for employees would also provide management with a comprehensive overview of the workforce situation, enabling informed decisions regarding hiring, training, and task assignments.

Building an employee management application with features such as worker clustering, performance prediction, and relevant statistics will assist the factory in optimizing workforce management, enhancing productivity, and driving operational efficiency.

1.4.2. Business questions

- How can we cluster workers based on their performance metrics? What are the most significant predictors of a worker's productivity and efficacy?
    - Develop a machine learning model to cluster workers based on their daily efficacy into categories of high, moderate, or low efficacy that can be either relatively stable or highly variable.
    - Build a model to predict the efficacy rate of employees.
- How to track current employee performance?
    - Build statistical charts related to employees and their performance.

1.5. Methods and models

1.5.1. Methods

In the research design, we utilize a structured dataset comprising a total of 42 data fields. These variables are categorized into two main types: quantitative and qualitative.

Quantitative variables include the age of employees, actual performance levels, and mental health indicators such as commitment and awareness. Employee age is a continuous variable measuring the age of each individual in the factory. Actual performance level is a numerical variable measuring the degree of performance exhibited by each employee on a specific working day. Mental health indicators such as

commitment and awareness assess the level of commitment and awareness of employees towards their work and work environment.

Qualitative variables include gender, event type, and role within the organization. Gender categorizes employees into male or female. Event type includes events such as participation, resignation, and termination, reflecting specific events experienced by each employee. Role within the organization categorizes employees based on their roles, such as team leader, supervisor, or regular staff.

Quantitative variables provide information about the magnitude and degree of important factors in the study, while qualitative variables help classify and identify specific characteristics of each employee. The combination of both types of variables will provide a comprehensive view of performance and labor attrition in a manufacturing environment.

1.5.2. Models

In this subsection, we specify the machine learning models employed for predictive tasks:

- Cluster analysis model:

We employ clustering algorithms such as K-means or hierarchical clustering. The primary objective is to group employees based on similarities in their characteristics. These clustering algorithms partition the dataset into groups (clusters) where individuals within the same cluster share similar traits or behavior patterns. By identifying distinct clusters representing different segments of the workforce, this model helps in understanding the diversity within the employee population and tailoring strategies accordingly.

- Efficacy prediction model:

For predicting the efficacy of employees, we deploy prediction techniques such as Random Forest, XGBoost and LightGBM. The aim of this model is to analyze historical data to detect patterns and make predictions. By accurately predicting employee efficacy, organizations can proactively manage resources and optimize workforce scheduling to meet operational demands.

1.6. Deliverables

- Data source identification
    - Identify all data sources containing relevant employee data. This includes daily efficacy, demographics, worker performance by shift, by day.
    - Ensure data quality and consistency in MySQL database.
- User interaction system that is for worker performance and attrition in the factory can perform the following tasks:
    - Tracks and statistic performance workers performance and efficacy.
    - Cluster types of employees and get characteristics of each group.
    - Predict the worker efficacy in the future based on Machine Learning model.
- Data analysis and insights generation:
    - Analyze worker efficacy based on demographics, health statements, working skills, working performance through statistics.

1.7. Tools and programming language

- Tools: Pycharm or Visual Studio Code, Qt Designer, MySQL Workbench
- Programming: Python, SQL

1.8. Structure of project

The project consists of 4 chapters related to project from general to detail, from concept to implementation from data understanding to project results:

- Chapter 1: Project overview
- Chapter 2: Theoretical Background
- Chapter 3: EDA and Data processing
- Chapter 4: Experimental results
- Chapter 5: Discussion and Future work

Chapter 2. Theoretical Background

2.1. Human resource management

The smart manufacturing industry is witnessing strong convergence between automation systems, big data, and machine learning (ML). This combination opens up new opportunities to optimize production processes and improve operational efficiency. Human resource management (HRM) is no exception, with ML playing a key role in transforming the way talent is recruited, trained, managed and retained.

Some applications of ML in HRM in the smart manufacturing industry can be mentioned as:

- Hiring and recruiting: According to (Sarker & Associates, 2020; Talib & Associates, 2021), ML can automate the screening of individuals, finding the best fit based on qualifications, experience, interests, and other crucial variables. Artificial intelligence (AI)-powered chatbots can respond to often asked queries during the hiring process, improve the applicant experience, and draw in new candidates.
- Develop and evaluate employee performance: HR may create focused training programs and make plans by utilizing predictive analytics with machine learning to detect skill gaps in the workforce. Individualized professional growth for staff members. Machine learning (ML) can be used to improve performance evaluation by analyzing data from several sources, including production output, safety records, quality control measurements, and manager comments. This gives HR a data-driven, unbiased evaluation of employee performance, which aids in more precise and efficient decision-making regarding rewards and promotions. (Chen et al., 2020; Ramesh et al., 2019)
- Benefits and retention: HR can create targeted retention tactics, such as bonuses, increases, improved benefits, or chances for advancement, by using ML algorithms to analyze employee data and detect potential risks of employee turnover. Chatbots with machine learning capabilities can handle benefits enrollment procedures and provide answers to queries regarding employee benefits, freeing up HR staff time for more difficult jobs.

Thanks to those applications, the benefits of applying ML in HRM in smart manufacturing industry:

- Improve recruitment efficiency: ML helps reduce recruitment time and costs, while attracting candidates that best suit the needs of the business.
- Improve employee performance: ML helps identify and train employees more effectively, thereby improving productivity and work quality.
- Minimize risks: ML helps predict and prevent potential problems such as workplace accidents, layoff or workforce shortages.
- Enhance employee engagement: ML helps create a safe, healthy and motivating work environment, thereby enhancing employee engagement and engagement.

2.2. Models

2.2.1. Kmeans

K-means clustering falls under the umbrella of unsupervised learning, where the data is unlabeled and the goal is to uncover inherent groupings within it. The theoretical foundation relies on minimizing the within-cluster variance, achieving a compact and well-separated cluster structure.

Mathematically, K-means iteratively refines a set of k centroids, which represent the center points of each cluster. Each data point is assigned to the closest centroid, based on a distance metric like Euclidean distance. Once all points are assigned, the centroids are recomputed as the mean of their assigned points. This process continues until a convergence criterion is met, often when the centroids stabilize or the change in within-cluster variance falls below a threshold [1].

The K-means algorithm follows these iterative steps:

- Initialization: Select k initial centroids. These can be chosen randomly or by using more sophisticated methods like the K-means++ algorithm.
- Assignment: Assign each data point to the nearest centroid. The distance between a data point and a centroid is typically measured using Euclidean distance.

- Update: Recompute the centroids by calculating the mean of all data points assigned to each centroid.
- Convergence: Repeat the Assignment and Update steps until the centroids no longer change significantly or the change in within-cluster variance is below a predefined threshold.

The optimality of K-means is an ongoing area of research. Hartigan and Wong (1979) established theoretical guarantees on the convergence of the algorithm to a local optimum, proving it achieves a minimized squared error objective under specific conditions. However, K-means is sensitive to the initial centroid placement and the number of clusters chosen (k). Arthur and Vassilvitskii (2007) proposed a seeding strategy to improve the initial centroid selection, leading to better convergence properties.

2.2.2. Random Forest

Random Forest is a widely-used machine learning algorithm employed for classification and regression tasks. Introduced by Leo Breiman in 2001, it quickly emerged as a potent tool for addressing complex problems in data science and machine learning [1]. Random Forest operates by utilizing a collection of decision trees to make final predictions. Each decision tree in the ensemble independently learns patterns from the data and contributes to the overall prediction. By combining the predictions from multiple trees, Random Forest enhances both the accuracy and generalization capability of the model, making it robust to noise and outliers in the data.

Random Forest is a popular machine learning method that operates based on the "bagging" (bootstrap aggregating) technique. This technique randomly samples with replacement to create multiple subsets of data from the original dataset (Liaw, A., & Wiener, M 2002). Each decision tree in the forest is then trained on a different subset of data. Subsequently, each subset is used to construct a decision tree, and these trees are trained independently. During the tree construction process, a small number of features are randomly selected at each node to find the best feature for splitting the data. Assuming there are M features in the dataset:

- At each node, randomly select m features ($m \ll M$).
- Choose the best feature from the selected m features to split the data.

Predictions from each decision tree are combined to create the final prediction of the Random Forest. The method of combining predictions depends on the type of problem:

- For classification tasks, the majority voting method is used. The final prediction is the label predicted by the majority of trees.
- For regression tasks, the average of predictions from the trees is used. The final prediction is the average of predictions from all trees.

Random Forest presents numerous advantages over alternative algorithms. Firstly, it demonstrates robustness in managing data heterogeneity and missing values (Cutler, D and et, 2007). This robustness allows it to handle datasets with varying data types and incomplete information effectively. Secondly, Random Forest offers a feature importance evaluation mechanism, enabling the identification of significant variables within the model Díaz-Uriarte, R., & De Andrés, S. A. (2006). This feature is particularly valuable for understanding the underlying relationships and influential factors driving predictions. Lastly, Random Forest exhibits excellent scalability, making it suitable for deployment on large datasets without sacrificing performance. Its ability to parallelize computations and distribute the workload efficiently contributes to its scalability, ensuring efficient processing even with vast amounts of data.

2.2.3. XGBoost

XGBoost (eXtreme Gradient Boosting) is a highly regarded machine learning algorithm widely used for classification and regression tasks. Introduced by Tianqi Chen and Carlos Guestrin in 2016, XGBoost quickly established itself as an effective tool for addressing complex issues in data science and machine learning. The algorithm works by sequentially adding decision trees to a model to minimize a specific loss function. Each tree in the ensemble learns patterns from the data and contributes to the overall prediction. By combining the predictions from multiple trees, XGBoost enhances both the accuracy and generalization capabilities of the model, making it robust against noise and outliers in the data.

The mathematical formulation of XGBoost involves optimizing a regularized objective function. The objective combines the loss function LL (e.g., mean squared error for regression, log loss for classification) and a regularization term $\Omega\Omega$ to control the complexity of the model and prevent overfitting:

$$\text{Obj(t)}=\sum\sum_{i=1}^{n} L\left(y_i, y_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t)$$

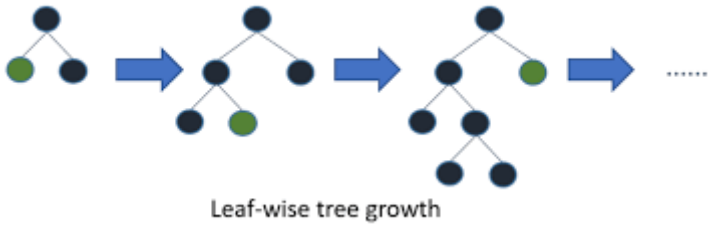
where y_i represents the true label, , $y_i^{(t-1)}$is the predicted value at iteration $t-1$, f_t is the new tree added at iteration t, and Ω accounts for regularization.

Numerous studies have demonstrated the effectiveness of XGBoost. In "XGBoost: A Scalable Tree Boosting System," Chen and Guestrin showcased the superior performance of XGBoost on various benchmark datasets, attributing its success to advanced techniques like parallel processing and tree pruning. Friedman's foundational work on gradient boosting (2001) provided crucial insights into the theoretical underpinnings and practical applications of boosting algorithms, laying the groundwork for the development of XGBoost. He et al. (2018) explored hyperparameter optimization techniques, including grid search and Bayesian optimization, to fine-tune XGBoost, demonstrating significant performance improvements through effective parameter tuning. Additionally, XGBoost supports various objective functions and evaluation metrics, allowing customization for different types of predictive modeling tasks. In summary, XGBoost is a versatile and powerful algorithm that has proven effective across numerous domains, supported by extensive research and practical success in data science and machine learning applications.

2.2.4. LightGBM

Before discovering LightGBM, we should know that Gradient Boosting is a machine learning ensemble technique that associates the predictions of multiple weal learners, typically decision tree, sequentially. LightGBM (Light Gradient Boosting Machine), which is a Gradient Boosting framework that uses tree-based learning algorithms, is officially released in January 2017 by Microsoft Gradient Boosting is a machine learning ensemble technique that associates the predictions of many weal learners, usually decision trees, sequentially. This is important to know before we learn about LightGBM. Microsoft officially launched LightGBM (Light Gradient Boosting Machine), a Gradient Boosting framework that makes use of tree-based learning techniques, in January 2017 (Ju, Y. et al., 2019). According to the research of Zhang et al. (2019), LightGBM has been demonstrated to be up to 20 times faster than on the same data, compared to the XGBoost, implementation of Gradient Boosting.

It is predicated on decision trees, which are intended to decrease memory utilization and increase model efficiency. It combines a number of cutting-edge methods, such as Gradient-based One-Side Sampling (GOSS), which maximizes training time and memory consumption by retaining instances with large gradients only during training. Furthermore, LightGBM uses methods based on histograms to generate trees more effectively. These methods increase LightGBM's efficiency and offer it an advantage over competing gradient boosting frameworks and competing optimizations include leaf-wise tree development and effective data storage formats. (Ke, G. et al., 2017).



[1]

Figure 2.1. Leaf-wise tree growth

LightGBM is a modern variation of the GB algorithm. It enhances the algorithm's efficiency and scalability while maintaining its strong performance. Seven hyperparameters that influence the performance of the LightGBM classifier were optimized within the predefined ranges recommended by the package manual: number of leaves ("num_leaves", 30−500), number of feature bins ("max_bin", 250−500), number of iterations ("n_estimators", 50−900), number of samples in one leaf ("min_child_samples", 30−500), model depth ("max_depth", 5−12), learning rate ("learning_rate", 10^{-6}−10^{-1}), and fraction of data used for training ("bagging_fraction", 0.8) (Zhang, J. et al., 2019).

In the study of Yu Wang (2007) and Xiaojun et al. (2018), we can find some advantages for this algorithm like:

- Quick training speed: To expedite the training process, LightGBM buckets continuous feature values into discrete bins.

[1] Source: https://vtitech.vn/xgboost-bai-2-toan-canh-ve-ensemble-learning-phan-2/

- Low memory consumption: To save memory, discrete values are used in place of continuous values.
- Greater accuracy: By using a leaf-wise split strategy, it can create considerably more complicated trees.
- Support for parallel learning: It allows for both data parallel and feature parallel operations.

Chapter 3. Data Processing and Modelling

3.1. Data understanding

This dataset offers a rich view into the daily operations of a factory over a year and a half. It encompasses data for 508 workers, though the total number of individuals appearing reflects employee turnover (687). Daily recordings capture both routine occurrences, such as attendance and worker efficacy levels, alongside significant one-time events. These one-time events could include accidents, employee departures, or the introduction of new hires. By providing a comprehensive picture of both the everyday and the exceptional, this dataset allows for in-depth analysis of the factory's workforce dynamics and performance.

This level of detail allows us to explore various aspects of the factory's operations. We can examine trends in worker performance over time, identify factors that influence employee efficacy, and investigate the relationship between attendance and productivity. Additionally, the data on one-time events can shed light on potential causes of attrition and areas for improvement in onboarding processes or safety protocols. By delving into this comprehensive dataset, we can gain valuable insights into how to optimize the factory's workforce management and overall performance.

3.2. Data description

The dataset describes records for tracking employee behaviors at work, containing three main types of fields: those related to the subject (employee), those related to the supervisor, and those related to specific events.

- Subject-related fields: These fields detail information about an employee involved in a workplace event. It includes basic identifiers and demographics (ID, name, age, sex, shift, team, role) and deeper, hidden psychological traits (health, commitment, perceptiveness, dexterity, sociality, morality, strength, open-mindedness, workstyle) that are not directly accessible to supervisors but influence the employee's behaviors.
- Supervisor-related fields: These fields provide information about the supervisor who observed and recorded the employee's behavior. It includes the supervisor’s

ID, name, age, gender difference with the subordinate, role, and hidden traits (commitment, perceptiveness, morality).

- Event-related fields: These fields describe the specific workplace event involving the employee. This includes the timing of the event (date, week, day, weekday number and name), the type of behavior displayed by the employee, and the reason behind any resignation or termination if applicable. There are hidden fields showing the true nature or level of the behavior which are generally invisible in the system. Supervisors' records include the type of behavior observed, reasons for termination, estimated efficacy levels, and any additional notes. The accuracy of the supervisor's observation is indicated in the record (True Positive for accurate recording and False Negative when a behavior goes unrecorded).

3.3. Check duplicate and missing data

Table 3.1. Missing data in columns

Column	Null count
sub_ID	0
sub_lname	0
sub_age	0
sub_sex	0
sub_shift	0
sub_team	0
sub_role	0
sub_coll_IDs	812
sub_colls_same_sex_prtn	991
sub_health_h	0
sub_commitment_h	0

Column	Null count
sub_perceptiveness_h	0
sub_dexterity_h	0
sub_sociality_h	0
sub_goodness_h	0
sub_strength_h	0
sub_openmindess_h	0
sub_workstyle_h	0
sup_ID	812
sup_fname	812
sub_lname	812
sub_age	812
sup_sub_age_diff	812
sup_sex	812
sup_role	812
sup_commitment_h	812
sup_perceptiveness_h	812
sup_goodness_h	812
event_date	0
event_week_in_series	0
event_day_in_series	0
event_weekday_num	0
event_weekday_name	0

Column	Null count
behav_comptype_h	102
behav_cause_h	411871
actual_efficacy_h	220291
record_comptype	3938
record_cause	411846
recorded_efficacy	220676
recorded_note_from_sup	393359
record_conf_maxtrix_h	390233

We have also checked the duplicate records in the dataset but there is no duplicate value.

3.4. Data Modelling:

The model we use for training don't have any missing data so we don't have to fill it. In this part, I will describe more about the process of data modelling in this project.

3.4.1. Efficacy prediction process

For efficacy prediction, the code establishes a preprocessing pipeline utilizing scikit-learn's ColumnTransformer, a pivotal step in preparing data for machine learning models. This pipeline incorporates a StandardScaler, which standardizes numerical features by scaling them to have a mean of zero and a standard deviation of one. Standardizing numerical features is essential because it ensures that all features contribute equally to the model's performance, preventing features with larger ranges from dominating the model's learning process. The pipeline includes only the numerical columns selected by the user, thereby ensuring that the appropriate features are processed and used for training the model.

```
self.preprocessor = ColumnTransformer(
    transformers=[
        ('scaler', StandardScaler(), numerical_columns)
    ], remainder='passthrough'
)
```

Figure 3.1. Standard Scaler for data processing

Next, we got the tune_hyperparameters method plays a vital role in model tuning and evaluation. This method employs randomized search cross-validation to identify the optimal hyperparameters for a given model. Hyperparameters are the settings that must be defined before the learning process begins and significantly impact the model's performance. The tune_hyperparameters method modifies these settings in accordance with the type of model employed, be it a Random Forest, XGBoost, or LightGBM model. By exploring a multitude of hyperparameter combinations, this method ensures that each model is optimized for optimal accuracy and efficiency, thereby enhancing the overall performance of the machine learning task.

```
def tune_hyperparameters(self, model, X, y, cv=5):
    if isinstance(model, RandomForestRegressor):
        param_grid = {
            'n_estimators': [100, 200, 300],
            'max_depth': [None, 5, 10],
            'min_samples_split': [2, 5, 10]
        }
    elif isinstance(model, XGBRegressor):
        param_grid = {
            'max_depth': [3, 5, 7],
            'n_estimators': [100, 200, 300],
            'learning_rate': [0.01, 0.1, 0.3]
        }
    elif isinstance(model, LGBMRegressor):
        param_grid = {
            'num_leaves': [31, 63, 127],
            'learning_rate': [0.01, 0.1, 0.3],
            'n_estimators': [100, 200, 300]
        }
    else:
        return model
```

Figure 3.2. Search for optimize parameters

The evaluate_models method is responsible for assessing the performance of three different regression models: Random Forest, XGBoost, and LightGBM. It begins by tuning the hyperparameters for each model using the tune_hyperparameters method. Once the optimal settings are identified, the method constructs a scikit-learn pipeline that integrates the preprocessing steps and the tuned model. The comprehensive pipeline is then fitted to the data and evaluated using performance metrics, including mean

squared error (MSE) and R-squared (R2). MSE measures the average squared difference between the predicted and actual values, while R2 indicates how well the model's predictions approximate the real data points. By comparing these metrics, the evaluate_models method identifies the best-performing model, ensuring that the most effective and accurate model is selected for the task at hand. This approach not only enhances the reliability of the predictions but also optimizes the model's capacity to generalize to new, unseen data.

```python
def evaluate_models(self, X, y):
    models = [
        ("Random Forest", RandomForestRegressor()),
        ("XGBoost", XGBRegressor()),
        ("LightGBM", LGBMRegressor())
    ]

    best_model = None
    best_score = float('-inf')

    for name, model in models:
        tuned_model = self.tune_hyperparameters(model, X, y)
        pipeline = Pipeline([
            ('preprocessor', self.preprocessor),
            ('regressor', tuned_model)
        ])
        pipeline.fit(X, y)
        y_pred = pipeline.predict(X)
        mse = mean_squared_error(y, y_pred)
        r2 = r2_score(y, y_pred)
        print(f"{name} - MSE: {mse:.2f}, R2: {r2:.2f}")
        if r2 > best_score:
            best_model = pipeline
            best_score = r2
    return best_model
```

Figure 3.3. Evaluate to find out best model for using

3.4.2. Employee clustering process

The process starts by connecting to a MySQL database to retrieve relevant data, depending on whether the focus is on laborers or supervisors. For laborers, attributes such as age, health, commitment, perceptiveness, dexterity, sociality, goodness, strength, open-mindedness, the age difference between worker and supervisor, and actual efficacy are retrieved. For supervisors, the data includes attributes like age, commitment, perceptiveness, goodness, and actual efficacy. This data is organized into a Pandas DataFrame, with features separated from the target variable, which is the actual efficacy.

A critical part of this process involves evaluating the importance of each feature using permutation importance, which helps in identifying the most relevant features for

clustering. The negative mean squared error (neg_mean_squared_error) metric plays a vital role here. Initially, a KMeans model with three clusters is fitted to the data, which includes all the selected features. Permutation importance works by randomly shuffling the values of each feature, one at a time, and observing how this shuffling affects the model's performance. The performance is measured using the negative mean squared error, which calculates the average squared difference between the model's predictions and the actual values. A higher impact on the model's performance (i.e., a more significant increase in error) when a feature is shuffled indicates that the feature is crucial for the model's accuracy. This means that the model heavily relies on this feature to make accurate predictions. By ranking features based on the increase in error caused by shuffling, the code identifies the top three features with the highest importance, ensuring that the clustering process focuses on the most impactful attributes.

```
def evaluate_feature_importance_for_clustering(self, X, y):
    model = KMeans(n_clusters=3, random_state=0)
    model.fit(X)
    result = permutation_importance(model, X, y, scoring='neg_mean_squared_error', n_jobs=-1)
    sorted_indices = result.importances_mean.argsort()[::-1]
    sorted_features = X.columns[sorted_indices]
    return sorted_features[:3]
```

Figure 3.4. Evaluate feature importance by permutation

Once the key features are identified, the data is grouped by unique worker or supervisor IDs, and the mean values of these features are calculated to create a summarized dataset. The optimal number of clusters is then determined by evaluating the inertia for different cluster counts, ranging from one to ten, and identifying the point where additional clusters provide diminishing returns. A KMeans model with this optimal number of clusters is fitted to the data, and each data point is assigned to a cluster, stored in a new column called 'efficacy_cluster'. Finally, the clustered data is visualized to help interpret the results, and any errors encountered during the process are handled gracefully by displaying a warning message and printing error details for debugging. This comprehensive approach ensures meaningful segmentation of workers or supervisors based on their attributes and efficacy, providing valuable insights into their performance patterns.

```python
def find_optimal_clusters(self, data, max_clusters=10):
    inertias = []
    for k in range(1, max_clusters + 1):
        kmeans = KMeans(n_clusters=k, random_state=0)
        kmeans.fit(data)
        inertias.append(kmeans.inertia_)
    optimal_clusters = np.argmin(np.diff(inertias)) + 3
    return optimal_clusters
```

Figure 3.5. Find optimal number of clusters

Chapter 4. Experimental Results

4.1. Overview about the App

This app is aimed to manage employees as well as view statistics about employee information and performance. In addition, the app also has a number of machine learning models for predicting employee performance as well as being able to cluster employees with their respective roles.

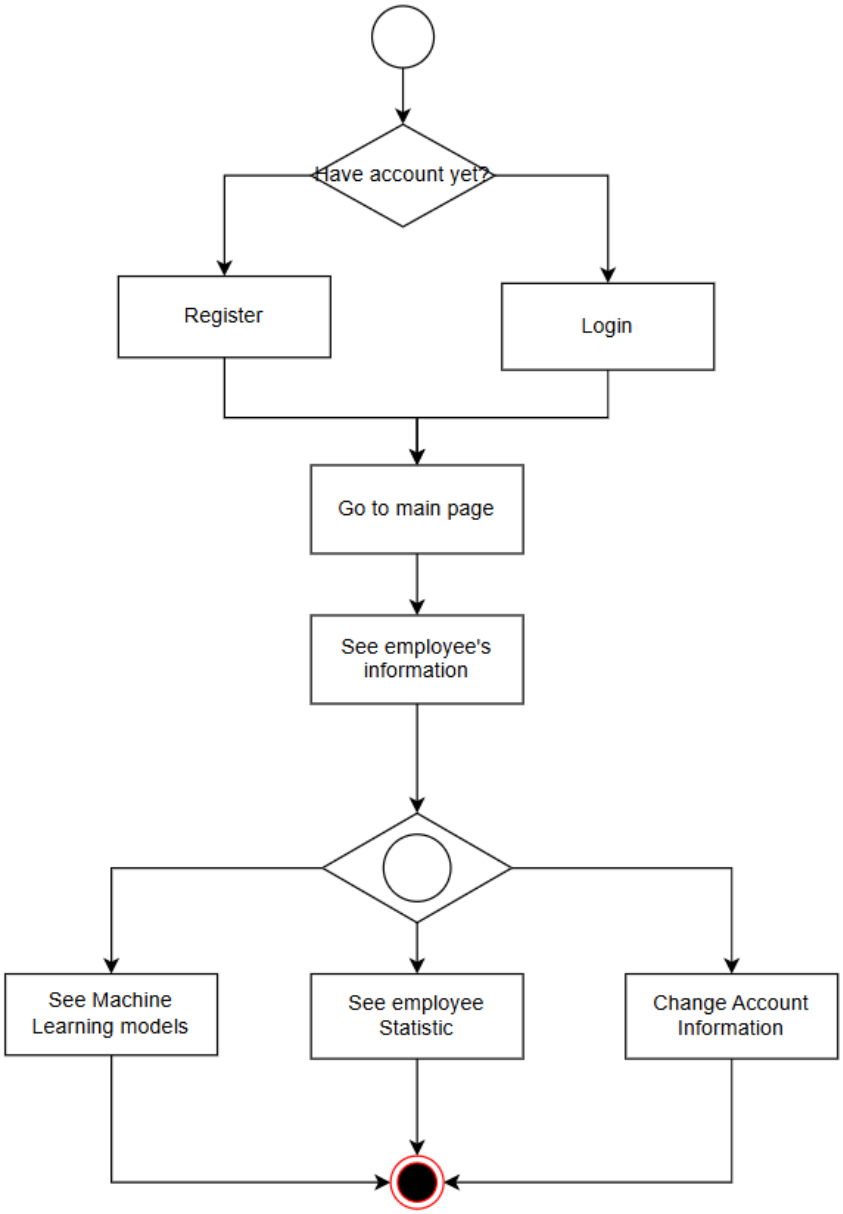


Figure 4.1: General flow of the app

The main features of the app:

- Log in, register an account.

- Main page: Shows employee information according to their role, and also contains buttons to lead to other pages.
- Machine learning page: Contains models about cluster analysis and Efficiency prediction.
- Statistic page: Contains statistical charts related to employees and their performance.
- Change Information: Change information about login account.

These features will be described in detail in the following sections.

4.2. Register and Login page

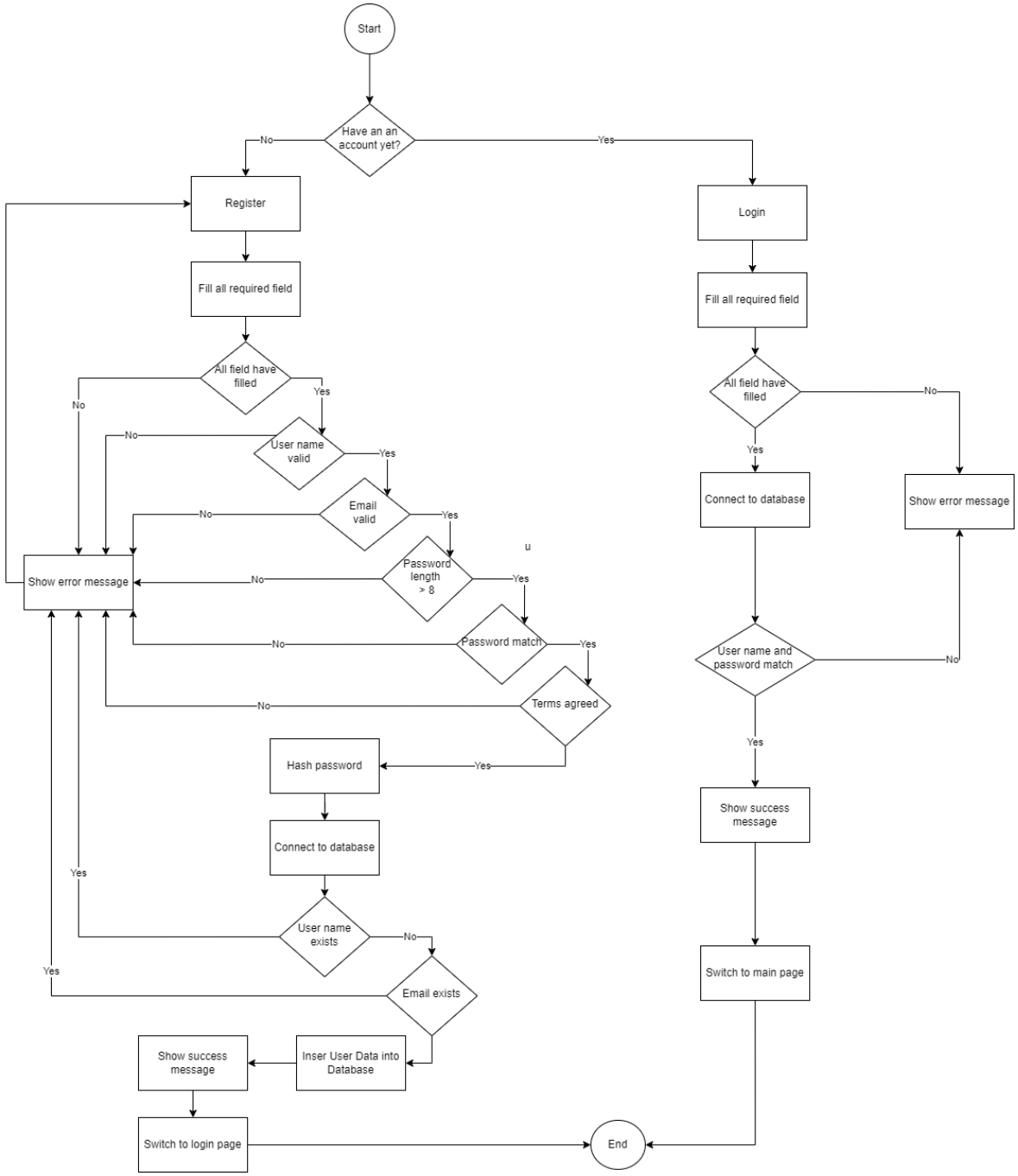


Figure 4.2. BPMN Login and Register flow

4.2.1. Register

When the MyApp interface starts running, a registration window will appear as shown in the picture.

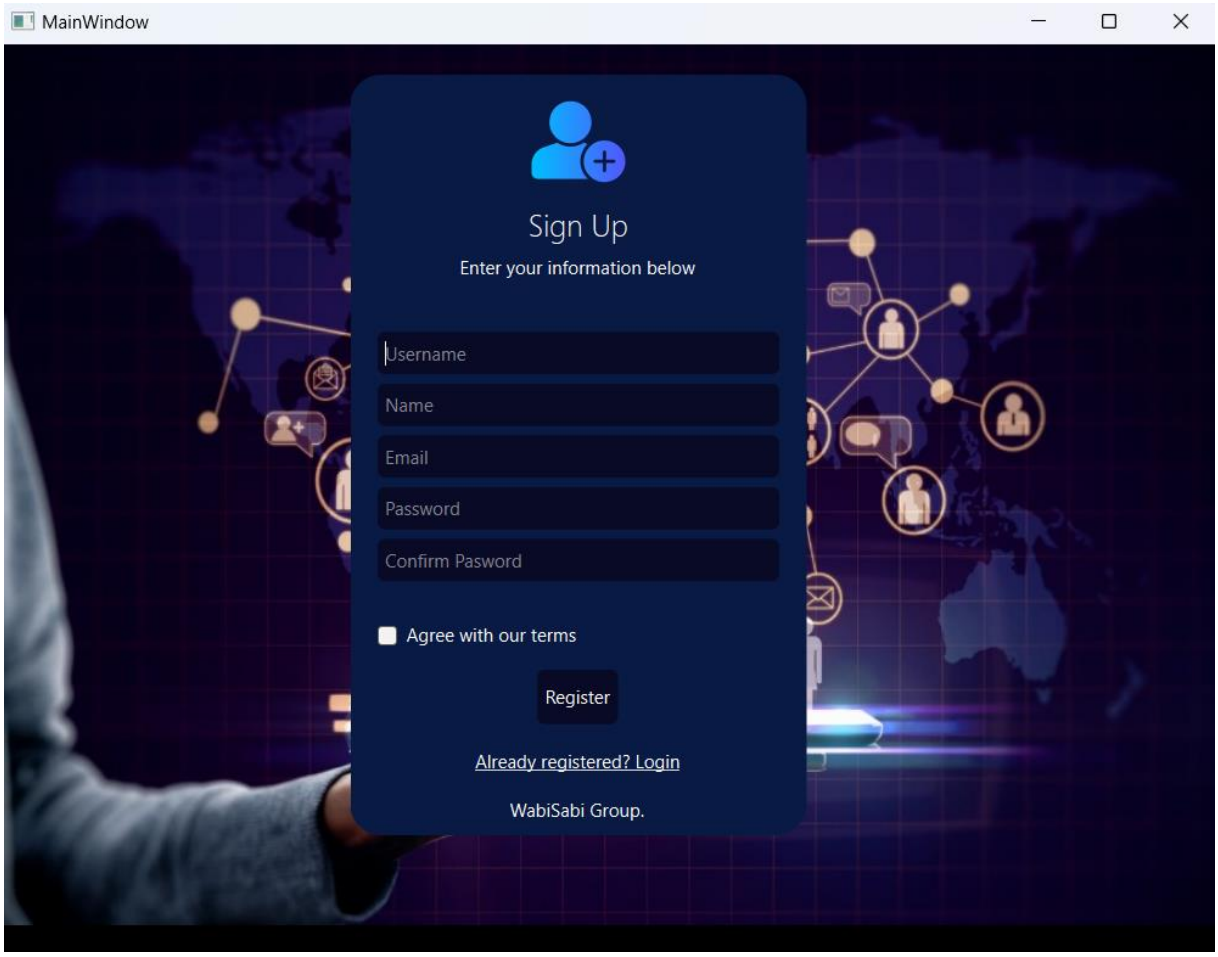


Figure 4.3. Sign Up interface

Subsequently, the user will be prompted to enter information into the blank fields. If the information provided is incomplete, a warning pop-up will appear as shown below.

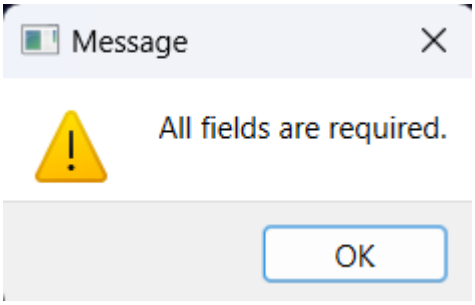


Figure 4.4. Message show when still have blank fields

Next, after the required information has been fully entered, the front end will validate whether the user inputs meet the specified formats. This includes verifying that the Username does not contain special characters, the Email is correctly formatted, the Password is at least 8 characters long, and the entered Passwords match. Illustrative images are presented below:

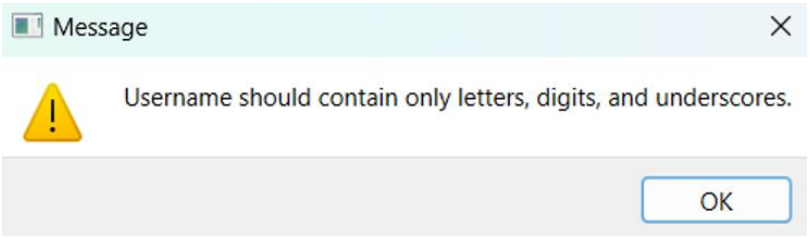


Figure 4.5. Message show when Username invalid

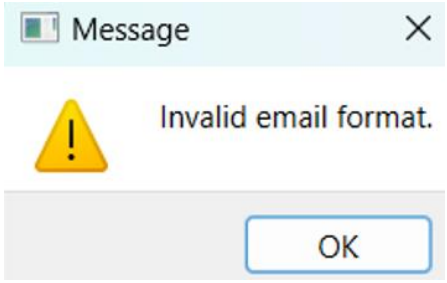


Figure 4.6. Message show when Email invalid

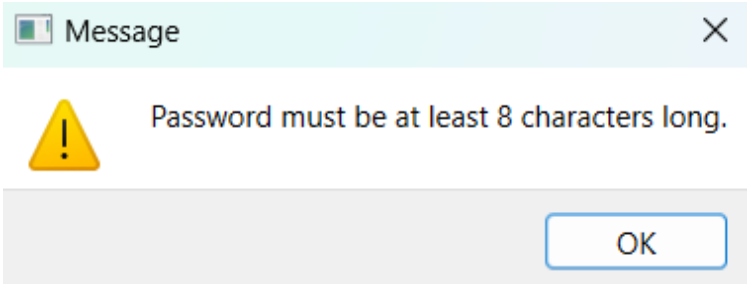


Figure 4.7. Message show when Passwords less than 8 characters

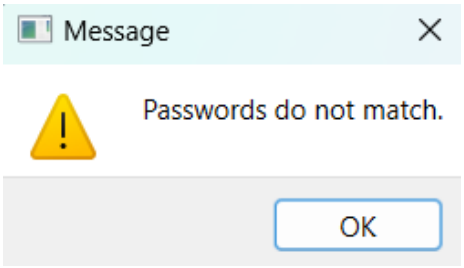


Figure 4.8. Message show when Passwords not match

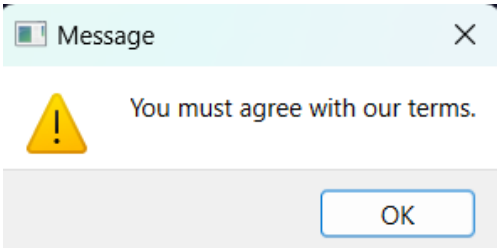


Figure 4.9. Message show when the Our terms wasn't click

Then the system will then connect to the Database to verify whether the registered Username and Email already exist. If both conditions are met, a success message will be displayed, and the interface will transition to the Sign-in screen.

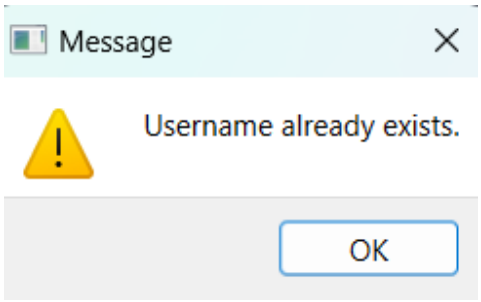


Figure 4.10. Message show when Username already exists

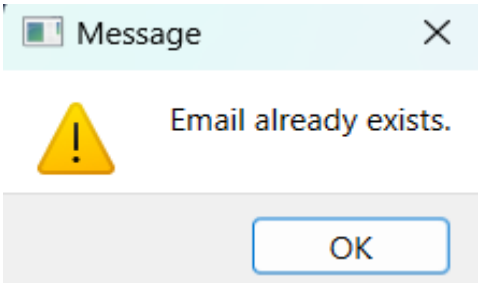


Figure 4.11. Message show when Email already exists

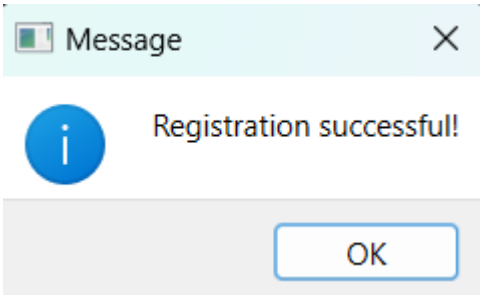


Figure 4.12. Message show when registration successful

4.2.2. Login

After the user completes the registration or clicks the sign-in button, they will be redirected to the sign-in interface as shown below.

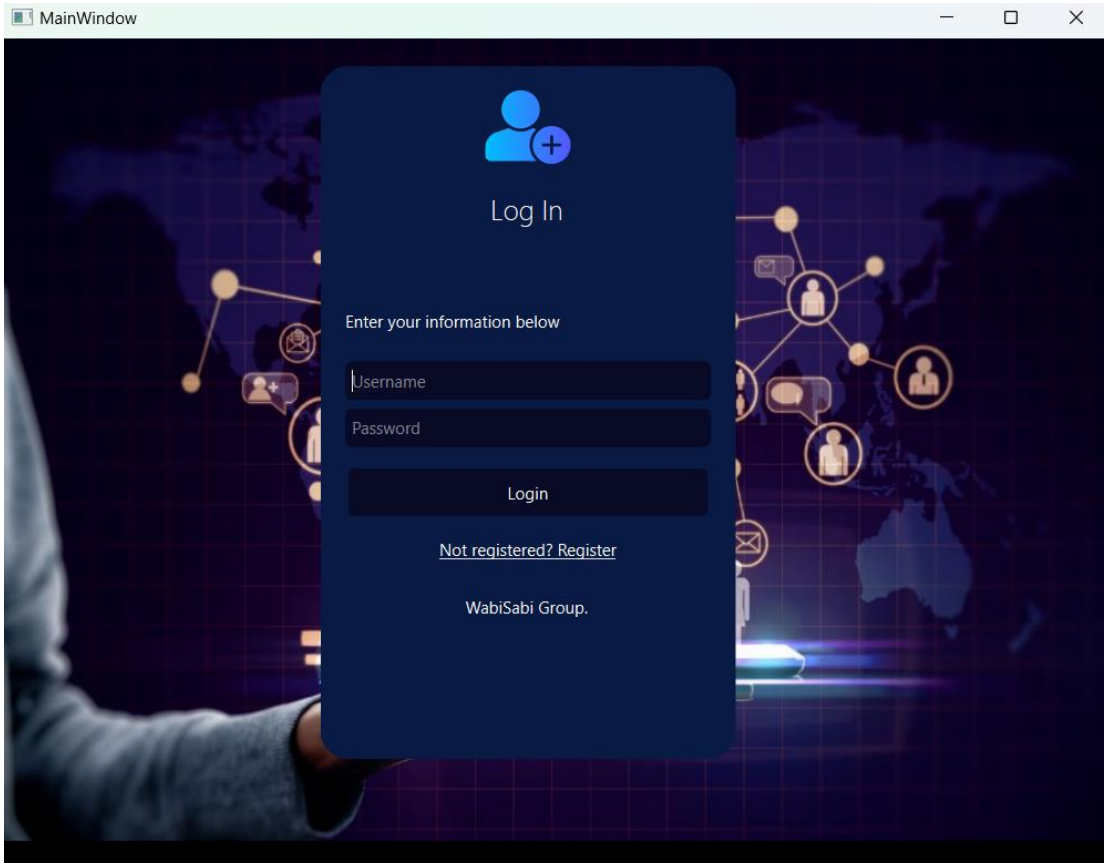


Figure 4.13. Login interface

Similar to the Sign-up process, the user must enter both the Username and Password to sign in; otherwise, an error message will be displayed.

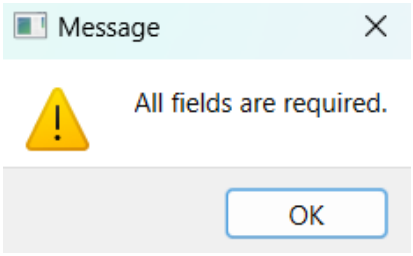


Figure 4.14. Message show when Field are not filled

If either the Username or Password is entered incorrectly, the following error message will be displayed.

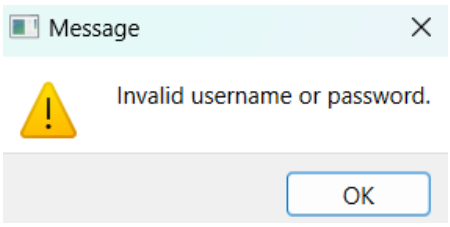


Figure 4.15. Message show when Username ỏ password incorrect

Upon meeting the aforementioned conditions, a success message will be displayed, indicating successful login, and the user will be directed to the main interface of the software.

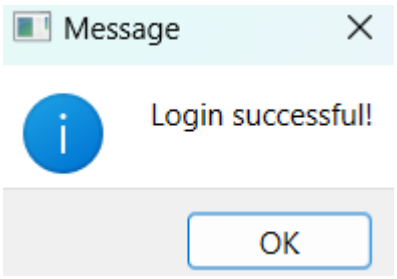


Figure 4.16. Message show when login successful

4.3. Main page

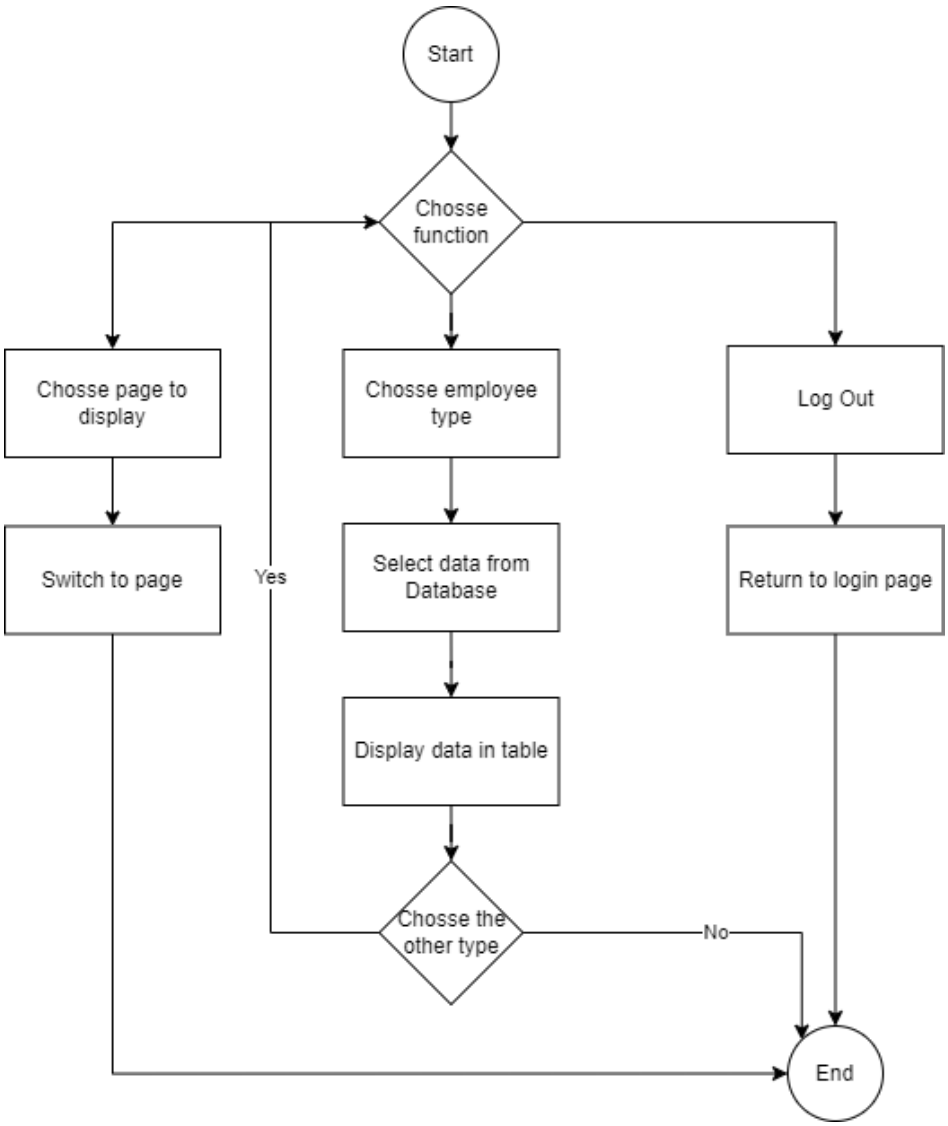


Figure 4.17. BPMN of Main page flow

On this Main page, users can have an overview of employee-related information through a table section. In this content, users can optionally select the type of employee they want to view through a Combo box, as shown in the image above, with options like "Labor" or "Supervisors".

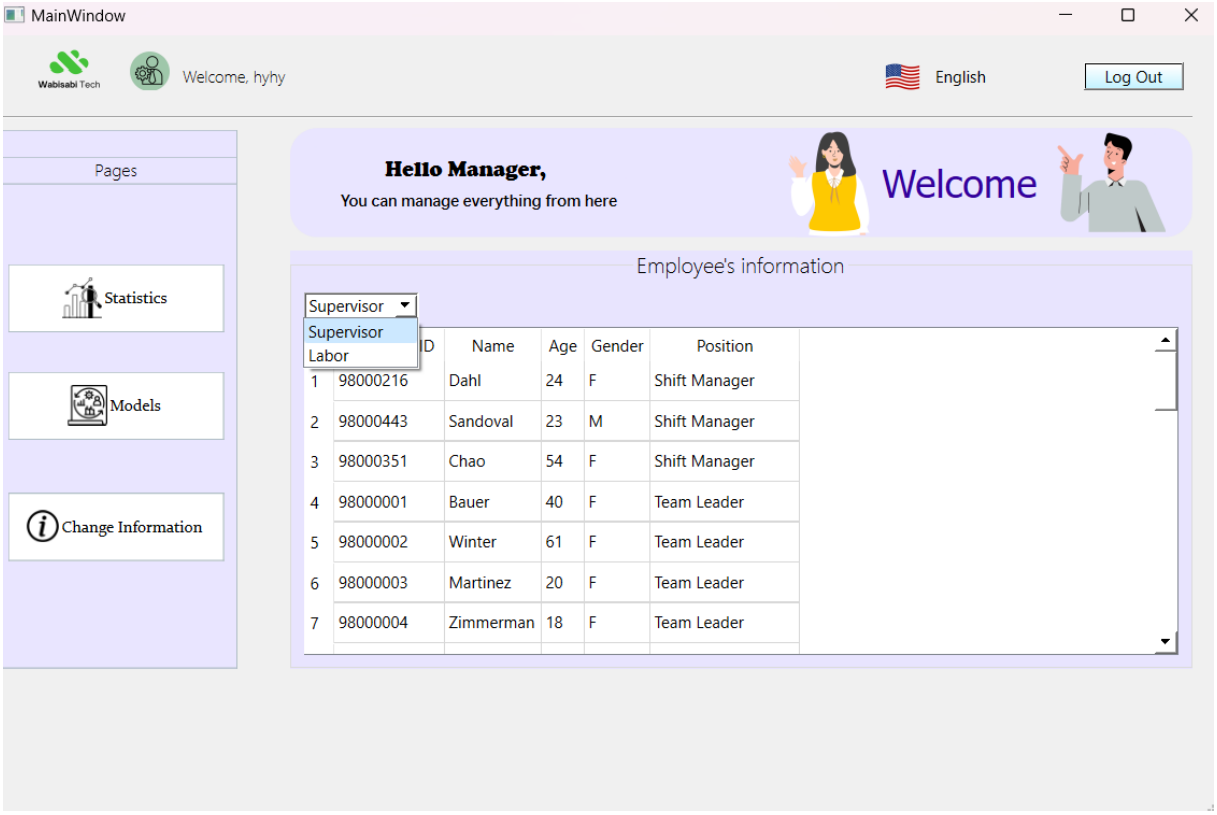


Figure 4.18. Main page of app

Next, in the "Pages" section on the left, users can select different pages to view specific content. For example, the "Statistic" page displays statistics related to sales performance or employee demographics. The "Models" page allows users to select input variables to review model predictions. Finally, on the "Change Information" page, users can update their account details, including Username and Password.

Subsequently, if users wish to log out and switch to a different account, they can use the "Log Out" button located at the top right corner of the interface to perform this action.

4.4. Change information

On the "Change Information" page, users can change their "Name" and "Password," while other displayed information is fixed and cannot be modified. With that, the condition still applied for this filed so if password length is below 8, the warning

message box will pop up and inform user the text “Password must be at least 8 characters long.”.

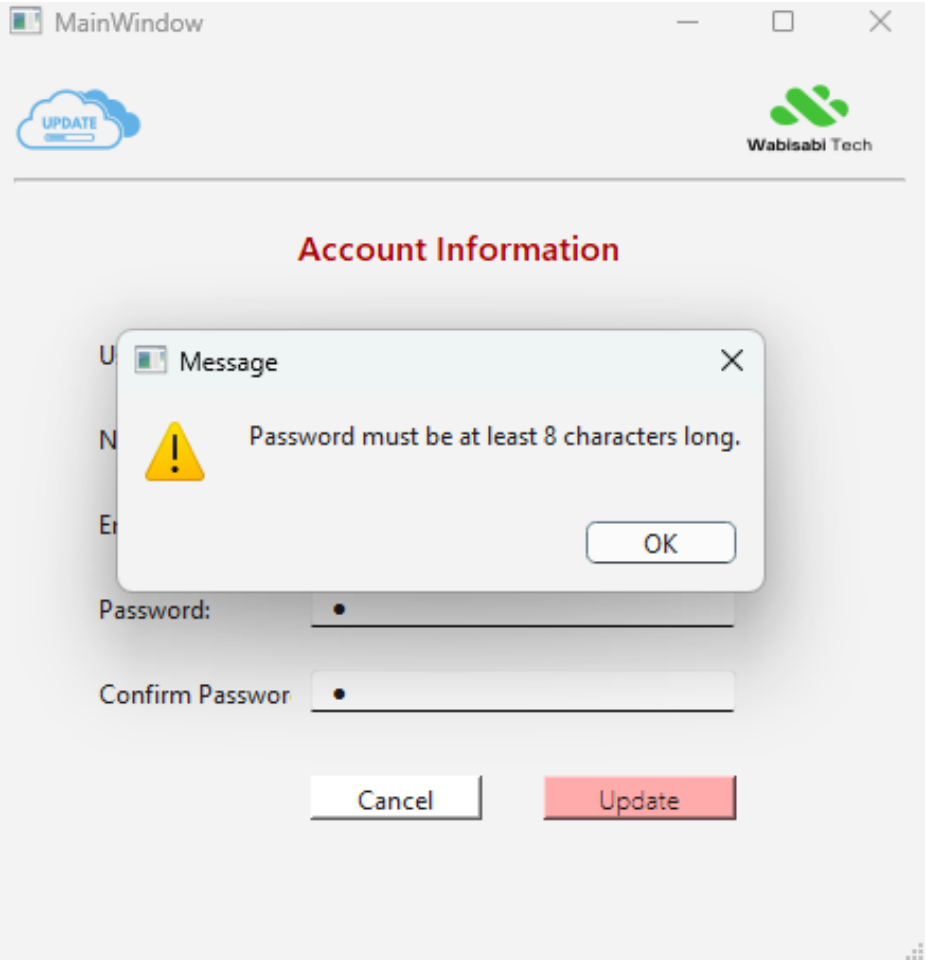


Figure 4.19. Warning message box: password length must be greater than 8

If the information entered by the user meets the specified conditions, a message will appear indicating that the information has been successfully updated.

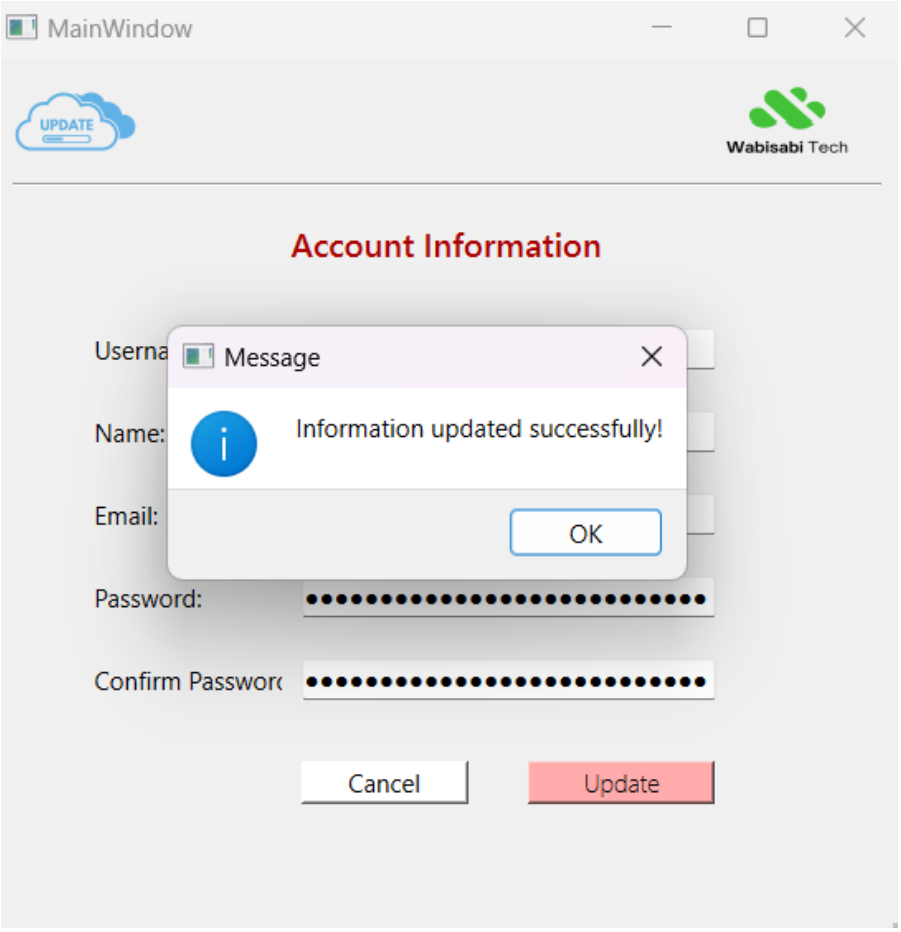


Figure 4.20. Message show when information updated successfully

4.5. Statistics page

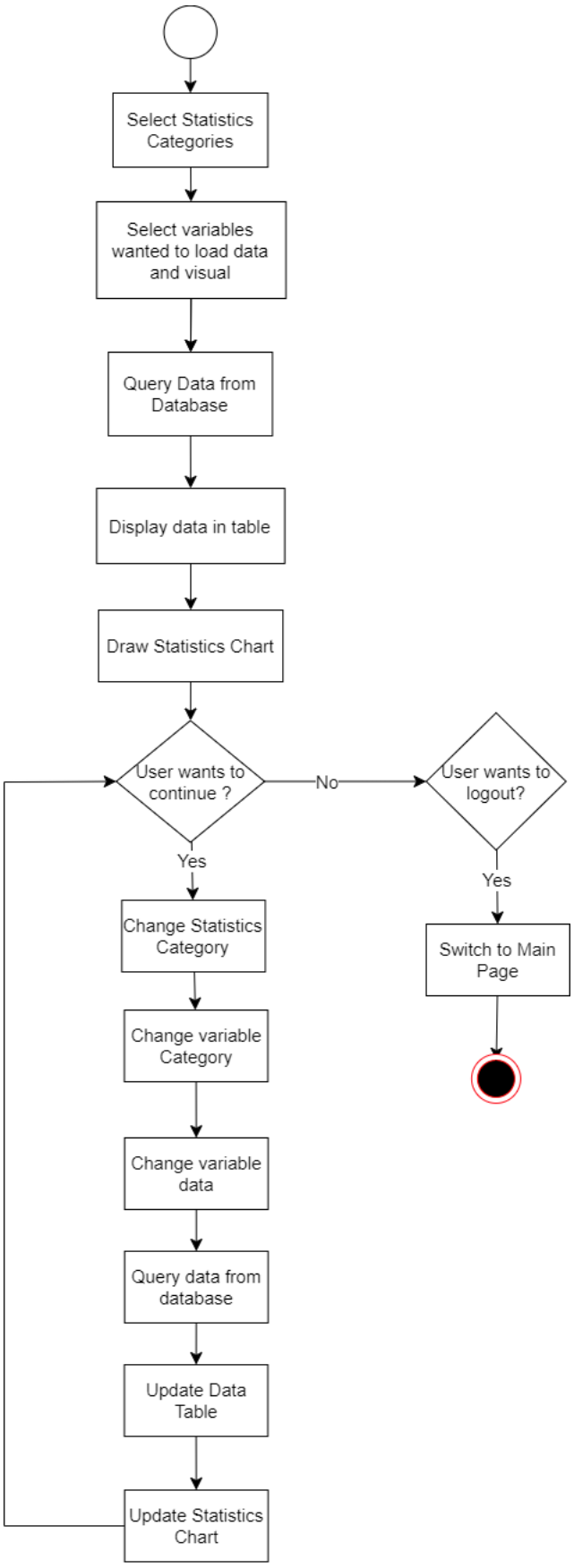


Figure 4.21. BPMN of statistics page

With this diagram, Statistics page showed features for loading and visualizing data. After accessing this page, beside button Log out, team logo, language, it shows two boards in user interface. Upper board through Table Widget in Qt Designer presents loaded data and lower board through bigger Frame draws statistics chart of data. Data is chosen in categories (Employee and Performance), then users need to variables in each category which is presented like Item lists for users to choose one of them. In Employee, there are statistics about distribution of gender, distribution of age, average worker in shifts by gender, top 5 resignation reasons. Besides, in Performance, some variables you need to select such as: average efficacy of each work group, average efficacy in groups with below – average and above – average health, average efficacy in groups with below – average and above- average goodness, top 5 supervisors have most worker resignation.

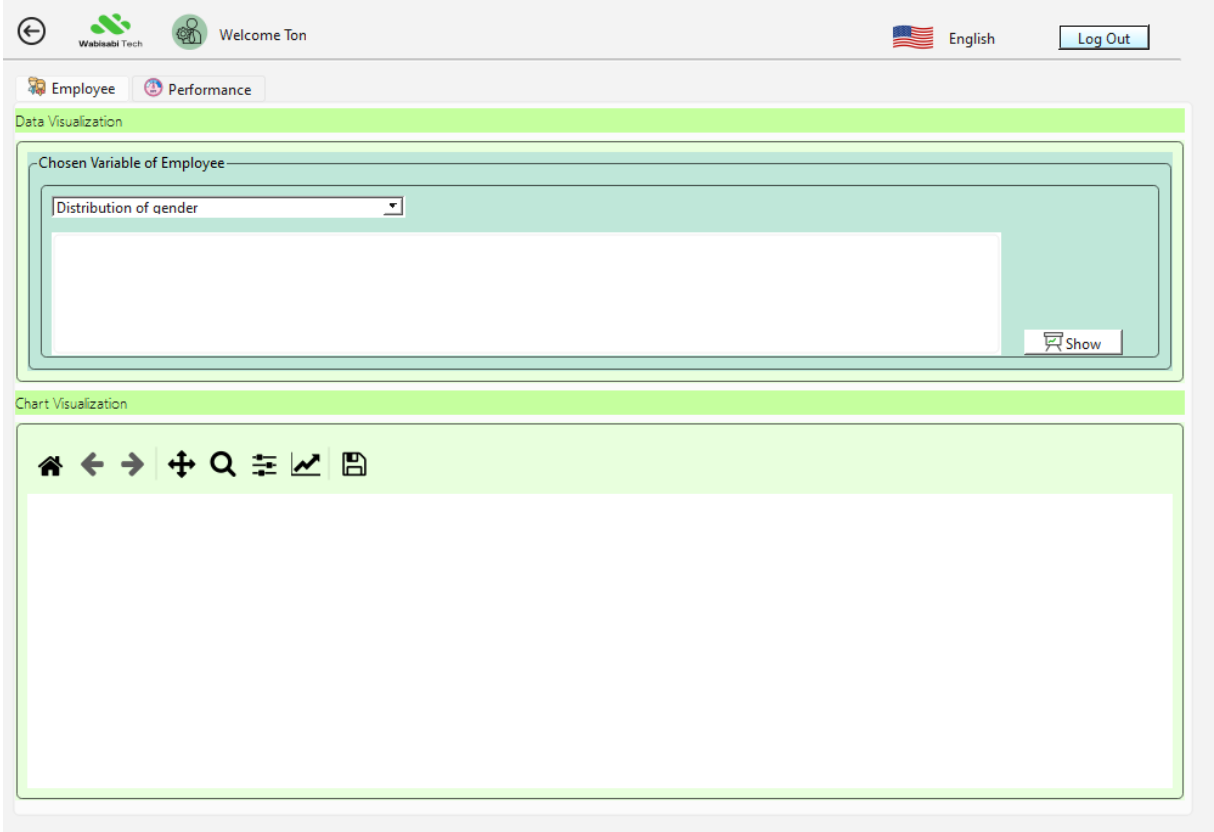


Figure 4.22. Overall view of statistics page

In the green background, Statistics page is presented with friendly user interface for users to perform Statistics on two blank boards about data related to Employee and Performance for us to analyze.

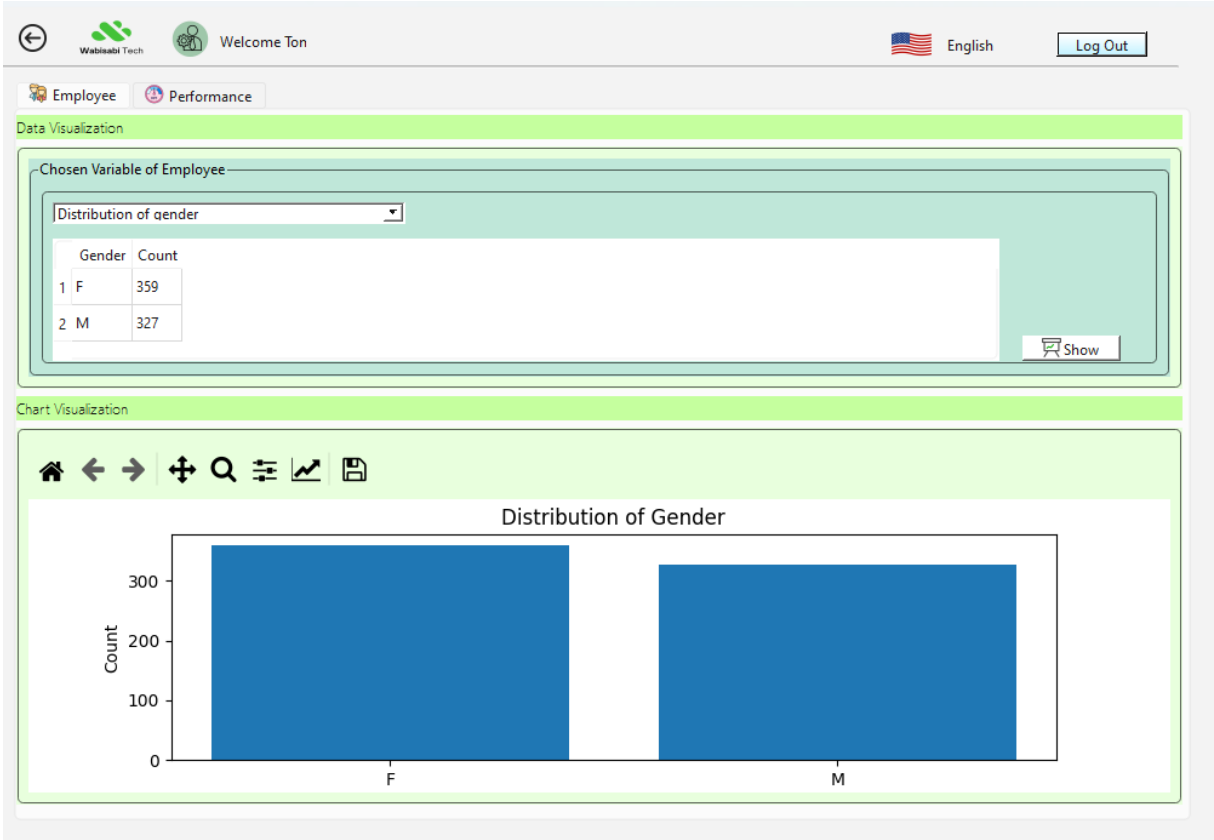


Figure 4.23. View of table widget and visualization

After choosing department and variable, user needs to click button "Show", program start to query data which is selected by user from the database, and load data table into upper board, then draw statistics chart. This iterative process continues to perform until user want to go against into Main Window, they must click button "Log Out".

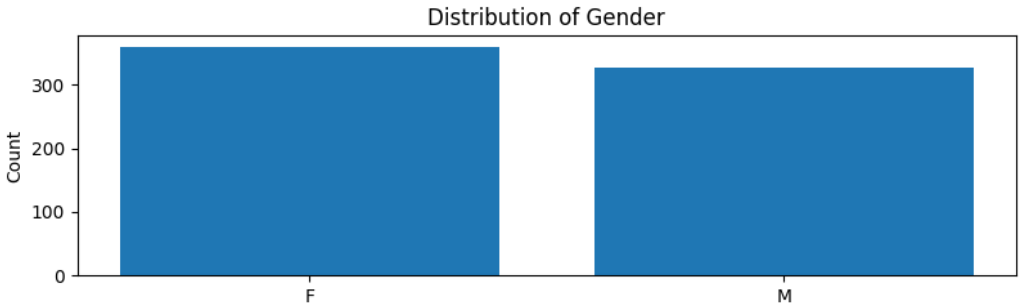


Figure 4.24. Distribution of Gender

Next, from visualization chart, we can analyze data to understand information of Human Resource. The first chart is distribution of Gender showing the amount of female and male employees. The number of Female is higher than the number of Male and not

much different, about 10-30 people. It describes human resources gender is distributed quite evenly.

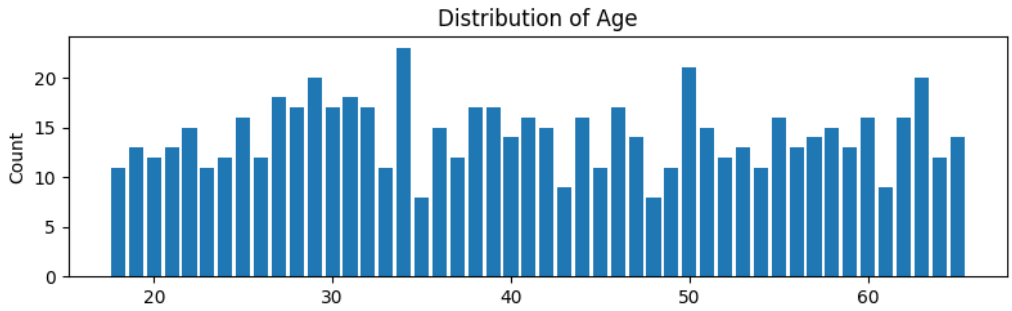


Figure 4.25. Distribution of Age

This bar chart shows distribution of employee age, primarily in the range of 20-60. Employee age is also distributed evenly, the age of about 35 is highest, and employees who are about 36,37 or 48,49 is lowest. There is no big gap between employees' age, it can prefer that the business has suitable recruitment policy, for balancing the young people who are excited, enthusiastic and the experienced personnel that is decisive and self-disciplined.

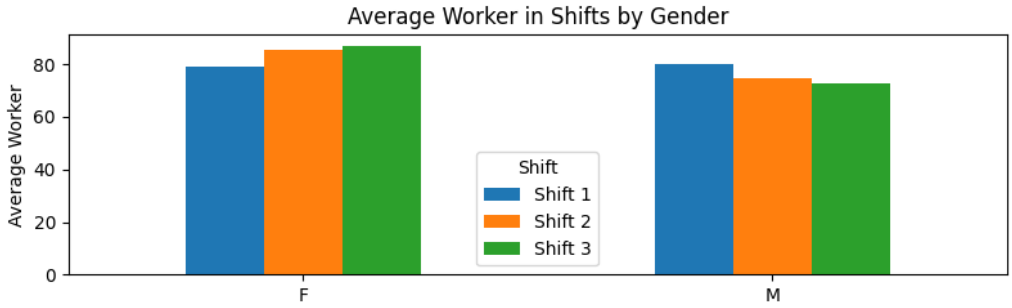


Figure 4.26. Average worker in shifts by gender

The chart can show that the number of female workers is more than the number of male workers in every shift in workday. This fact is easy to understand because the amount of female labors is much more the male. Besides, we can recognize that the employee number, including men and women, is decrease from Shift 1 to Shift 3.

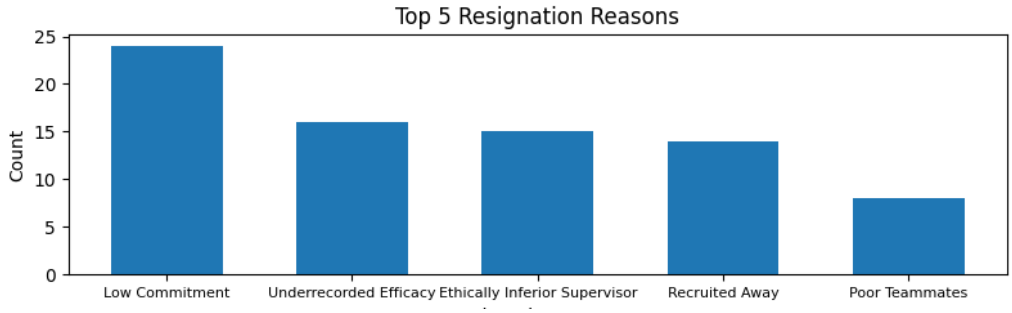


Figure 4.27. Top 5 resignation reasons

With this chart, business can understand the top 5 common reasons of the resignation.

- In the first, Low Commitment is at 24 employees
- The second reason is Under recorded Efficacy is at 16 employees.
- The third is Inferior Supervisor with 15 employees, this cause is not from employees but from biased supervisors, so business needs to re-evaluate transparency in assessment about employee efficacy.
- The two last reasons are Recruited Away (at 13 people) and Poor Teammates (at about 7 people).

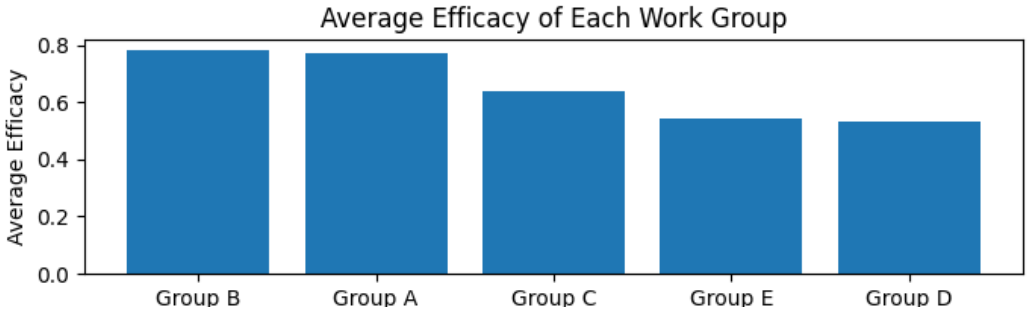


Figure 4.28. Average efficacy of each work group

In the Performance category, there are 5 work groups which are arranged from high to low by average efficacy:

- Group B is the group which efficacy is the best, with score at approximately 0.8
- The second group, Group A is less than Group B a little bit, at more than 0.7
- Group C with the 3rd highest efficacy, ranging from 0.6 to 0.7
- The 4th best efficacy is Group E, at more than 0.5
- The last group, Group D has lowest efficacy, with the number of 0.5

Therefore, business can reward some highest efficacy groups and organize activities to improve performance in low efficacy groups.

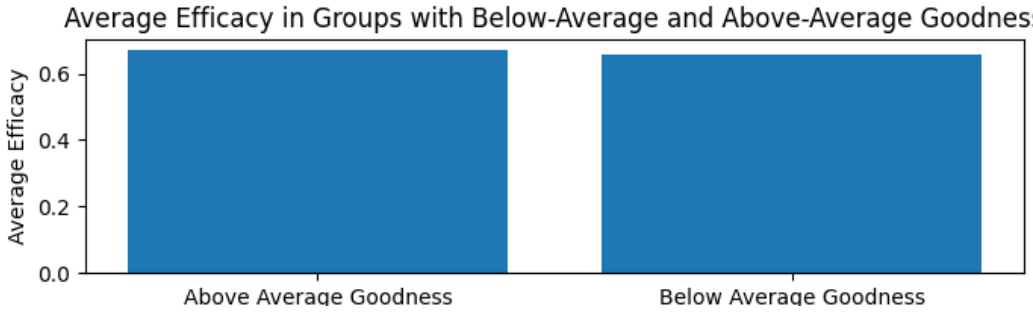


Figure 4.29. Average efficacy in groups with below-average and above-average goodness

Furthermore, this is binary chart with X-axis as Goodness and Y-axis as Efficacy, one bar shows average efficacy calculated in employees having above average goodness, one is in employees that have below average goodness. Therefore, we can see that the similarity between two bars, at more than 0.6, shows that goodness score does not impact efficacy of employees.

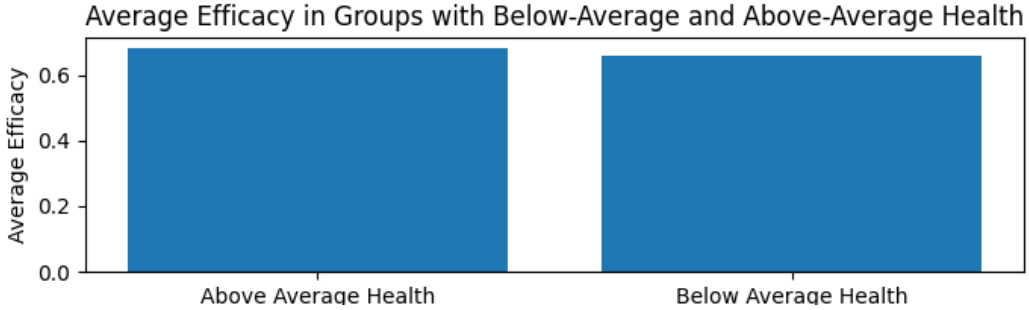


Figure 4.30. Average efficacy in groups with below-average and above-average health

Also, this chart is binary chart with X-axis as Health and Y-axis as Efficacy, one bar presented average efficacy calculated in employees having above average health, one is in employees that have below average health. Therefore, we can see that the similarity between two bars, at more than 0.6. It means that although employee health is good or bad, this problem does not affect average efficacy of workers.

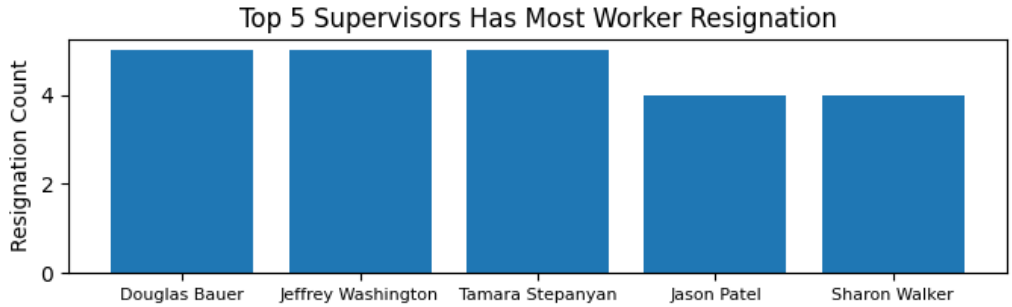


Figure 4.31. Top 5 supervisors have most worker resignation

The last plot is the top 5 supervisors who have most worker resignation. Maybe employees are dissatisfied about the supervisor attitude towards them, or their bad way of working:

- There are 3 people which have the most resignation employees, including: Douglas Bauer, Jeffrey Washington, Tamara Stepanyan, Jason Patel, Sharon Walker.
- The 4th supervisor which has most resignation employees is Jason Patel
- The last supervisor is Sharon Walker.

Business needs to find out if the cause of resignation is from Inferior Supervisors. From that, business needs to suggest for these supervisors change to better in supervising employees suitably and equally as well as encouraging employees to continue the job when intending to resign.

4.6. Machine learning page

4.6.1. Employee clustering

4.6.1.1. Business Process Model and Notation

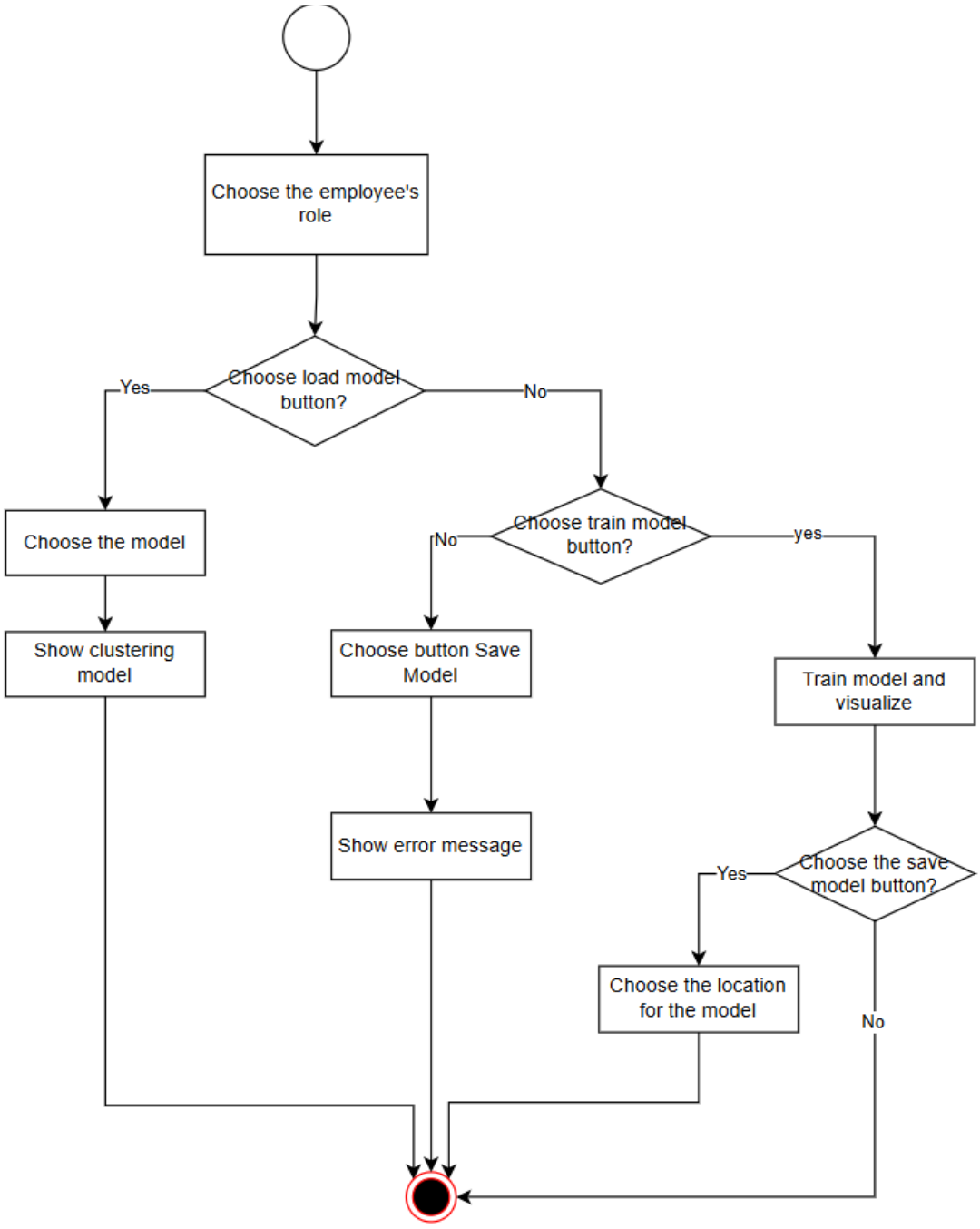


Figure 4.32. BPMN of employee clustering

After selecting the Model button on the main page, the interface will switch to the Machine learning page. On the Clustering model tab, the steps will be described as follows:

- First, users select the employee role they want to cluster.

- Next, user will have some options:
    - If the user selects the load model button, they will select an existing cluster model, then the cluster model will be displayed on the interface.
    - If the user selects the train model button, then the model will be trained and the interface will appear. After displaying the clustering results, the user can choose to save the model by selecting Save model button. If the user chooses to save the model, they choose the location where the model is stored.
    - If the user selects the Save model button, an error message will appear.

4.6.1.2. Actual on the app

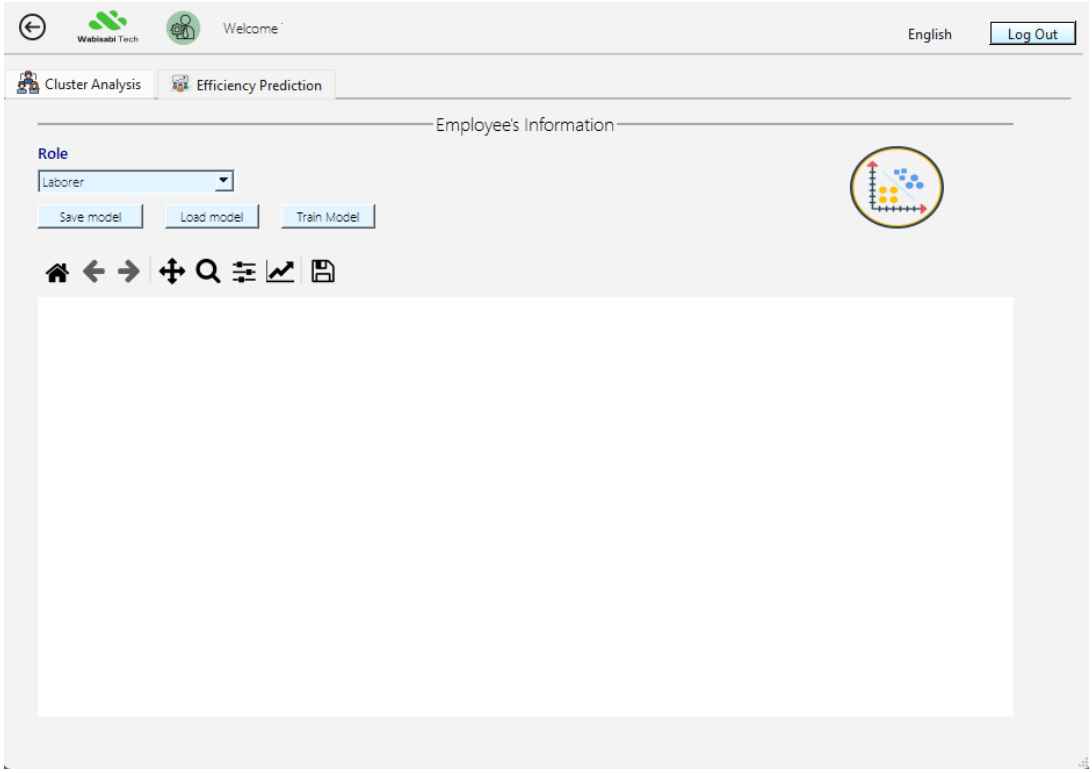


Figure 4.33. Clustering main page

This is the overall interface of the Machine Learning model page, specifically the Cluster Analysis tab. The first is the back button, if the user clicks this button, they will be returned to the main page. If the user clicks the Logout button, they will log out of their account and return to the Login page.

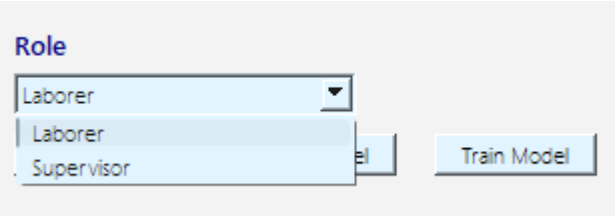


Figure 4.34: Choose the role want to see clustering

In the Cluster Analysis tab, there will be a Combo box for users to select the roles they want to cluster. There are 2 roles here: Laborer and Supervisor.

Figure 4.35: Train Laborer

The first is the train Laborer model, the user clicks on the laborer role, then press the Train model button, the model will appear on the interface.

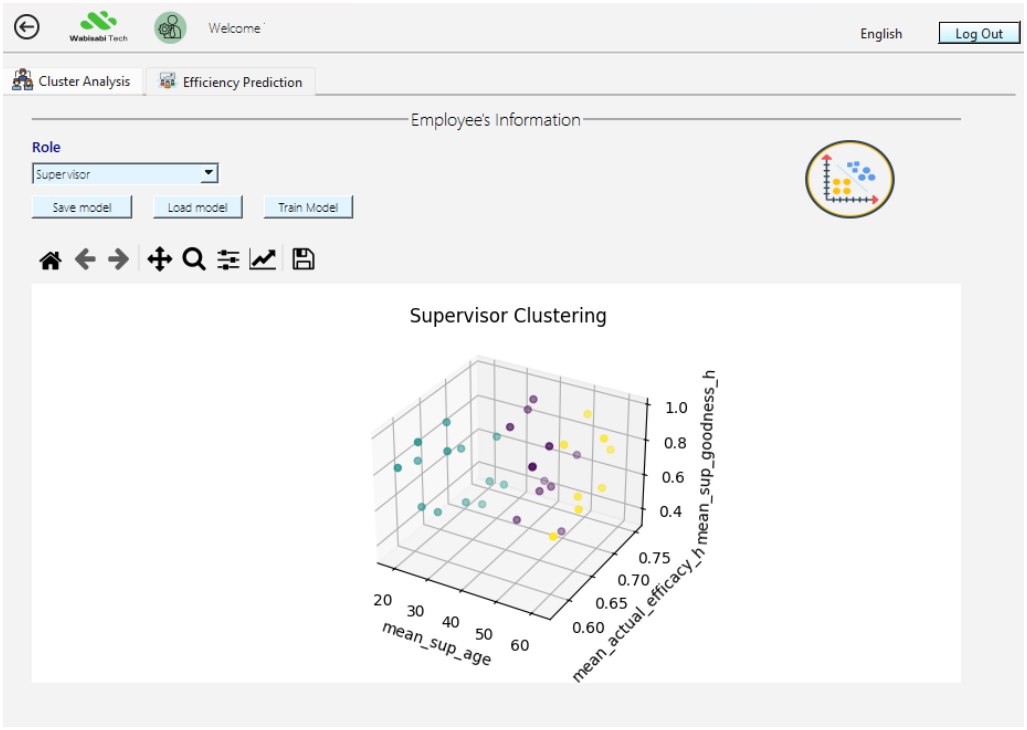


Figure 4.36: Train Supervisor

Same for train Supervisor.

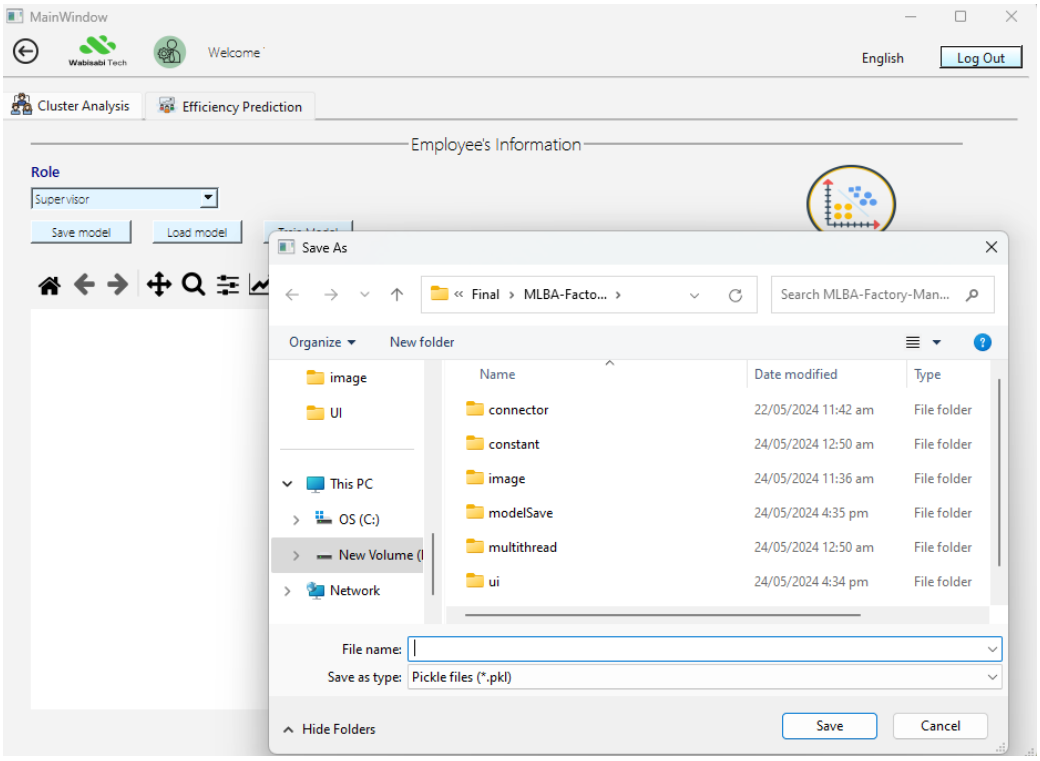


Figure 4.37: Choose the location to save cluster model

After successfully training the model, if the user clicks Save model, a pop up will appear and the user will choose a storage location for that model.

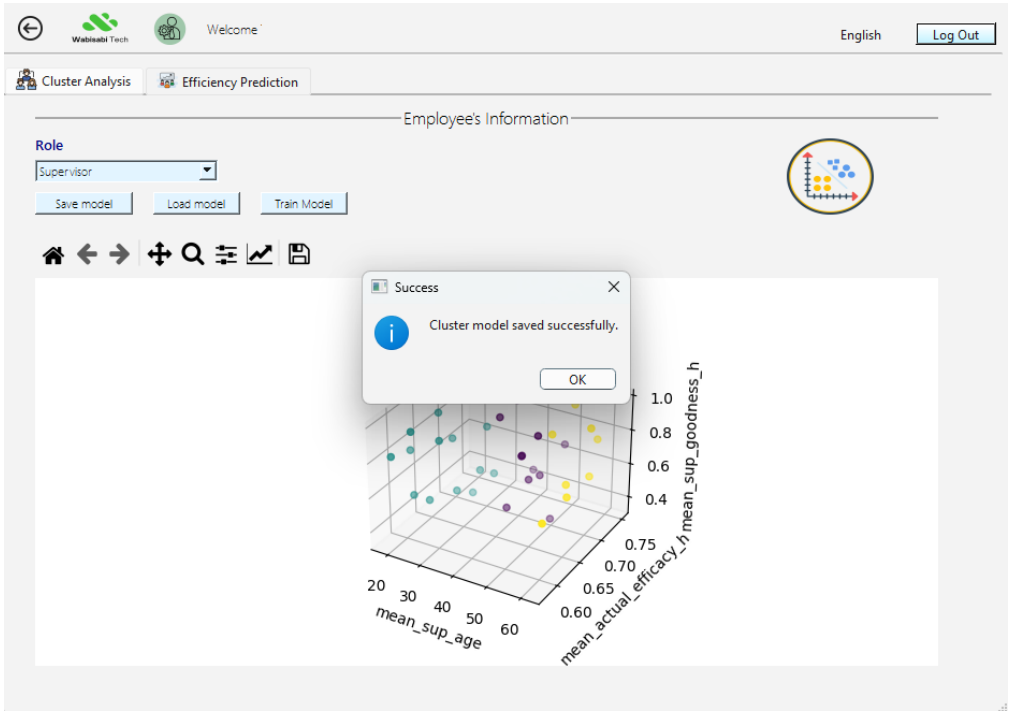


Figure 4.38: Pop up show saved cluster model

After successfully saving the model, a pop up will appear stating that the model has been saved.

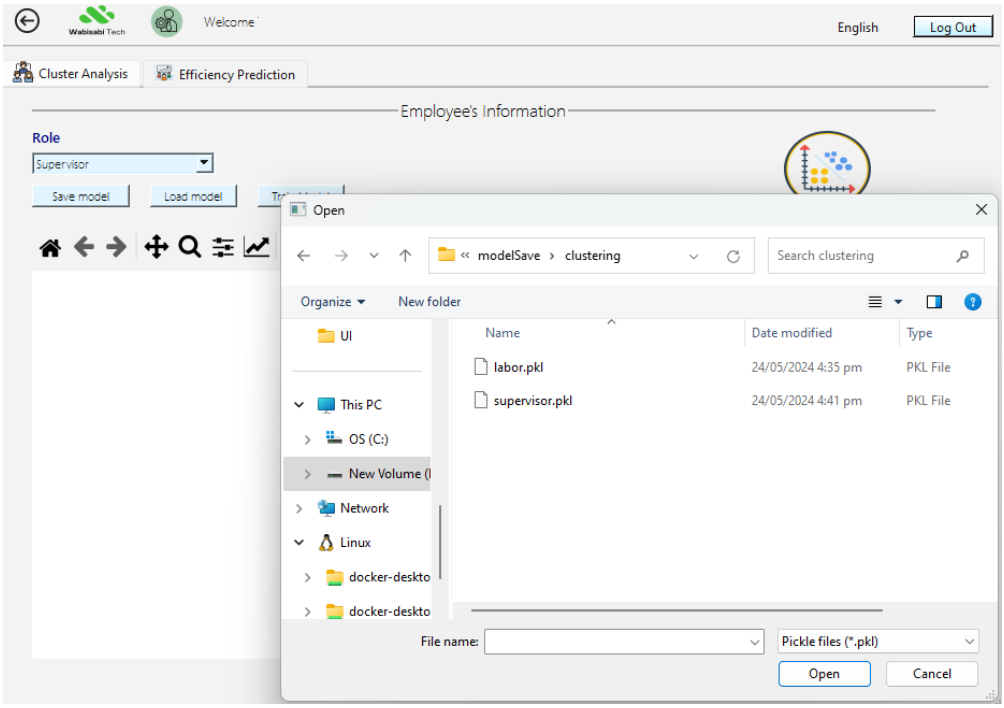


Figure 4.39: Load cluster model

When users select the Load model button, they will have to select the cluster model they want to display.

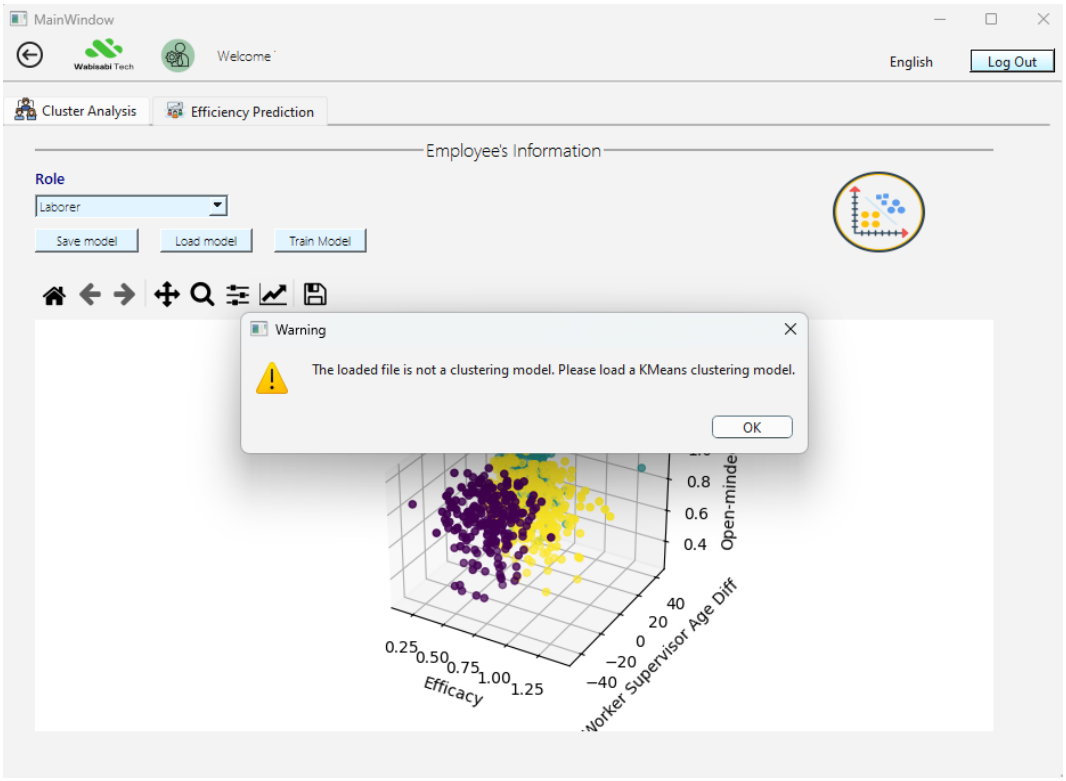


Figure 4.40: Load wrong cluster model

If the user chooses to upload a model file that is not a cluster model, an error message will appear.

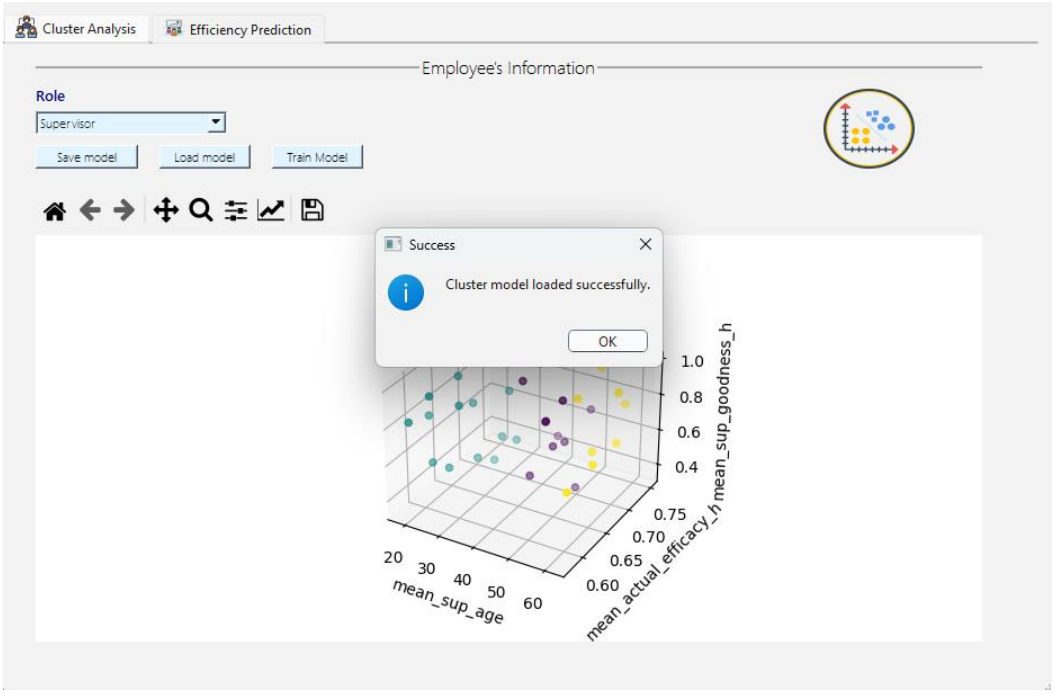


Figure 4.41: Load cluster model successful

After the user selects the correct Cluster model, a pop up will appear informing them that the model has been successfully loaded, and the model will also appear on the interface.

4.6.2. Efficacy prediction

4.6.2.1. Business Process Model and Notation

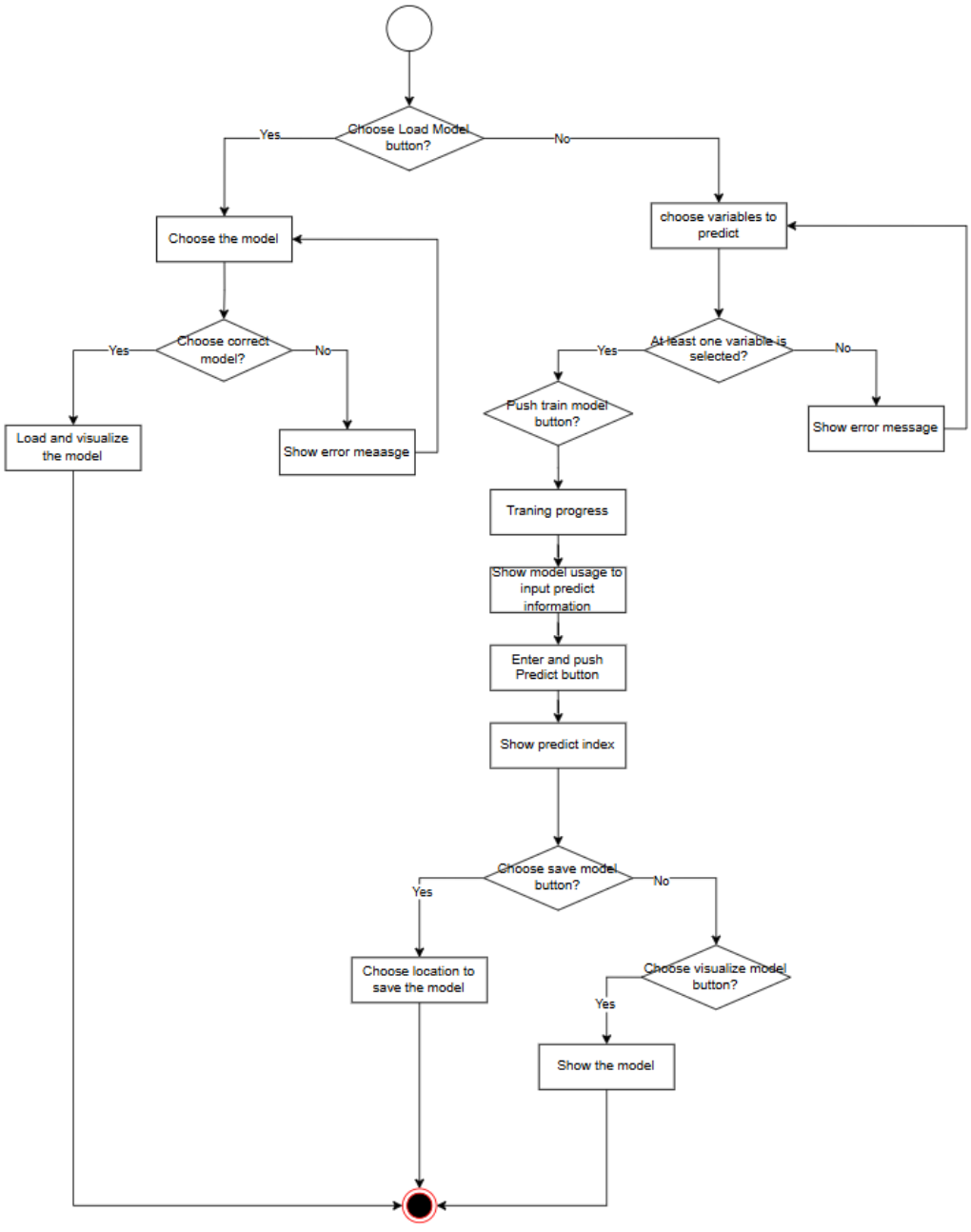


Figure 4.42: Flow of efficiency prediction

First, the user can choose to load an existing model by pressing the Load Model button.

- If the user selects this option, they will choose the model to load.

    - If the user selects the wrong model, a message will appear showing that the wrong model has been selected. Then if the user wants to continue using this feature they must select the model again.
    - If the user selects the correct model, the model will be displayed on the interface.

- If the user does not choose to load an existing model, they will select input variables for the prediction model.
    - If the user selects at least one variable, they will then press the Train model button. The interface will display a progress bar to show the model training process. After successful training, lines will appear, users will enter information to predict, then they will press the Predict button and the prediction results will be displayed.
        - After having the prediction results, if the user wants to save the model, they will press the Save model button and then choose where to store the model.
        - If they want to visualize the model, select the Visualize model button and then the prediction chart will be displayed on the interface.

4.6.2.2. Actual on the app

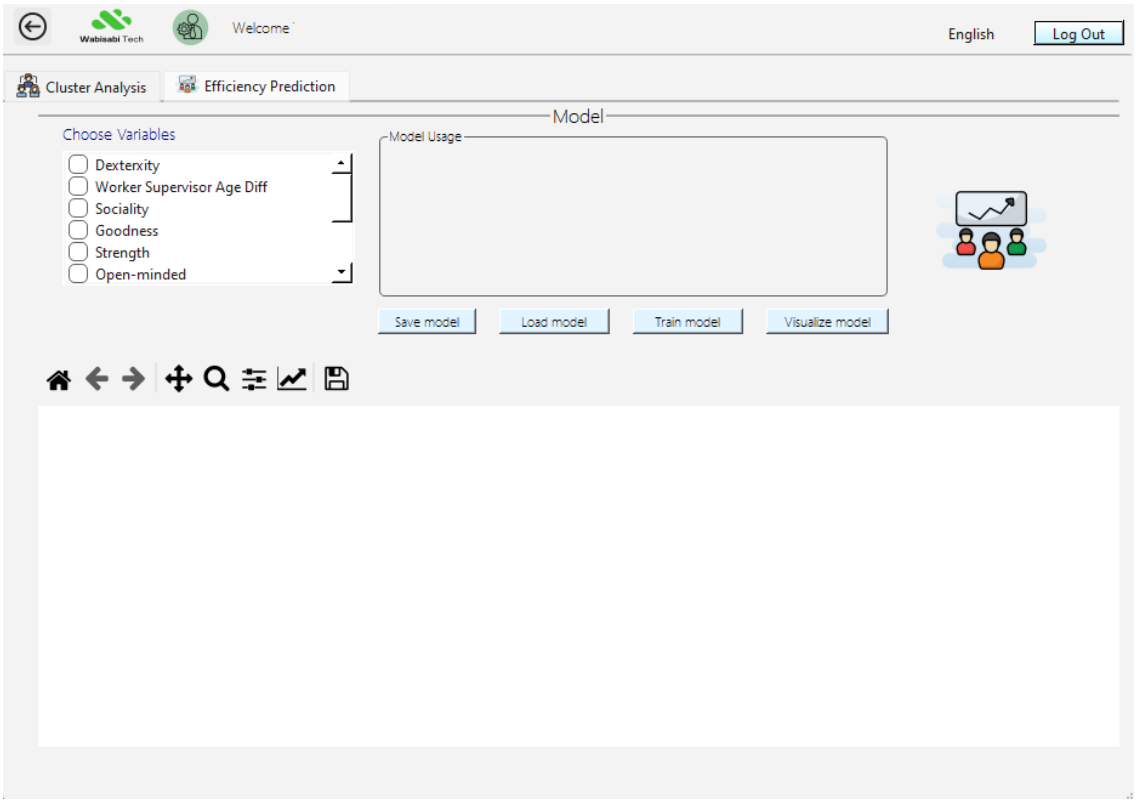


Figure 4.43: Main page of Efficiency prediction tab

This is the overview interface of Efficiency prediction. There will be a List representing the variables that the user wants to predict. Next there will be a place to show the model and predicted results, and finally a place to visualize the model.

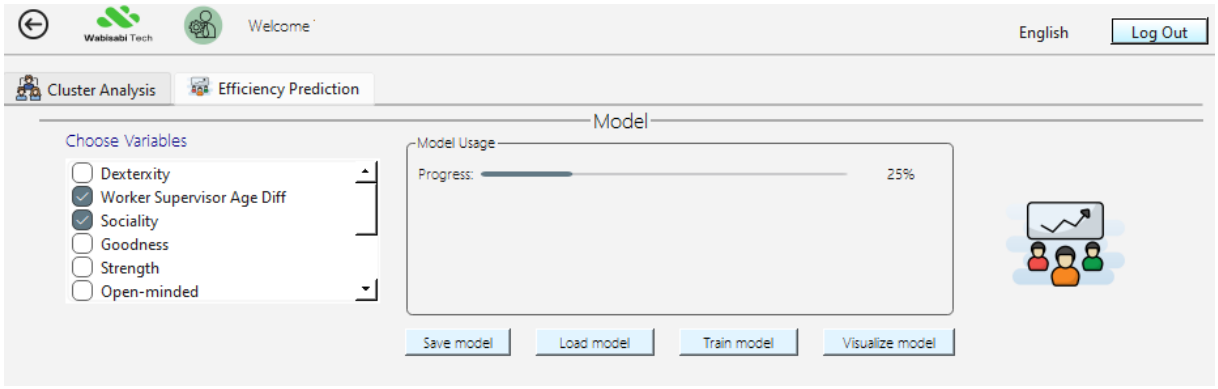


Figure 4.44: Progress bar of Training model

After the user selects the variables to predict and presses the Train model button, there will be a Progress bar to show the model training process.

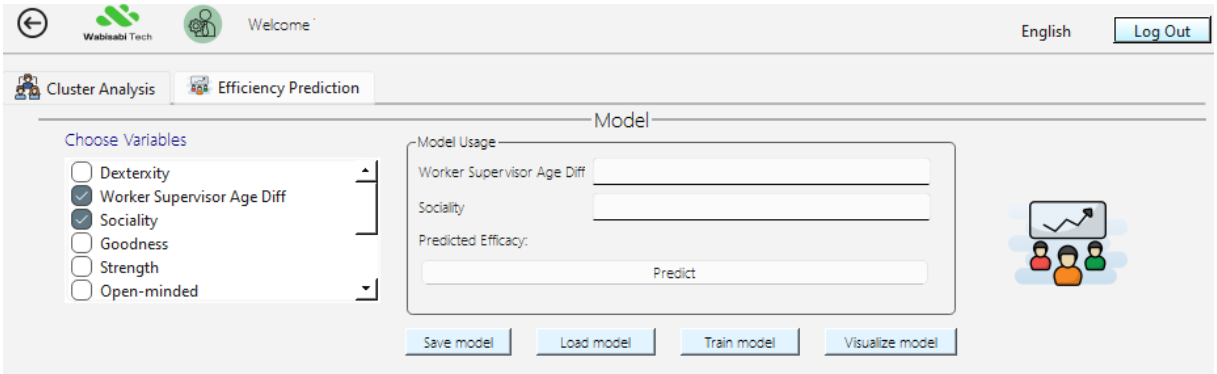


Figure 4.45: Training completed

When training is successful, the interface will display lines for users to enter information as shown above. The user will then enter the relevant indicators and press the predict button.

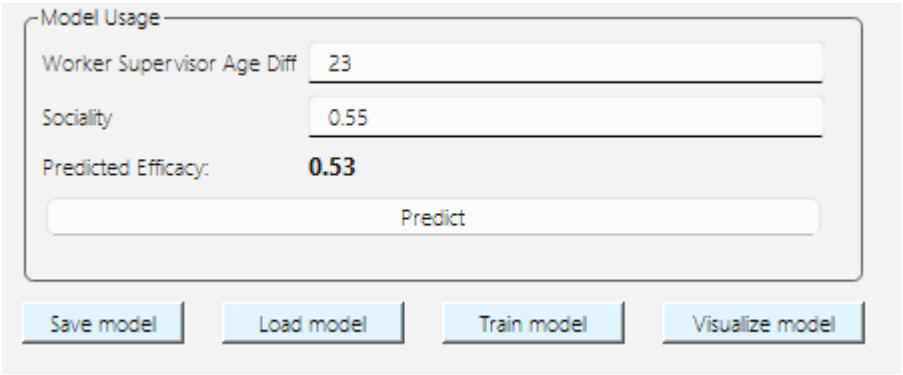


Figure 4.46: Prediction result

The prediction results will be displayed as shown above.

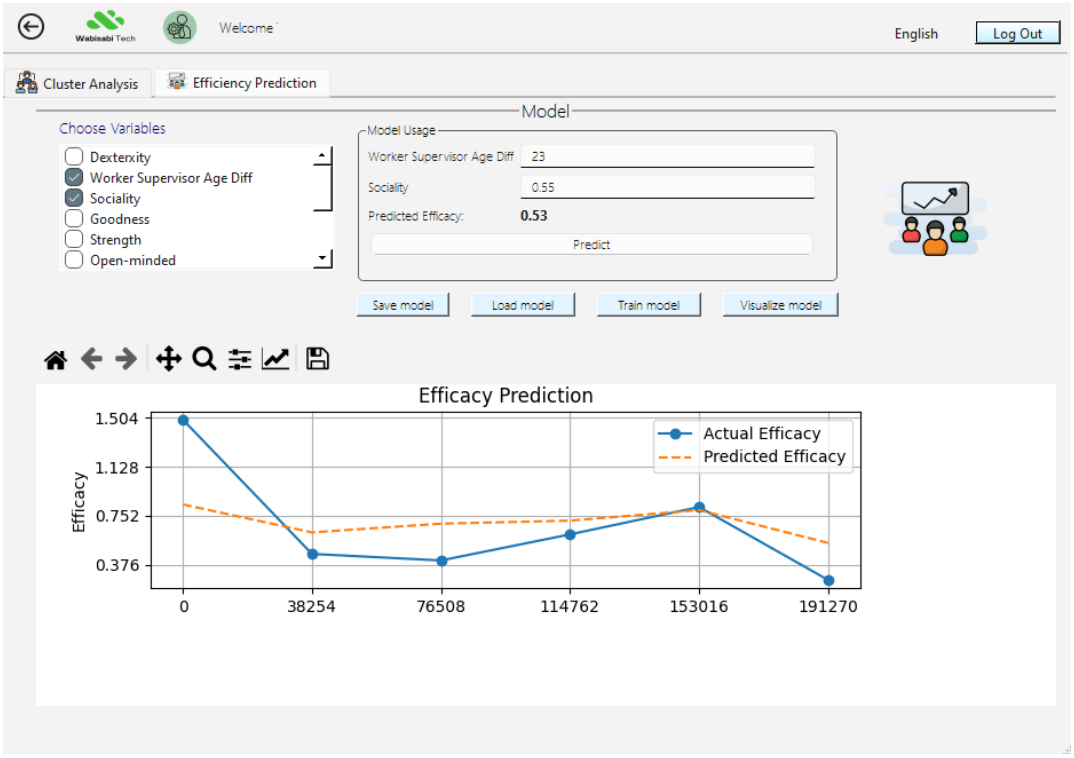


Figure 4.47: Visualize the result of Efficiency prediction

When the user presses the Visualize model button, the chart showing the predicted results will be shown as shown above.

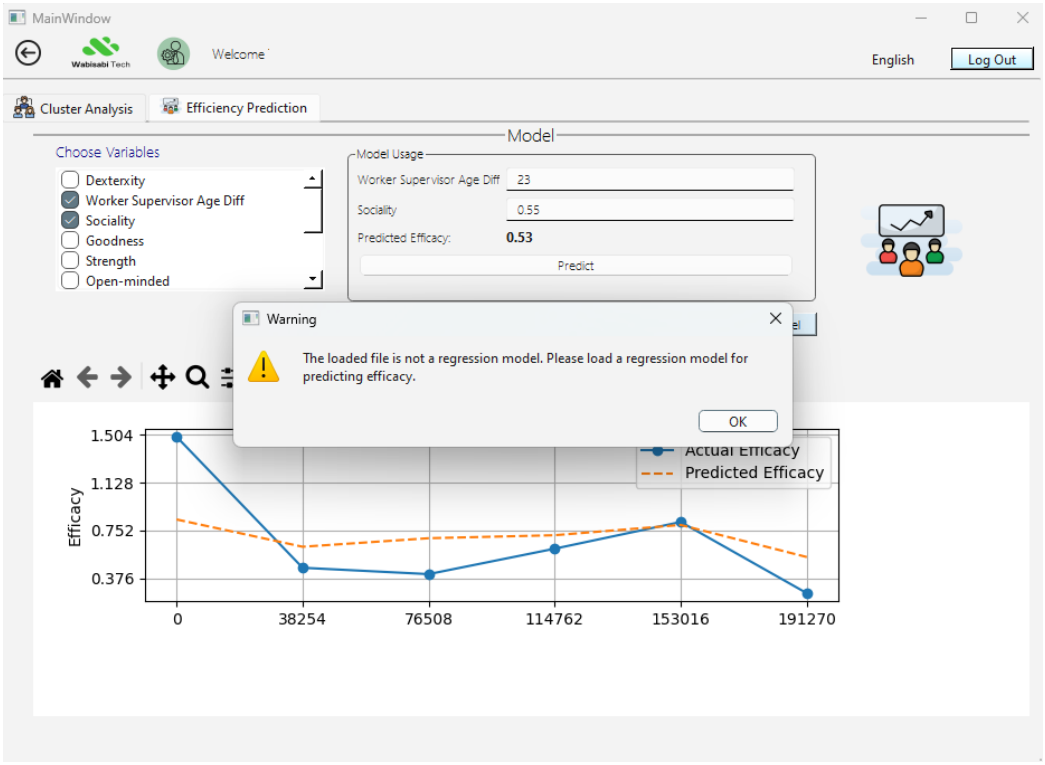


Figure 4.48: Load wrong efficiency prediction model

The interfaces for Save model and Load model are similar to the Cluster Analysis tab. The image above is the user notification interface when loading the wrong prediction model.

Chapter 5. Discussion and Future Works

5.1. Discussion

This project proposes a fully solution for factory management with all features allow user to create, manage, tracking performance, clustering worker and supervisor in the factory. Moreover, we have also developed efficacy prediction feature for manager to predict their employee performance based on health, commitment, strength…and some statistics for it. In conclusion, we believe the business can utilize this app to leverage their management and overall performance of the factory.

5.2. Limitation and Future works

5.2.1. Limitations

This research only focuses on analyzing the performance of 508 employees in one factory, the amount of data is not large enough to ensure machine learning provides an accurate model. In addition, the design team interface on Qt Designer is at a basic level, not attractive enough to users, thereby limiting the UI/UX of the app. Some statistical models are still at a basic level, unable to show the correlation between variables to find insights that need to be improved in the factory. The team has not been able to come up with a churn rate prediction model based on the indicators given in the data set.

5.2.2. Future works

Enhancing UI/UX in the app:

- Professional design tools: Utilize advanced design tools and frameworks such as Adobe XD, Figma, or Sketch to create more sophisticated and user-friendly interfaces.
- User Research: Conduct user research to understand the needs and preferences of the end-users. Use this feedback to design a more intuitive and attractive UI.
- Interactive Elements: Incorporate interactive elements such as drag-and-drop features, real-time data visualization, and customizable dashboards to enhance user engagement.

Developing statistics and prediction model:

- Correlation analysis: Use advanced statistical techniques such as Pearson correlation, Spearman rank correlation, or even partial correlation to understand the relationships between variables.
- Multivariate analysis: Implement multivariate statistical techniques like Principal Component Analysis (PCA) or Factor Analysis to identify underlying patterns and correlations between multiple variables.
- Regression models: Utilize more complex regression models, such as multiple regression, logistic regression, or hierarchical linear models to capture the relationships between dependent and independent variables more accurately.

Developing the churn rate prediction model:

- Indicator selection: After expanding data collection, carefully select indicators that are most likely to influence employee churn, such as job satisfaction, performance metrics, tenure, and demographic factors.
- Churn prediction model: Develop and validate predictive models specifically for churn prediction. Consider using classification algorithms such as Logistic Regression, Decision Trees, Random Forest, or even neural networks tailored for classification tasks.
- Evaluation metrics: Use appropriate evaluation metrics such as ROC-AUC, precision-recall curves, and F1-score to assess the performance of the churn prediction model.

References

[1] Hartigan, J. A., & Wong, M. A. (1979). *Algorithm AS 136: A K-means clustering algorithm*. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28(1), 100-108.

[2] Arthur, D., & Vassilvitskii, S. (2007). *K-means++: The advantages of careful seeding*. In Proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms (pp. 1027-1035). Society for Industrial and Applied Mathematics.

[3] Chen, T., & Guestrin, C. (2016*). XGBoost: A scalable tree boosting system*. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM.

[4] Lundberg, S. M., Erion, G. G., Chen, H., DeGrave, A. J., Prutkin, J. M., Nair, B., ... & Lee, S. I. (2018). From local explanations to global understanding with explainable AI for trees. Nature Machine Intelligence, 2(1), 56-67.

[5] Brown, I., & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. Expert Systems with Applications, 39(3), 3446-3453.

[6] Friedman, J. H. (2001). *Greedy function approximation: A gradient boosting machine.* Annals of Statistics, 29(5), 1189-1232.

[7] He, H., Zhao, J., Xie, M., & Xu, C. (2018). *Hyperparameter optimization in XGBoost using genetic algorithm.* In Proceedings of the 2018 International Conference on Computer Science and Artificial Intelligence (pp. 111-115). ACM.

[8] Zhang, J., Mucs, D., Norinder, U., & Svensson, F. (2019). *LightGBM: An effective and scalable algorithm for prediction of chemical toxicity–application to the Tox21 and mutagenicity data sets*. Journal of chemical information and modeling, 59(10), 4150-4158

[9] Ju, Y., Sun, G., Chen, Q., Zhang, M., Zhu, H., & Rehman, M. U. (2019). *A model combining convolutional neural network and LightGBM algorithm for ultra-short-term wind power forecasting*. IEEE Access, 7, 28309-28318.

[10] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). *LightGBM: A highly efficient gradient boosting decision tree. Advances in neural information processing systems*, 30.