**IMPLEMENTATION**

**MODULES:**

* Data Collection
* Dataset
* Data Preparation
* Feature Extraction
* Splitting the dataset
* Model Selection
* Analyze and Prediction
* Accuracy on test set
* Saving the Trained Model
* Prediction Module
* Model Evaluation Module

**MODULES DESCSRIPTION:**

**Data Collection:**

* In the first module of the Employee Layoff Prediction using Machine Learning, we make the data collection process. This is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get; the better our model will perform.
* There are several techniques to collect the data, like web scraping, manual interventions. The dataset is located in the model folder. The dataset is referred from the popular dataset repository called kaggle. The following is the link of the dataset:
* Kaggle Dataset Link:

https://www.kaggle.com/datasets/jayaprakashpondy/layoff

**Dataset:**

* In this module, we use the dataset which is the primary source of data for the system. This dataset contains 3612 instances and 9 attributes, with a target feature is "Total Laid Off"
* User Input: Data provided by users through the web interface, allowing for real-time malware detection based on user-uploaded files or input data.

**Data Preparation:**

* This module is responsible for preparing the LAYOFF dataset for analysis. It involves tasks such as data cleaning, normalization, and feature selection. Specifically, 9 relevant attributes are selected from the original 12 attributes to optimize the machine learning models.
* Wrangle data and prepare it for training. Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.).
* Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data.
* Visualize data to help detect relevant relationships between variables or class imbalances (bias alert!), or perform other exploratory analysis.
* NaN values are dropped from the dataset.
* Preprocess your data to handle missing values, outliers, and categorical variables. This step also involves scaling or normalizing numerical features if necessary.

**Feature Extraction:**

* If the dataset contains raw binaries or other non-numeric data, extract features that can be used by the machine learning models. This may involve static analysis (e.g., analyzing the binary's structure) or dynamic analysis (e.g., monitoring the binary's behavior during execution).
* A subset of features (permissions) is selected for model training to reduce dimensionality and focus on relevant attributes.
* Extract relevant features from your dataset. This may involve feature engineering to create new features or transform existing ones.

**Splitting the dataset:**

* Data Splitting and Validation is crucial for training and evaluating the model. This module divides the dataset into training, validation, and testing sets. It ensures that the model's performance is assessed accurately using proper validation techniques like cross-validation. Split the dataset into train and test. 80% train data and 20% test data.

**Model Selection:**

* This module handles the training of the machine learning models using the preprocessed data. It implements the Gradient Boosting Regressor and Random Forest Regressor

***Gradient Boosting Regressor:***

* Gradient Boosting Regressor is a machine learning algorithm used for regression tasks.
* It belongs to the ensemble learning methods, specifically boosting techniques and The algorithm builds an ensemble of weak learners, typically decision trees, sequentially.
* Each subsequent learner corrects the errors made by the previous ones. Gradient Boosting uses gradient descent optimization to minimize the loss function
* Common hyperparameters include the number of trees (n\_estimators), the learning rate, and tree depth.
* Gradient Boosting can handle both numerical and categorical features.
* Feature importance can be extracted from a trained Gradient Boosting model.
* It's widely used in various domains such as finance, healthcare, and marketing due to its high predictive accuracy and ability to capture complex relationships in data.
* Overall, Gradient Boosting Regressor is a versatile and powerful algorithm for regression tasks, offering high predictive accuracy and the ability to handle complex data patterns.

***Random Forest Regressor:***

* Random Forest Regressor is a machine learning algorithm used for regression tasks.
* The algorithm builds an ensemble of decision trees, where each tree is trained on a random subset of the training data and features.
* Random Forest combines the predictions of multiple decision trees to produce the final output, typically by averaging the predictions for regression tasks.
* It's robust against overfitting and noise, making it suitable for a wide range of datasets.
* Random Forest can handle both numerical and categorical features.
* It's less sensitive to hyperparameters compared to other ensemble methods like Gradient Boosting.
* Common hyperparameters include the number of trees (n\_estimators), the maximum depth of the trees, and the number of features considered for splitting at each node.
* Feature importance can be derived from a trained Random Forest model, indicating the contribution of each feature to the prediction.
* Random Forest is widely used in various domains such as finance, healthcare, and ecology due to its simplicity, scalability, and ability to provide reliable predictions with minimal hyperparameter tuning.

**Analyze and Prediction:**

* Use the trained models to make predictions on the test data. Compare the predicted values with the actual values to evaluate performance and Analyze the results to understand the factors contributing to layoffs and the predictive power of the selected models. You can also examine feature importance scores to identify the most influential factors

**Accuracy on test set:**

* Evaluate the performance of your models using metrics like Mean Absolute Error on the test set.
* Import the mean\_absolute\_error function from sklearn.metrics.
* Use the mean\_absolute\_error function to calculate the MAE for both the Gradient Boosting and Random Forest models.
* Print the calculated MAE values for both models to assess their performance on the test set.
* The Gradient Boosting Regressor achieves Training Set Mean Absolute Error: 0.2197 and Test Set Mean Absolute Error: 1.5444. The Random Forest Regressor attains a Training Set Mean Absolute Error: 0.5434 and Test Set Mean Absolute Error: 1.4992.

**Saving the Trained Model:**

* Once you’re confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into an .h5 or .pkl file using a library like pickle.
* Make sure you have pickle installed in your environment.
* Next, let’s import the module and dump the model into .pkl file.

**Prediction Module:**

* To create a module that handles real-time predictions using trained models to predict total layoffs based on new input data

**Model Evaluation Module**

* Develop a module/function to evaluate the performance of the model using metrics like Mean Absolute Erro on new data.
* This function takes a trained model and new data as input, makes predictions on the new data using the model, and then calculates the Mean Absolute Error between the actual values in the new data and the predicted values using the mean\_absolute\_error function from scikit-learn. The calculated MAE is returned as the output.