# Predicting Employee Attrition

Project Serial No. 27 of team 6

A Course Project report submitted

In partial fulfillment of requirement for the award of degree

BACHELOR IN TECHNOLOGY

IN

CSE (AI & ML)


By :

| | |
|---|---|
| Ch. Sai Vamshi | Ht. No.: 2203A52221 |
| G. Srivarsha | Ht. No.: 2203A52167 |
| V. Rahul Reddy | Ht. No.: 2203A52187 |
| M. Sujay Raj | Ht. No.: 2203A52160 |

Under the Guidance of

**Dr. Soumik Podder**

Assistant Professor, School of CS & AI.

SR UNIVERSITY

Ananthasagar (V), Hasanparthy (M), Hanumakonda Dist, T.S. – 506371


Ananthasagar (V), Hasanparthy (M), Hanumakonda Dist, T.S. – 506371

# CERTIFICATE

This is to certify that this project entitled **"Predicting Employee Attrition"** is the bonafide work carried out by Ch. Sai Vamshi, G. Srivarsha, V. Rahul Reddy & M. Sujay Raj as a Course Project for the partial fulfillment to award the degree **BACHELOR OF TECHNOLOGY** in **School of Computer Science and Artificial Intelligence** during the academic year 2023-2024 under our guidance and Supervision.

**Dr. Soumik Podder**

Assistant Professor,
School of CS & AI
SR University

**Prof. Sheshikala Martha**

Professor & Head,
School of CS & AI
SR University

**Reviewer – 1**

Name:

Designation:

Signature:

**Reviewer – 2**

Name:

Designation:

Signature:

# ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our Course project guide DR**. SOUMIK PODDER, Assistant Professor** as well as Head of the School of CS&AI, **Prof. Sheshikala Martha,** for guiding us from the beginning through the end of the Course Project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which was always constructive and encouraging and ultimately drove us to the right direction.

We express our thanks to AI & ML course coordinator, Dr. Arpita Baronia, for her encouragement and support.

Finally, we express our gratitude and sincere thanks to all the teaching and non-teaching staff of the School of Computer Science & Artificial Intelligence, for their suggestions and timely support.

# INDEX


| S.No | TITLE | Pg.No |
|---|---|---|
| 1 | Abstract | 5 |
| 2 | About the Organization | 6 |
| 3 | Introduction | 7 |
| 4 | Problem Statement | 8 |
| 5 | Methodology | 10 |
| 6 | About the Data Set | 11 |
| 7 | Exploratory Data Analysis | 12 |
| 8 | Visualization of Categorical Features | 13 |
| 9 | Splitting the Data | 15 |
| 10 | Training and Testing | 17 |
| 11 | Logistic Regression | 19 |
| 12 | Support Vector Machine (SVM) | 21 |
| 13 | K – Nearest Neighbour (KNN) | 23 |
| 14 | Perceptron | 25 |
| 15 | Random Forest | 27 |
| 16 | Results | 19 |
| 17 | Summary | 31 |
| 18 | Links of the Project | 33 |

# ABSTRACT

Employee attrition, the gradual departure of employees from a company, poses significant challenges for organizations, affecting projects and overall workforce stability. To address this, a study utilized logistic regression, a statistical method, to predict attrition, aiming for an objective, data-driven approach free from human biases.

The study's primary aim was to pinpoint key factors contributing to attrition. Three feature selection methods—information gain, select k-best, and recursive feature elimination (RFE)—were employed to identify influential variables from the data. These methods streamline data by focusing on relevant predictors, enhancing model efficiency and accuracy.

Evaluation of the predictive model was conducted using 10-fold cross-validation, a technique that splits data into subsets for training and testing. The initial model, without feature selection, achieved an accuracy of 0.865 and an AUC score of 0.932, indicating effective prediction capabilities. However, the researchers sought to refine the model's performance through feature selection.

Among the techniques tested, RFE emerged as the most promising. RFE iteratively removes less important features, refining the model to focus on critical predictors. While resulting in a slightly lower accuracy (0.853), RFE successfully identified influential factors contributing to attrition. Despite the minor decrease in accuracy, the model incorporating RFE maintained strong discriminatory power, as evidenced by an AUC score of 0.925.

In conclusion, leveraging logistic regression and feature selection methods like RFE enables organizations to predict and mitigate employee attrition risks effectively. By understanding the factors driving attrition, companies can implement proactive retention strategies, fostering a stable and productive workforce. These findings contribute to enhancing workforce management practices, ultimately supporting organizational resilience and success.

**Keywords**: employee; attrition; logistic regression; classification; prediction

# ABOUT THE ORGANISATION

SR University is a private university located in Warangal, Telangana, India. It was established in 2018 under the Telangana State Private Universities (Establishment and Regulations) Act 2018. SR University is accredited with an 'A' grade by the National Assessment and Accreditation Council (NAAC).

SR University offers a variety of undergraduate and postgraduate programs in engineering, technology, management, commerce, and arts. The university has a strong focus on industry-relevant education and offers a variety of opportunities for students to gain hands-on experience through internships, projects, and workshops. SR University also has a strong incubation center that supports students in developing and launching their startups.

SR University has a well-equipped campus with state-of-the-art facilities, including classrooms, laboratories, libraries, sports facilities, and hostels. The university also has a strong commitment to research and has published several papers in reputed journals and conferences.

SR University has a good placement record. In 2023, the university achieved 90% placements for its engineering students. The university has a strong alumni network that includes several successful entrepreneurs and professionals.

Overall, SR University is a good choice for students who are looking for an industry-relevant education and a strong focus on innovation and entrepreneurship.

# INTRODUCTION

Human resources (HR) are important for a company's success. One challenge they face is employee attrition, where employees gradually leave the company. High attrition rates mean lots of employees are leaving. There are three main reasons why employees leave: personal characteristics, company systems, and work environment. When employees leave, it costs the company time and money to hire replacements and can disrupt projects and the company's structure, affecting its performance. So, HR needs to know why employees leave. One solution is to use machine learning to predict attrition early, without human bias, so HR can take action to reduce it.

In this study, we used a method called logistic regression to predict attrition. We also used feature selection methods like information gain and select k-best to identify important factors and make training data simpler. We tested four scenarios: predicting attrition without feature selection, with information gain, with select k-best, and with recursive feature elimination (RFE). We wanted to see which method worked best by comparing accuracy, precision, recall, f1-score, and AUC score. We also used k-fold cross-validation, splitting data into parts for testing and training.

Previous research looked at employee turnover prediction using different methods like decision trees, logistic regression, and support vector machines. Some studies focused on identifying factors influencing turnover, like age and job satisfaction. Others used methods like support vector machines to predict who would leave a company, especially in the IT industry. One study compared machine learning methods for predicting churn, while another experimented with different data balancing techniques. All these studies aimed to help companies understand and prevent employee attrition.

# PROBLEM STAEMENT

The problem of employee attrition presents a significant challenge for organizations worldwide. Employee attrition, the gradual departure of employees from a company, not only disrupts workflow but also incurs substantial costs for recruitment and training of replacements. Moreover, it can adversely affect project continuity and the overall structure of the organization, leading to decreased productivity and diminished performance.

Identifying the underlying causes of employee attrition is paramount for organizations to effectively address this issue. Research indicates that attrition can be attributed to various factors, including personal characteristics, dissatisfaction with company systems, and unfavorable work environments. Understanding these factors is critical for devising targeted retention strategies and fostering a more conducive workplace environment.

Traditional approaches to addressing employee attrition often rely on subjective assessments and may overlook crucial predictive indicators. Additionally, manual analysis of vast amounts of data can be time-consuming and prone to human biases. Hence, there is a growing need for more sophisticated, data-driven methodologies to accurately predict and preempt employee attrition.

Machine learning techniques offer a promising solution to this problem by leveraging historical data to identify patterns and predict future outcomes. Logistic regression, a statistical method commonly used in predictive modeling, can be particularly effective in this context. By analyzing factors such as employee demographics, job satisfaction, and performance metrics, logistic regression models can provide insights into the likelihood of attrition for individual employees or groups.

Furthermore, feature selection methods play a crucial role in enhancing the predictive accuracy of these models. By identifying the most influential factors contributing to attrition, feature selection techniques such as information gain, select k-best, and recursive feature elimination (RFE) help streamline the data and improve model efficiency.

Despite the potential of machine learning approaches, there remains a need for comprehensive research to compare and evaluate different methodologies for predicting employee attrition. Furthermore, practical implementation of these predictive models within organizational HR systems requires consideration of real-world constraints and challenges.

In conclusion, the problem of employee attrition underscores the importance of adopting data-driven approaches to workforce management. By leveraging machine learning techniques and feature selection methods, organizations can gain valuable insights into attrition patterns, enabling them to develop proactive retention strategies and foster a more stable and productive workforce.

# METHODOLOGY

Our research method can be broken down into several steps. We began by utilizing the IBM HR employee attrition dataset to gather insights into employee turnover. To understand the dataset and factors associated with employee attrition, we conducted Employee Exploratory Data Analysis (EEDA). This initial analysis helped us identify patterns and relationships within the data.

Following EEDA, we employed feature engineering techniques to refine the dataset and identify the most relevant parameters for building our predictive model. This process involved encoding features to prepare them for analysis and prediction.

During our analysis, we observed that the dataset was imbalanced, with varying numbers of employees in different categories. To address this imbalance, we applied the SMOTE data resampling technique. This method helped create a balanced dataset, ensuring fair representation of all categories and improving the reliability of our model.

With a balanced dataset in hand, we proceeded to split it into two parts: 85% for training the model and 15% for testing its performance. This division allowed us to assess how well the model generalized to unseen data.

Before training the model, we carefully tuned its parameters to optimize its performance. This step involved adjusting various settings to ensure the model's accuracy and effectiveness in predicting employee attrition.

Once the model was trained and tested, we had a refined and generalized version ready for practical use. By inputting the details of employees into the model, we could predict whether they were likely to leave the company or not.

In summary, our research involved data exploration, feature engineering, dataset balancing, model training, parameter tuning, and predictive modeling. Through these steps, we aimed to gain valuable insights into the factors driving employee attrition and develop a reliable tool to assist organizations in managing their workforce effectively.

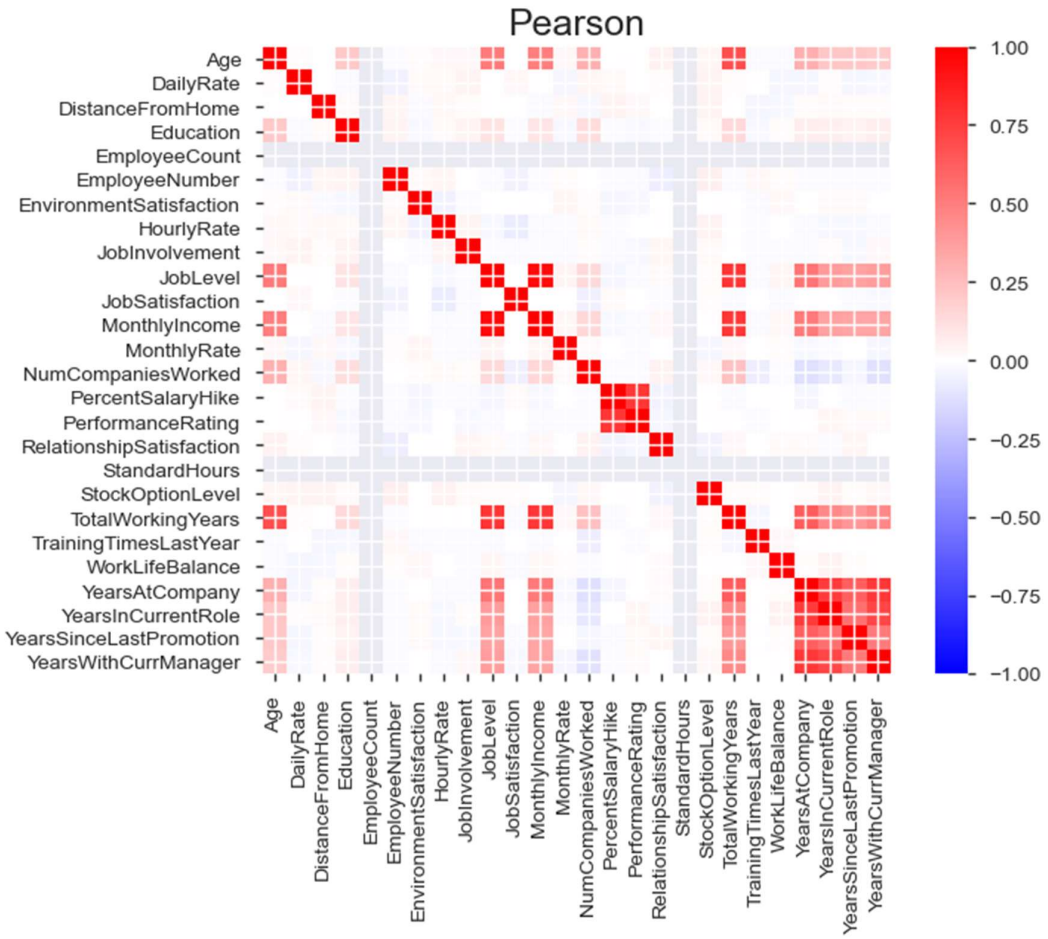
# ABOUT THE DATA SET

We utilized the IBM HR Employee Attrition dataset for our data analysis and machine learning model development to predict employee turnover. Created by IBM data scientists, this dataset comprises 35 features that provide insights into various aspects of employee behavior and characteristics within the workplace. Our goal was to investigate the factors contributing to employee attrition by analyzing this dataset. By examining these factors, we aimed to develop a generalized machine learning model capable of predicting employee turnover, particularly among valuable employees. Through this analysis, we sought to provide organizations with valuable insights into potential attrition risks and enable proactive measures to retain their workforce effectively.

| Column | Non-Null Count | Data Type | Column | Non-Null Count | Data Type |
|---|---|---|---|---|---|
| Age | 1470 | int64 | MonthlyIncome | 1470 | int64 |
| Attrition | 1470 | object | MonthlyRate | 1470 | int64 |
| BusinessTravel | 1470 | object | NumCompaniesWorked | 1470 | int64 |
| DailyRate | 1470 | int64 | Over18 | 1470 | object |
| Department | 1470 | object | OverTime | 1470 | object |
| DistanceFromHome | 1470 | int64 | PercentSalaryHike | 1470 | int64 |
| Education | 1470 | int64 | PerformanceRating | 1470 | int64 |
| EducationField | 1470 | object | RelationshipSatisfaction | 1470 | int64 |
| EmployeeCount | 1470 | int64 | StandardHours | 1470 | int64 |
| EmployeeNumber | 1470 | int64 | StockOptionLevel | 1470 | int64 |
| EnvironmentSatisfaction | 1470 | int64 | TotalWorkingYears | 1470 | int64 |
| Gender | 1470 | object | TrainingTimesLastYear | 1470 | int64 |
| HourlyRate | 1470 | int64 | WorkLifeBalance | 1470 | int64 |
| JobInvolvement | 1470 | int64 | YearsAtCompany | 1470 | int64 |
| JobLevel | 1470 | int64 | YearsInCurrentRole | 1470 | int64 |
| JobRole | 1470 | object | YearsSinceLastPromotion | 1470 | int64 |
| JobSatisfaction | 1470 | int64 | YearsWithCurrManager | 1470 | int64 |
| MaritalStatus | 1470 | object | | | |

Above table showing Samples of Dataset

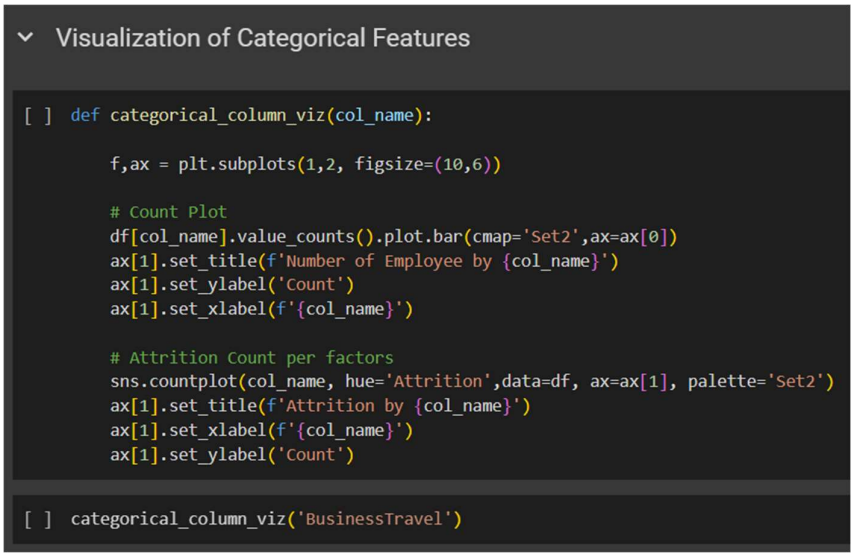
# EXPLORATORY DATA ANALYSIS




The Above Picture Showing Corelations from the taken Dataset

- Job level is strongly correlated with total working hours
- Monthly income is strongly correlated with Job level
- Monthly income is strongly correlated with total working hours
- Age is strongly correlated with monthly income

# VISUALIZATION OF CATEGORICAL FEATURES



Visualization of Categorical Features

```
[ ] def categorical_column_viz(col_name):

        f,ax = plt.subplots(1,2, figsize=(10,6))

        # Count Plot
        df[col_name].value_counts().plot.bar(cmap='Set2',ax=ax[0])
        ax[1].set_title(f'Number of Employee by {col_name}')
        ax[1].set_ylabel('Count')
        ax[1].set_xlabel(f'{col_name}')

        # Attrition Count per factors
        sns.countplot(col_name, hue='Attrition',data=df, ax=ax[1], palette='Set2')
        ax[1].set_title(f'Attrition by {col_name}')
        ax[1].set_xlabel(f'{col_name}')
        ax[1].set_ylabel('Count')
```

```
[ ] categorical_column_viz('BusinessTravel')
```

This Python function, `categorical_column_viz`, is designed to visualize categorical columns in a dataset, specifically for exploring factors related to employee attrition.

1. `def categorical_column_viz(col_name):`: This line defines a function named `categorical_column_viz` that takes a single argument `col_name`, which represents the name of the categorical column to be visualized.
2. `f,ax = plt.subplots(1,2, figsize=(10,6))`: This line initializes a matplotlib figure (`f`) with two axes (`ax`) arranged in a single row and two columns. It sets the figure size to 10 inches in width and 6 inches in height.
3. `df[col_name].value_counts().plot.bar(cmap='Set2',ax=ax[0])`: This line creates a bar plot on the first axis (`ax[0]`) showing the count of each category in the specified column (`col_name`) of the DataFrame (`df`). It uses the 'Set2' colormap for coloring.
4. `ax[1].set_title(f'Number of Employee by {col_name}')`: This line sets the title of the second axis (`ax[1]`) to indicate that it displays the number of employees by the specified column.
5. `ax[1].set_ylabel('Count')`: This line sets the y-axis label of the second axis to 'Count'.
6. `ax[1].set_xlabel(f'{col_name}')`: This line sets the x-axis label of the second axis to the name of the specified column (`col_name`).

7. `sns.countplot(col_name, hue='Attrition',data=df, ax=ax[1], palette='Set2')`: This line creates a count plot on the second axis (`ax[1]`) to visualize the distribution of attrition (`Attrition`) across different categories of the specified column (`col_name`). It uses the seaborn library's `countplot` function, passing the DataFrame (`df`), the column name (`col_name`), and the `Attrition` column as arguments. The `hue` parameter is set to 'Attrition' to differentiate between attrition and non-attrition cases, and the 'Set2' palette is used for coloring.
8. `ax[1].set_title(f'Attrition by {col_name}')`: This line sets the title of the second axis to indicate that it displays attrition counts by the specified column.
9. `ax[1].set_xlabel(f'{col_name}')`: This line sets the x-axis label of the second axis to the name of the specified column (`col_name`).
10. `ax[1].set_ylabel('Count')`: This line sets the y-axis label of the second axis to 'Count'.

In summary, this function creates a visual representation of categorical columns in a dataset, showing the distribution of categories and the relationship between categories and employee attrition. It uses matplotlib and seaborn libraries for plotting and visualization.

# SPLITTING THE DATA


This code snippet is related to splitting a dataset into training and testing sets for machine learning purposes, followed by displaying the first few rows of the training data.

1. `X_train,X_test, y_train, y_test = train_test_split(X_all,y, test_size=0.30)`: This line uses the `train_test_split` function from the `sklearn.model_selection` module to split the dataset into training and testing sets. It assigns four variables:

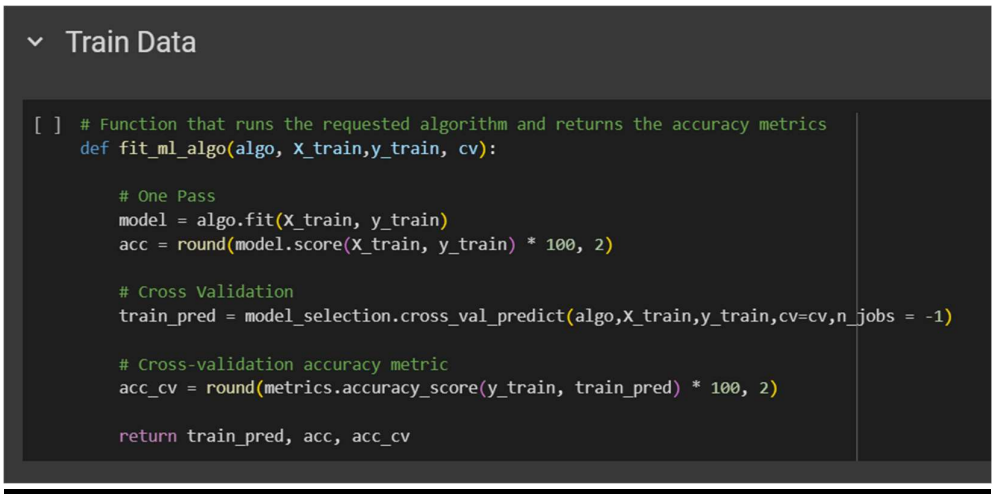
    - `X_train`: The training data features (independent variables).
    - `X_test`: The testing data features.
    - `y_train`: The target variable (dependent variable) corresponding to the training data.
    - `y_test`: The target variable corresponding to the testing data.
    - `X_all`: Represents the features of the entire dataset.
    - `y`: Represents the target variable (labels) of the entire dataset.
    - `test_size=0.30`: Specifies that 30% of the dataset will be allocated for testing, while the remaining 70% will be used for training.

2. `print(f"Train data shape: {X_train.shape}, Test Data Shape {X_test.shape}")`: This line prints the shapes (dimensions) of the training and testing datasets. It uses formatted string literals (f-strings) to include the shapes of `X_train` and `X_test` in the output.

3. `X_train.head()`: This line displays the first few rows of the training dataset (`X_train`). The `.head()` method is used to retrieve the top rows of the DataFrame. This is helpful for quickly inspecting the structure and content of the training data.

In summary, this code splits the dataset into training and testing sets, prints the shapes of the resulting datasets, and displays the first few rows of the training data to provide an initial glimpse into its structure. This is a common procedure in machine learning to ensure that models are trained and evaluated on separate, independent datasets.

# TRAINING AND TESTING




This Python function, `fit_ml_algo`, is designed to train a machine learning algorithm on a given dataset and return accuracy metrics both for the training data and through cross-validation.

1. `def fit_ml_algo(algo, X_train,y_train, cv):`: This line defines a function named `fit_ml_algo` that takes four arguments:
   - `algo`: Represents the machine learning algorithm to be trained.
   - `X_train`: The features (independent variables) of the training dataset.
   - `y_train`: The target variable (dependent variable) of the training dataset.
   - `cv`: Represents the number of folds for cross-validation.

2. `model = algo.fit(X_train, y_train)`: This line trains the specified machine learning algorithm (`algo`) on the training data (`X_train`, `y_train`) using the `fit` method. It returns a trained model.

3. `acc = round(model.score(X_train, y_train) * 100, 2)`: This line calculates the accuracy of the trained model on the training data. It uses the `score` method of the trained model to compute the accuracy, which is then rounded to two decimal places.

4. `train_pred = model_selection.cross_val_predict(algo,X_train,y_train,cv=cv,n_jobs = -1)`: This line performs cross-validation on the training data using the specified machine learning algorithm (`algo`). It uses the `cross_val_predict` function from the `model_selection` module, which generates cross-validated predictions for each sample in the training data. The `cv` parameter specifies the number of folds for cross-validation, and `n_jobs = -1` indicates that the computation will be parallelized across all available CPU cores.

5. `acc_cv = round(metrics.accuracy_score(y_train, train_pred) * 100, 2)`: This line calculates the accuracy of the cross-validated predictions (`train_pred`) compared to the actual target values (`y_train`). It uses the `accuracy_score` function from the `metrics` module to compute the accuracy, which is then rounded to two decimal places.

6. `return train_pred, acc, acc_cv`: This line returns three values:
    - `train_pred`: The cross-validated predictions generated during cross-validation.
    - `acc`: The accuracy of the trained model on the training data.
    - `acc_cv`: The accuracy of the cross-validated predictions.

In summary, this function trains a machine learning algorithm on a given dataset, calculates the accuracy of the trained model on the training data, and evaluates the model's performance through cross-validation, returning both training and cross-validation accuracy metrics.

# LOGISTIC REGRESSION


Logistic Regression is a statistical method used for binary classification tasks, where the goal is to predict the probability of a binary outcome (e.g., yes/no, true/false). It's commonly employed in machine learning for scenarios where the dependent variable is categorical.

1. Logistic Regression:

- Logistic Regression is instantiated with `LogisticRegression()`, creating an instance of the Logistic Regression model. This model will learn to predict the probability of a binary outcome based on the input features.

2. Training the Model:

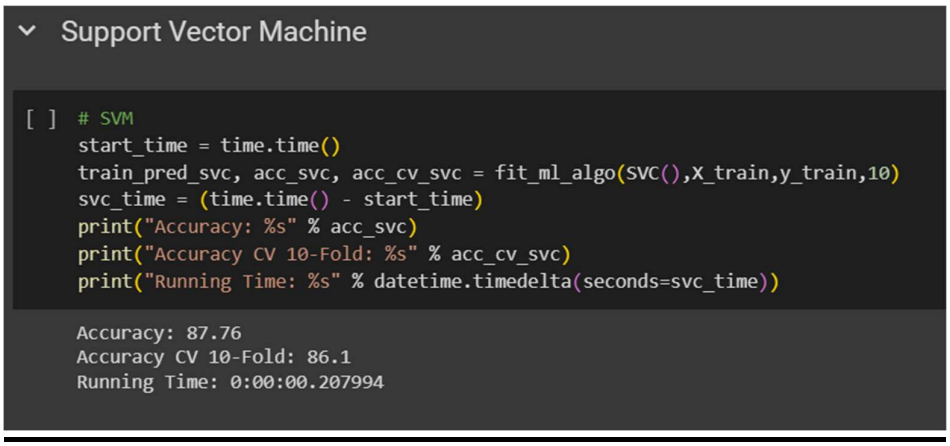
- `fit_ml_algo(LogisticRegression(), X_train,y_train, 10)`: This line calls the `fit_ml_algo` function defined earlier, passing the Logistic Regression model (`LogisticRegression()`), training features (`X_train`), training labels (`y_train`), and the number of folds for cross-validation (10).
- The function trains the Logistic Regression model on the training data (`X_train`, `y_train`) and returns the cross-validated predictions (`train_pred_log`), accuracy on the training data (`acc_log`), and accuracy through 10-fold cross-validation (`acc_cv_log`).

3. Performance Evaluation:

- `print("Accuracy: %s" % acc_log)`: This line prints the accuracy of the Logistic Regression model on the training data.
- `print("Accuracy CV 10-Fold: %s" % acc_cv_log)`: This line prints the accuracy of the Logistic Regression model through 10-fold cross-validation.
- `log_time = (time.time() - start_time)`: This line calculates the time taken to train the Logistic Regression model.
- `print("Running Time: %s" % datetime.timedelta(seconds=log_time))`: This line prints the running time of the Logistic Regression model training process in a human-readable format using the `timedelta` function from the `datetime` module.

In summary, the code applies Logistic Regression to a dataset, evaluates its accuracy on the training data and through cross-validation, and reports the running time of the training process. This allows for an assessment of the model's performance and efficiency in predicting binary outcomes.

# SUPPORT VECTOR MACHINE (SVM)




Support Vector Machine (SVM) is a powerful supervised learning algorithm used for both classification and regression tasks. It works by finding the hyperplane that best separates different classes in the feature space. SVM is effective in high-dimensional spaces and is particularly useful when the number of dimensions exceeds the number of samples.

1. SVM Model Initialization:

   - The code initializes an SVM model by calling `SVC()`, which creates an instance of the Support Vector Classification model.

2. Training the Model:

   - `fit_ml_algo(SVC(),X_train,y_train,10)`: This line calls the `fit_ml_algo` function, passing the initialized SVM model (`SVC()`), training features (`X_train`), training labels (`y_train`), and the number of folds for cross-validation (10).
   - Inside the function, the SVM model is trained on the training data (`X_train`, `y_train`), and cross-validated predictions (`train_pred_svc`), accuracy on the training data (`acc_svc`), and accuracy through 10-fold cross-validation (`acc_cv_svc`) are computed.

3. Performance Evaluation:

- `print("Accuracy: %s" % acc_svc)`: This line prints the accuracy of the SVM model on the training data.
- `print("Accuracy CV 10-Fold: %s" % acc_cv_svc)`: This line prints the accuracy of the SVM model through 10-fold cross-validation.
- `svc_time = (time.time() - start_time)`: This line calculates the time taken to train the SVM model.
- `print("Running Time: %s" % datetime.timedelta(seconds=svc_time))`: This line prints the running time of the SVM model training process in a human-readable format using the `timedelta` function from the `datetime` module.

In summary, the code trains an SVM model on a dataset, evaluates its accuracy on the training data and through cross-validation, and reports the running time of the training process. This allows for an assessment of the model's performance and efficiency in classifying data points into different categories.

# K – NEAREST NEIGHBOUR (KNN)


K Nearest Neighbors (KNN) is a simple yet effective supervised learning algorithm used for both classification and regression tasks. In KNN, the classification of a data point is determined by the class most commonly represented among its k nearest neighbors in the feature space. It's a non-parametric and instance-based algorithm, meaning it doesn't make assumptions about the underlying data distribution and stores the entire training dataset for prediction.

1. KNN Model Initialization:

- The code initializes a KNN model by calling `KNeighborsClassifier(n_neighbors = 3)`, which creates an instance of the K Neighbors Classifier model with `n_neighbors` set to 3. This means that the classification of each data point will be based on the class labels of its 3 nearest neighbors.

2. Training the Model:

- `fit_ml_algo(KNeighborsClassifier(n_neighbors = 3),X_train,y_train,10)`: This line calls the `fit_ml_algo` function, passing the initialized KNN model (`KNeighborsClassifier(n_neighbors = 3)`), training features (`X_train`), training labels (`y_train`), and the number of folds for cross-validation (10).
- Inside the function, the KNN model is trained on the training data (`X_train`, `y_train`), and cross-validated predictions (`train_pred_knn`), accuracy on the training data (`acc_knn`), and accuracy through 10-fold cross-validation (`acc_cv_knn`) are computed.

3. Performance Evaluation:

- `print("Accuracy: %s" % acc_knn)`: This line prints the accuracy of the KNN model on the training data.
- `print("Accuracy CV 10-Fold: %s" % acc_cv_knn)`: This line prints the accuracy of the KNN model through 10-fold cross-validation.
- `knn_time = (time.time() - start_time)`: This line calculates the time taken to train the KNN model.
- `print("Running Time: %s" % datetime.timedelta(seconds=knn_time))`: This line prints the running time of the KNN model training process in a human-readable format using the `timedelta` function from the `datetime` module.

In summary, the code trains a KNN model on a dataset, evaluates its accuracy on the training data and through cross-validation, and reports the running time of the training process. This allows for an assessment of the model's performance and efficiency in classifying data points based on their nearest neighbors.

# PERCEPTRON


The Perceptron is one of the simplest types of artificial neural networks used for binary classification tasks. It's a single-layer neural network with a single output neuron that applies a linear activation function. The Perceptron algorithm learns by adjusting its weights based on misclassifications to minimize errors and improve accuracy.

1. Perceptron Model Initialization:

- The code initializes a Perceptron model by calling `Perceptron()`, which creates an instance of the Perceptron model.

2. Training the Model:

- `fit_ml_algo(Perceptron(),X_train,y_train,10)`: This line calls the `fit_ml_algo` function, passing the initialized Perceptron model (`Perceptron()`), training features (`X_train`), training labels (`y_train`), and the number of folds for cross-validation (10).
- Inside the function, the Perceptron model is trained on the training data (`X_train`, `y_train`), and cross-validated predictions (`train_pred_gaussian`), accuracy on the training data (`acc_perceptron`), and accuracy through 10-fold cross-validation (`acc_cv_perceptron`) are computed.

3. Performance Evaluation:

- `print("Accuracy: %s" % acc_perceptron)`: This line prints the accuracy of the Perceptron model on the training data.
- `print("Accuracy CV 10-Fold: %s" % acc_cv_perceptron)`: This line prints the accuracy of the Perceptron model through 10-fold cross-validation.
- `perceptron_time = (time.time() - start_time)`: This line calculates the time taken to train the Perceptron model.
- `print("Running Time: %s" % datetime.timedelta(seconds=perceptron_time))`: This line prints the running time of the Perceptron model training process in a human-readable format using the `timedelta` function from the `datetime` module.

In summary, the code trains a Perceptron model on a dataset, evaluates its accuracy on the training data and through cross-validation, and reports the running time of the training process. This allows for an assessment of the model's performance and efficiency in binary classification tasks.

# RANDOM FOREST


Random Forest is an ensemble learning method used for both classification and regression tasks. It operates by constructing multiple decision trees during training and outputting the class that is the mode of the classes (classification) or the mean prediction (regression) of the individual trees. Random Forests are known for their robustness and ability to handle high-dimensional data with ease.

1. Random Forest Model Initialization:

- The code initializes a Random Forest model by calling `RandomForestClassifier(n_estimators=100)`. This creates an instance of the Random Forest Classifier model with 100 decision trees (`n_estimators=100`).

2. Training the Model:

- `fit_ml_algo(RandomForestClassifier(n_estimators=100),X_train, y_train,10)`: This line calls the `fit_ml_algo` function, passing the initialized Random Forest model (`RandomForestClassifier(n_estimators=100)`), training features (`X_train`), training labels (`y_train`), and the number of folds for cross-validation (10).
- Inside the function, the Random Forest model is trained on the training data (`X_train`, `y_train`), and cross-validated predictions (`train_pred_dt`), accuracy on the training data (`acc_rf`), and accuracy through 10-fold cross-validation (`acc_cv_rf`) are computed.

3. Performance Evaluation:

- `print("Accuracy: %s" % acc_rf)`: This line prints the accuracy of the Random Forest model on the training data.
- `print("Accuracy CV 10-Fold: %s" % acc_cv_rf)`: This line prints the accuracy of the Random Forest model through 10-fold cross-validation.
- `rf_time = (time.time() - start_time)`: This line calculates the time taken to train the Random Forest model.
- `print("Running Time: %s" % datetime.timedelta(seconds=rf_time))`: This line prints the running time of the Random Forest model training process in a human-readable format using the `timedelta` function from the `datetime` module.

In summary, the code trains a Random Forest model on a dataset, evaluates its accuracy on the training data and through cross-validation, and reports the running time of the training process. This allows for an assessment of the model's performance and efficiency in classification tasks, particularly on datasets with high dimensionality.

# RESULTS

The project centered on employing machine learning techniques to address the issue of employee attrition within organizations. The study aims to predict employee attrition using logistic regression and various feature selection methods, recognizing the detrimental impact attrition can have on company operations and employee structure. The IBM HR Employee Attrition dataset, comprising 35 features relevant to attrition, serves as the basis for analysis.

The methodology involves implementing classification algorithms such as Logistic Regression, Support Vector Machine (SVM), K Nearest Neighbors (KNN), Perceptron, and Random Forest. Each algorithm is trained on the dataset and evaluated for accuracy, both on the training data and through cross-validation. Additionally, feature selection methods like information gain, select k-best, and recursive feature elimination (RFE) are utilized to identify influential factors contributing to attrition.

Results from the analysis are expected to provide insights into the most accurate predictive models for employee attrition and the most influential factors contributing to it. By comparing the performance of different algorithms and feature selection methods, the study aims to offer practical recommendations for organizations to mitigate attrition and retain valuable employees.

Overall, the study follows a structured approach to addressing the problem of employee attrition, emphasizing methodological rigor and practical implications for organizational management. Through machine learning techniques, the research seeks to provide actionable insights to support HR departments in their efforts to reduce attrition rates and improve employee retention.

# SUMMARY

The Project revolves around utilizing machine learning methods to tackle the significant challenge of employee attrition within organizations. Employee attrition, the gradual reduction in employees leaving a company, poses detrimental effects on organizational structures and project continuity. The study aims to predict employee attrition using logistic regression and various feature selection techniques, recognizing the importance of understanding influential factors to develop effective retention strategies.

The IBM HR Employee Attrition dataset serves as the foundation for analysis, containing 35 features pertinent to attrition. Feature selection methods such as information gain, select k-best, and recursive feature elimination (RFE) are proposed to identify key factors contributing to attrition and streamline the data training process. Cross-validation, particularly 10-fold cross-validation, is employed to evaluate predictive models' performance.

The methodology extends to implementing several classification algorithms, including Logistic Regression, Support Vector Machine (SVM), K Nearest Neighbors (KNN), Perceptron, and Random Forest. Each algorithm undergoes training on the dataset and evaluation for accuracy, both on the training data and through cross-validation. These algorithms represent a spectrum of approaches to classification, from simpler models like Logistic Regression to more complex ones like SVM and Random Forest.

Results from the analysis are expected to provide insights into the most accurate predictive models for employee attrition and the most influential factors contributing to it. By comparing the performance of different algorithms and feature selection methods, the study aims to offer practical recommendations for organizations to mitigate attrition and retain valuable employees.

The findings are anticipated to shed light on which machine learning approaches are most effective in predicting attrition and which features are most influential in driving employee turnover. This information can empower HR departments and organizational leaders to proactively address attrition issues and implement targeted retention strategies.

Overall, the study emphasizes methodological rigor and practical implications for organizational management. By leveraging machine learning techniques and exploring various classification algorithms, the research seeks to provide actionable insights to support HR departments in their efforts to reduce attrition rates and improve employee retention. Through a structured approach and thorough analysis, the study aims to contribute to the development of effective strategies for managing employee attrition in organizations.

# LINKS OF THE PROJECT

## GIT HUB LINK FOR PROJECT IPYNB FILE :

https://github.com/Saivamshichintham/2203a52221-AIML2221/blob/main/Attrition.ipynb

## GIT HUB LINK FOR PROJECT DATA SET :

https://github.com/Saivamshichintham/2203a52221-AIML2221/blob/main/Employee-Attrition.csv

## GIT HUB LINK FOR PROJECT DOCUMENTATION :

# THANK YOU

Team -

Ch. Sai Vamshi

G. Srivarsha

V. Rahul Reddy

M. Sujay Raj