Journal of Business and Management Studies
ISSN: 2709-0876
DOI: 10.32996/jbms
Journal Homepage: www.al-kindipublisher.com/index.php/jbms

JBMS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Strategic Employee Performance Analysis in the USA: Deploying Machine Learning Algorithms Intelligently

Nisha Gurung[1] ✉ Md Sumon Gazi[2] and Md Zahidul Islam[3]
[123]MBA Business Analytics, Gannon University, USA
Corresponding Author: Nisha Gurung, **E-mail**: nishagurung2056@gmail.com

| ABSTRACT
Strategic employee performance assessment assists organizations in steering productivity, affirming employee satisfaction, and accomplishing strategic organizational goals. Machine learning algorithms provide several benefits over mainstream techniques in assessing employee performance. This research paper aimed to explore the deployment of machine learning algorithms in assessing employee performance. The prime objective of employee performance analysis is to assess an employee's achievement during a specific time frame. The dataset for this research revolved around the leadership team of a global retailer's specific store level in the USA, extending over 18 months. The dataset for this study was subjected to Python programming software for intensive and comprehensive data analysis as well as for visualization purposes. From the experiment design, it was evident that XG-Boost seems to be the best-performing model overall. In particular, it had the greatest AUC for both holdout and training data (0.86 and 0.88, respectively), and it has a relatively low runtime (16 minutes) and maximum memory utilization (12%). By contrast, Random Forest displayed an average AUC for training data (0.79) but a lesser AUC for holdout data (0.51), which indicates that it may be overfitting the training data; besides, it had a longer runtime than XG-Boost.

| KEYWORDS

Employee performance; Machine Learning; Linear Discriminant Analysis; Logistic Regression; Support Vector Machines; XG-Boost; Random Forest; Naïve Bayes; k-Nearest Neighbor.

| ARTICLE INFORMATION

ACCEPTED: 25 April 2024 **PUBLISHED:** 05 May 2024 **DOI:** 10.32996/jbms.2024.6.3.1

1. Introduction

Mourad (2022) indicates that employee performance evaluation is a paramount element of human resource management since it helps companies in America pinpoint high-performing staff, pinpoint areas for adjustment, and make data-oriented decisions. Traditional techniques of performance analysis frequently suffer from bias and subjectivity, leading to inaccurate evaluations. Nevertheless, with the advancement of machine learning, businesses in the USA can leverage the power of data to address these limitations and obtain more accurate and objective evaluations. This study aims to examine the deployment of machine learning algorithms in assessing employee performance.

Strategic employee performance assessment assists organizations in steering productivity, affirming employee satisfaction, and accomplishing strategic organizational goals. Conventional techniques for performance assessment, such as supervisor ratings and self-evaluation, are frequently subjective and biased. Machine learning models offer several benefits over mainstream techniques in assessing employee performance. Firstly, they can tackle complex and large datasets, facilitating the inclusion of an array of factors and variables that impact performance (Mourad, 2022). By utilizing innovative data processing methods, machine learning algorithms can detect correlations, patterns, and hidden information that may not be visible to human analysts. As a result, this reinforces the reliability and accuracy of performance analysis.

2. Literature Review

As per Nayem & Uddin (2024), the principal goal of employee performance analysis is to assess an employee's achievement during a specific time frame. These assessments aim to ascertain benchmarks that staff can commit to accomplish to contribute to the company's strategic objectives. As part of this research, the researcher reviewed three different sets of factors that influence employee performance. These factors entail environmental or physical factors, behavioral factors, and economic factors.

Patel et al. (2022), in their study, found that training, job security, and job stability significantly enhanced employee performance in the manufacturing sector. Patel et al. (2022) investigated the association between performance and social factors, encompassing work-life balance, training company tenure, and commute distance. Arasi and Babu (2023) classified staff into low and high performers based on leadership, achievement, and behavioral elements, underscoring discipline, attendance, and work output in the accomplishment category.

Recent research has witnessed an upsurge in the use of AI to tackle important HRM issues. However, the algorithmic approaches and solutions proposed by these researchers may still carry biases and overlook key dynamic environmental factors related to employee performance assessment. Patel et al. (2022) aimed to aid organizations in achieving more accurate employee performance evaluations by comparing various AI algorithms like XG-Boost, decision trees, random forests, and artificial neural networks. They introduced an ensemble approach named RanKer that combined these methods. Despite the substantial repercussions of environmental components, like supervisor support, workplace safety, and rewards, on employee performance, some of these factors were not entirely addressed in their research.

Punnoose (2022) proposes a technique utilizing XGBoost and gradient boosting to predict staff performance within a multinational corporation. While they performed statistical tests, they overlooked the computation of a correlation matrix. Turukmane (2022) similarly proposed an ensemble machine learning approach for predicting employee productivity and suggested statistical analysis. Patel et al. (2022) adopted multiple machine learning methods to predict staff attrition, employing a sectorial-standard real-time dataset from IBM for algorithm testing and training. The research pinpointed a prevalent oversight in recent research, where the impact of dynamic social, physical, environmental, and economic factors on employee performance assessment remains largely unaddressed, raising issues regarding bias in such evaluations.

On the other hand, Tansescu (2024) adopted a neuro-fuzzy framework to distinguish between low- and high-performing staff, discovering that the framework optimizes the goal function in workers' quality evaluation. This Artificial Intelligence technique specifically pinpoints employees needing career development and further training in leadership, achievement, and behavior categories.

3. Methodology

3.1 Dataset Description

The dataset for this study focused on the leadership team of a global retailer's specific store level in the USA, extending over 18 months. The disseminated sample comprised of different US locations, with data gathered quarterly. Two categories of labels, Terminated and Active, were allocated as 0 and 1, respectively (Pro-AI-Rokibul, 2024). Every person had a register for every quarter of active participation in the company, which converted category labels from active to terminated if attrition appeared during that time frame. The dataset constitutes 73,115 data points, with everyone labeled as terminated or active.

3.2 Feature Engineering and Selection

The dataset's features were chosen premised on the company's database and sourced from two avenues, most notably, the HRIS database of the company and the Bureau of Labor Statistics (BLS). The HRIS database provides necessary features, comprising demographic attributes such as age and compensation-associated features such as pay and team-associated features such as peer attrition (Pro-AI-Rokibul, 2024). The BLS data aided the research with crucial features, such as the unemployment rate and median household income.

Pre-processing

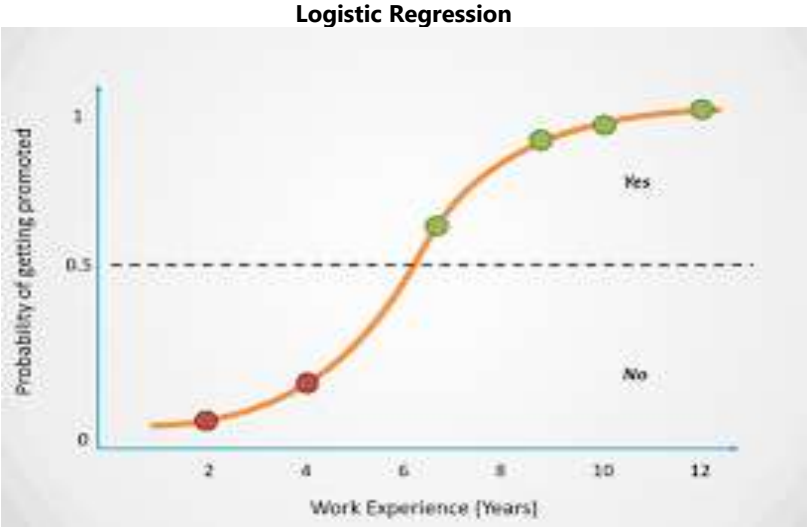
```
In [5]: # Check for missing or null values across the entire dataset
        # This creates a DataFrame with True/False indicating missing values
        null_check = df.isnull()

        # Check if there are any null values in the entire dataset
        has_nulls = null_check.any().any()  # True if any null values exist

        print("Are there any null values in the dataset?", has_nulls)

        # To get a more detailed view, you can check which columns have missing values
        missing_values_per_column = df.isnull().sum()

Are there any null values in the dataset? False

In [6]: print("Number of missing values per column:")
        print(missing_values_per_column)
```

Definite variables with missing values were completed utilizing the approach of the respective sector; conversely, numerical variables were imputed independently. Zero-imputation was particularly employed in sectors such as the number of promotions to prevent skewing data associated with employee promotions. Domain proficiency guided the imputation of particular numeric domains; for instance, the time since the last promotion was approximated utilizing tenure-in-position for accuracy. Median imputation was selected for particular numeric variables to regulate outliers effectively, unlike mean imputation. As a portion of data preprocessing, definite features went through One-Hot Encoding, converting each distinct value into binary fields.

3.3 Model Validation

The dataset was subdivided into holdout and training sets in an 80:20 ratio. A grid search was performed to enhance tuning parameters, such as penalty hyper-parameters or regularization, for every model. The best hyper-parameter context for every algorithm was set through a 10-fold cross-validation on the training set. Consequently, the models were trained using these optimal configurations on the training data. The trained algorithm from every algorithm was then employed to forecast and evaluate performance on the 20% holdout sample.

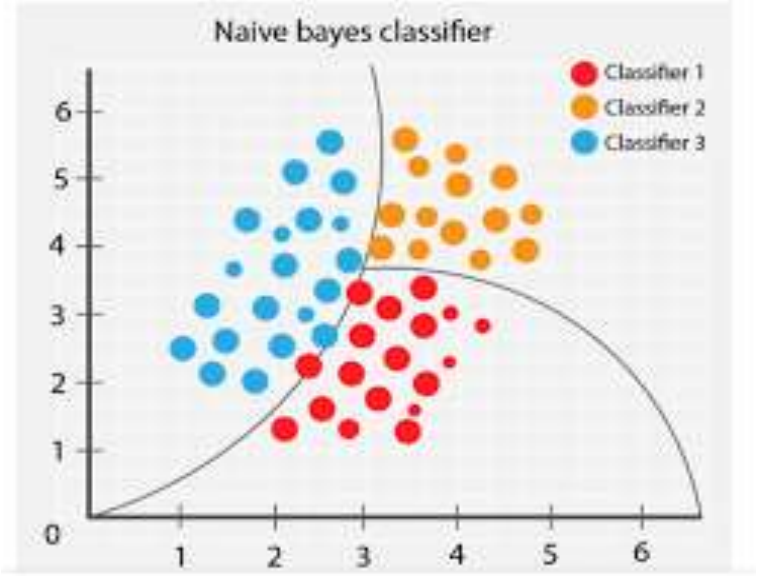
Models and Metrics
Logistic Regression



Logistic regression is a fundamental linear algorithm for classifying tasks. It thrives in forecasting binary or classifying dependent variables. Regularization methods, such as L1-norm or L2-norm penalties, are frequently applied to hamper overfitting in logistic regression. For this research, an L2-regularized logistic regression was deployed (Pro-AI-Rokibul, 2024). These techniques compute posterior probabilities by presuming a particular algorithm and approximating the parameters involved in that algorithm. The algorithm's expression is presented in (1) below:

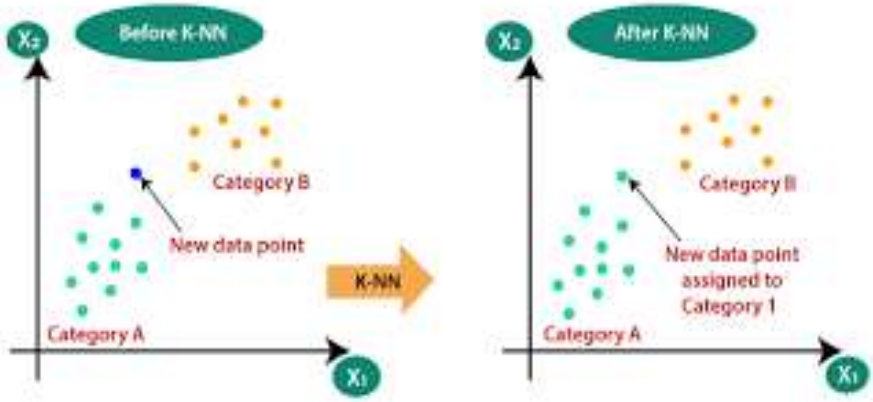
$$p(\text{churn}|w) = \frac{1}{1+ e^{-\left[w0+ \sum_{i=1}^{N} wiXi\right]}}$$

Naïve Bayes



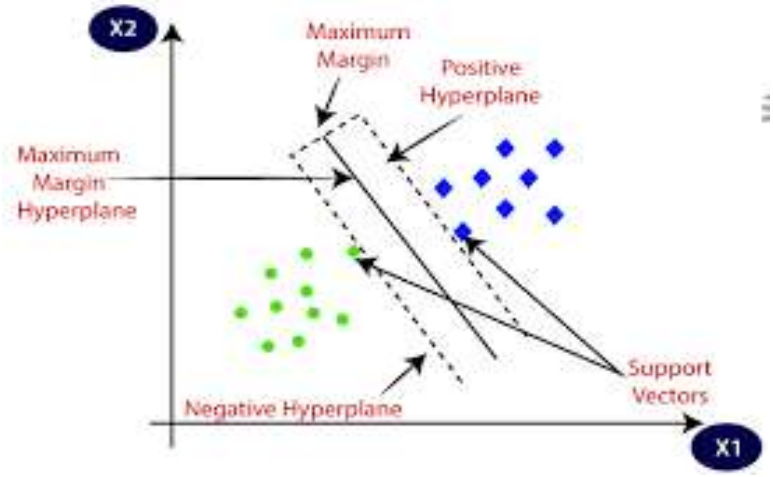
Naïve Bayes is a popular categorization technique renowned for its effectiveness and simplicity. Naïve Bayes categorizes based on probabilities, assuming conditional independence among all variables. The Naïve Bayes classifier demands less training data to approximate the essential parameters (variances and means of the variables) for classification. This method accommodates both discrete and real data types, making it versatile for different data scenarios. The implementation of Bayes' rule entails training a target function fn: X → Y, corresponding to P(Y|X). By employing training data, the researcher approximates P(X|Y) and P(Y) to learn the fundamental distributions (Pro-AI-Rokibul, 2024). With this probability approximation and Bayes' rule, new X samples can be efficiently classified based on the learned association between X and Y.

K-Nearest Neighbor (KNN)



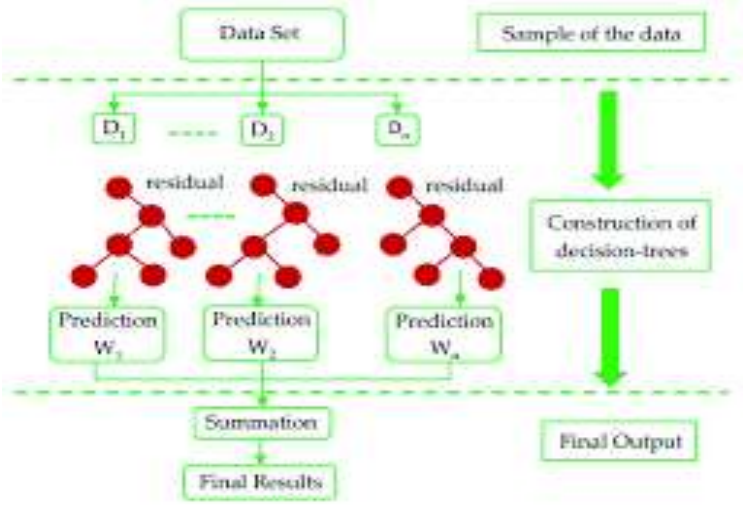
Nearest Neighbor Classification functions by classifying data points grounded on the category of their closest neighbors, frequently integrating multiple neighbors for enhanced accuracy, termed as K-nearest Neighbor (k-NN) categories. The classification procedures comprise two phases: pinpointing neighboring data points and ascertaining the class premised on the classes of these neighbors (Pro-AI-Rokibul, 2024). The neighbors can be pinpointed utilizing distance measures, such as Euclidean distance, to evaluate the proximity of data points.

Support Vector Machine (SVM)



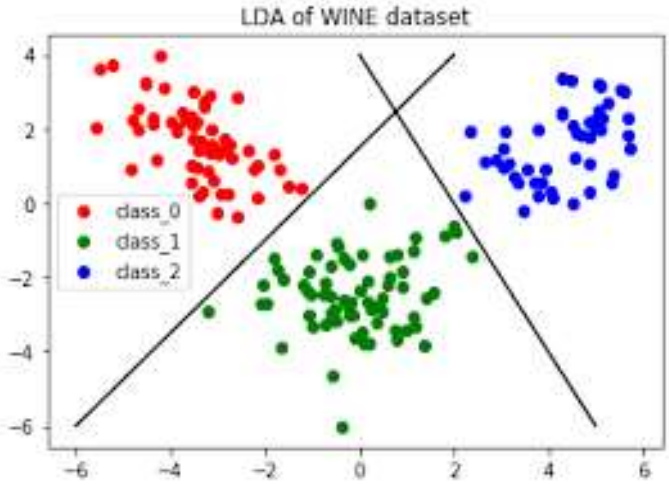
SVM is a supervised learning model premised on a statistical learning framework that addresses both nonlinear and linear binary categorization tasks. By creating hyper-planes in higher-dimensional dimensions, SVM targets efficiently separate classes. The vital notion is to locate a hyper-plane with the biggest distance to the nearest training data points of any category, as a bigger margin leads to lesser generalization errors, giving SVM the moniker of a supreme margin classifier (Pro-Al-Rokibul, 2024). Before algorithm designing, the data went through scaling to fit within the range.

Extreme Gradient Boosting (XG-Boosting)



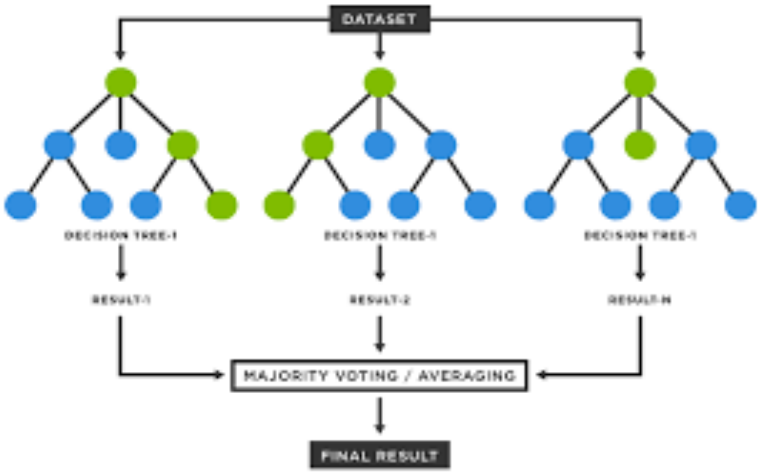
XG-Boost is a gradient-boosting model premised on decision trees. It differentiates itself from other gradient-boosted algorithms through a more monitored algorithm formalization, which prevents overfitting and improves performance. The principal learning goal comprises ascertaining the functions fi, each involving the tree structure and leaf scores.

Linear Discriminant Analysis



Linear Discriminant analysis involves developing one or more discriminant functions to elevate the variance between groups contrasted to the variance within categories. Linear Discriminant Analysis particularly concentrates on developing a variate or z-score, which is a linear consolidation of at least two individual variables created to optimally differentiate between two (or more) distinct groups or categories.

Random Forest



Random Forest is a popularly used ensemble learning method depending on tree-based algorithms. The 'ensembling' technique comprises bagging, where every tree is individually developed utilizing a distinct bootstrap sample of the dataset without depending on earlier trees. Eventually, a simple majority vote is employed for prediction. Random Forests vary from standard trees in that they randomly choose a subset of predictors for dividing at each node, presenting an extra layer of randomness that improves robustness against overfitting.

3.4 Experimental Design

The dataset for this study was subjected to Python programming software for intensive and comprehensive data analysis as well as for visualization purposes. Python software has been widely used across the globe to process large-volume datasets to make connections and make data-driven inferences.

Importing Libraries

In [1]:
```
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import pandas as pd
import warnings

# Suppress all warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df = pd.read_csv("Employee_Attrition.csv")
df
```

Output:

Out[2]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	Employ
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	
...	...	...	...	...	...	...	...	...	...	
1465	36	No	Travel_Frequently	884	Research & Development	23	2	Medical	1	
1466	39	No	Travel_Rarely	613	Research & Development	6	1	Medical	1	
1467	27	No	Travel_Rarely	155	Research & Development	4	3	Life Sciences	1	
1468	49	No	Travel_Frequently	1023	Sales	2	3	Medical	1	
1469	34	No	Travel_Rarely	628	Research & Development	8	3	Medical	1	

1470 rows × 35 columns

The method of a Pandas data frame was applied to the dataset to exhibit a clear summary of the data frame, which was useful for obtaining a quick overview of its structure and contents. The results can be displayed as follows:

```
In [3]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64
 9   EmployeeNumber            1470 non-null   int64
 10  EnvironmentSatisfaction   1470 non-null   int64
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64
 13  JobInvolvement            1470 non-null   int64
 14  JobLevel                  1470 non-null   int64
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64
 19  MonthlyRate               1470 non-null   int64
 20  NumCompaniesWorked        1470 non-null   int64
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64
 24  PerformanceRating         1470 non-null   int64
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

Output:

Out[4]:

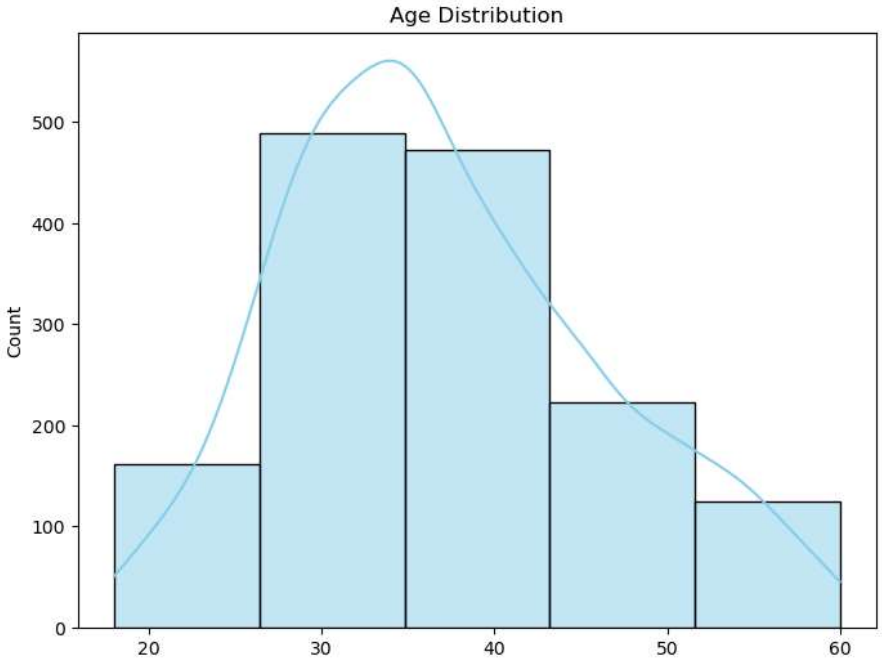
	Age	DailyRate	DistanceFromHome	Education	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	HourlyR
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.0	1470.000000	1470.000000	1470.000
mean	36.923810	802.485714	9.192517	2.912925	1.0	1024.865306	2.721769	65.891
std	9.135373	403.509100	8.106864	1.024165	0.0	602.024335	1.093082	20.329
min	18.000000	102.000000	1.000000	1.000000	1.0	1.000000	1.000000	30.000
25%	30.000000	465.000000	2.000000	2.000000	1.0	491.250000	2.000000	48.000
50%	36.000000	802.000000	7.000000	3.000000	1.0	1020.500000	3.000000	66.000
75%	43.000000	1157.000000	14.000000	4.000000	1.0	1555.750000	4.000000	83.750
max	60.000000	1499.000000	29.000000	5.000000	1.0	2068.000000	4.000000	100.000

8 rows × 26 columns

To compute the age group of the employees in the provided dataset, a code snippet was imposed to generate a histogram of the age-bracket distribution of the employees, as showcased below:

```
In [7]: # Histogram of Age
plt.figure(figsize=(8, 6))
sns.histplot(df['Age'], kde=True, bins=5, color='skyblue')
plt.title("Age Distribution")
plt.xlabel("Age")
plt.ylabel("Count")
plt.show()
```

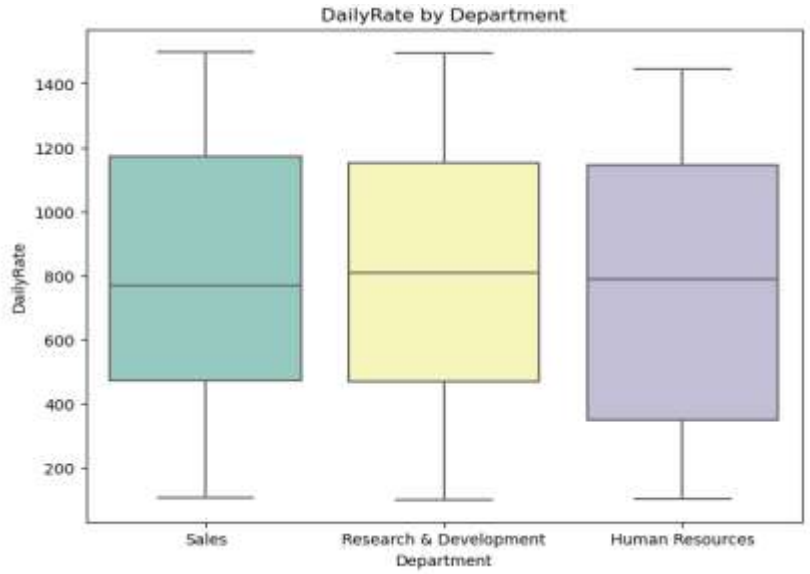
Output:



To visualize the daily rates by department, the analyst applied the appropriate code snippet to achieve the desired goal:

Output:

```
In [8]:
# Box Plot of DailyRate by Department
plt.figure(figsize=(8, 6))
sns.boxplot(x='Department', y='DailyRate', data=df, palette='Set3')
plt.title("DailyRate by Department")
plt.xlabel("Department")
plt.ylabel("DailyRate")
plt.show()
```



To ascertain the age group by daily rate attrition, a scatter plot was generated to determine the relationships between these variables:

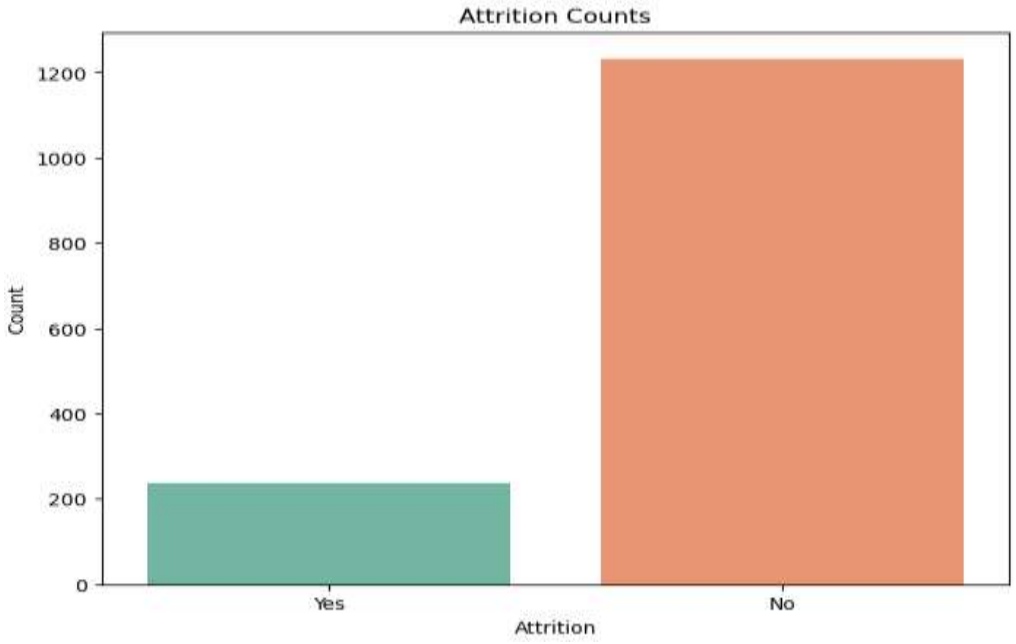
Output:

In [9]:
```
# Scatter Plot of Age vs DailyRate
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Age', y='DailyRate', hue='Attrition', data=df, palette='Set1')
plt.title("Age vs DailyRate (by Attrition)")
plt.xlabel("Age")
plt.ylabel("DailyRate")
plt.show()
```

Subsequently, to compute the attrition counts, the analyst implemented a code snippet to display the bar plot showcasing the relationship:

In [10]:
```
# Bar Plot of Attrition Counts
plt.figure(figsize=(8, 6))
sns.countplot(x='Attrition', data=df, palette='Set2')
plt.title("Attrition Counts")
plt.xlabel("Attrition")
plt.ylabel("Count")
plt.show()
```

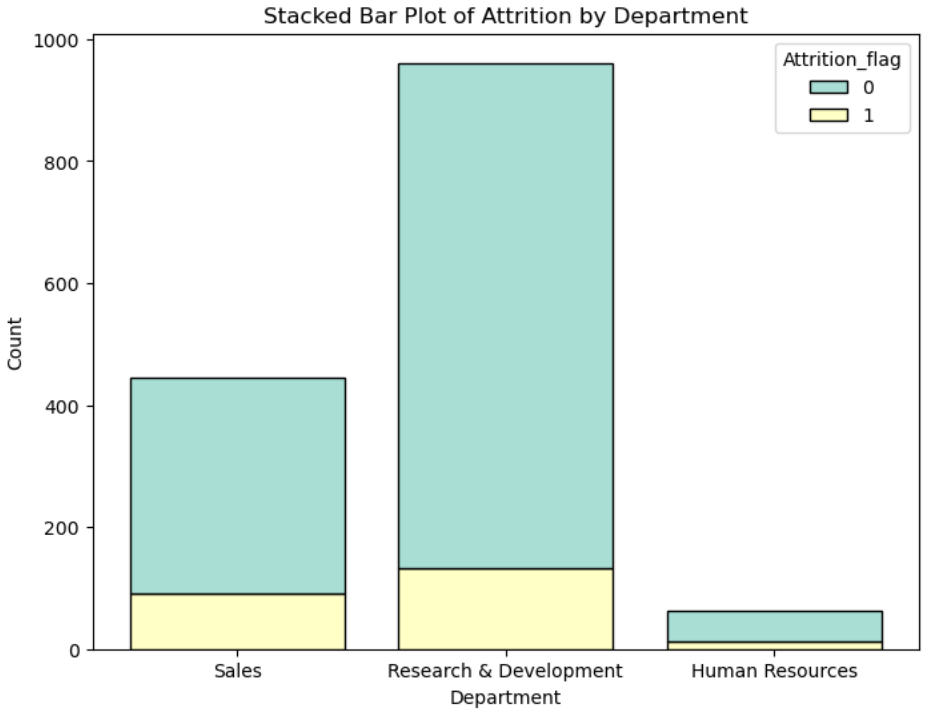
Output:



To visualize the correlation attrition rate, the analyst generated a bar plot to display turnover rates by department:

```
In [12]: # Stacked Bar Plot of Attrition by Department
plt.figure(figsize=(8, 6))
df['Attrition_flag'] = df['Attrition'].map({'Yes': 1, 'No': 0})  # Convert to numeric
sns.histplot(data=df, x='Department', hue='Attrition_flag', multiple='stack', palette='Set3', shrink=0.8)
plt.title("Stacked Bar Plot of Attrition by Department")
plt.xlabel("Department")
plt.ylabel("Count")
plt.show()
```

Output:



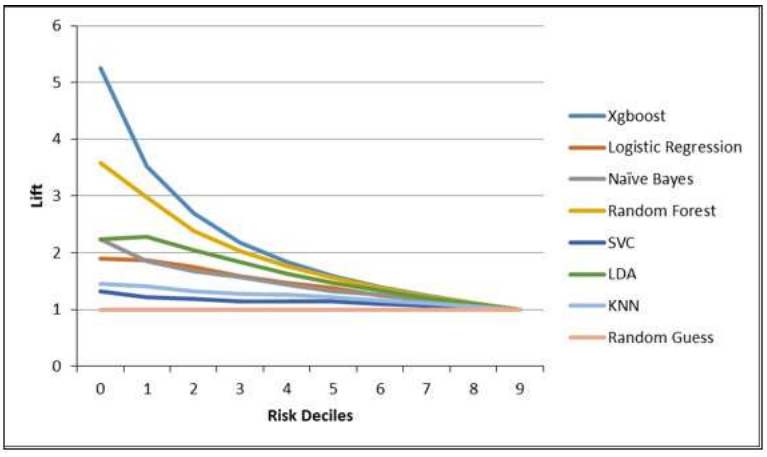
Models Performance Evaluation

The selected metric for comparing the performance accuracies of the models in this setting is the Area under the Receiver Operating Characteristic curve (ROC-AUC). This metric of 'predictiveness' facilitates a comparison of models' factors, such as class distributions and misclassification costs. Unlike measures such as error rate, AUC evaluates the likelihood that a classifier will compute a randomly chosen positive scenario greater than a randomly chosen negative one, akin to the Wilcoxon test of ranks. Model memory utilization and run-time are also employed to contrast the performance of the algorithms.

Algorithm	AUC (Training)	AUC (Holdout)	Run-time (Training)	Maximum Memory Utilization (Of 16 GB)
XGBoost	**0.88**	**0.86**	16 min 12 sec	**12%**
Logistic Regression	0.66	0.50	**52 sec**	20%
Naïve Bayesian	0.64	0.59	59 sec	20%
Random Forest (Depth controlled)	0.79	0.51	23 min 10 sec	29%
SVM (RBF kernel)	0.68	0.52	105 min 30 sec	21%
LDA	0.74	0.52	6 min 51 sec	35%
KNN (Euclidean distance)	0.52	0.5	180 min 12 sec[a]	35%

From the above table, it was evident that XG-Boost seems to be the best-performing model overall. In particular, it had the greatest AUC for both holdout and training data (0.86 and 0.88, respectively), and it has a relatively low runtime (16 minutes) and maximum memory utilization (12%). By contrast, Random Forest displayed an average AUC for training data (0.79) but a lesser AUC for holdout data (0.51), which indicates that it may be overfitting the training data; besides, it had a longer runtime than XG-Boost. Conversely, Naive Bayes and Logistic Regression have lower AUCs than XG-Boost, but they are also much faster to train and use less memory.

A lift chart was used to evaluate the risk ranking of staff in each decile to validate the algorithm's performance. The lift chart assists in visualizing the modification imposed on the model compared to a random guess.



From the figure above, it was apparent that the XG-Boost algorithm had better decile performance than other algorithms till the 7th decile (inclusive). It is also considerably and consistently better than the other algorithms.

4. Business Impact

4.1 Benefits of Adopting the Proposed Model on Business in the USA

1. **Improved Talent Management:** The XG-Boost algorithm can evaluate staff data to pinpoint individuals with a high probability of quitting. This enables companies in the USA to proactively resolve possible issues, grant retention incentives, or invest in upskilling programs.
2. **Intelligent Decision-Making:** The XG-Boost algorithm can forecast future employee needs based on historical trends and data. This enables companies to optimize recruitment efforts and resource allocation and expect potential skills gaps.
3. **Efficiency and Scalability**: XG-Boost can manage large volumes of datasets efficiently, making it perfect for companies with numerous employees.

4.2 Benefits to the USA Economy

1. **Enhanced Productivity and Efficiency**: By assisting government companies in pinpointing high possibilities and at-risk personnel, the XG-Boost system can contribute to a more effective allocation of human resources throughout the nation. As a result, the US government can optimize teams, reduce turnover costs, and affirm that the right talent is placed in the right roles.
2. **Boosting R&D:** XG-Boost's model capability to manage large datasets can help in research and development (R&D) schemes. For example, it can be utilized to assess a large volume of scientific data, identify trends, and elevate innovation across sectors.
3. **Streamlining Business Processes:** The XG-Boost model can be deployed to optimize various business processes, from supply and logistic chain management to customer service and marketing. As a result, this can facilitate faster turnaround times, cost reductions, and a more competitive US economy.

4.3 How to Use the Model

1. **Step 1-Data Preparation/Processing:** The business analyst should first gather and preprocess your dataset. This may comprise tackling missing values, scaling numerical features, encoding categorical variables, and splitting the data into testing and training sets.
2. **Step 2-Define Training and Testing Sets:** Business analysts should split Data by dividing the data into training and testing sets. The training set should be utilized to build the algorithms, while the testing set should be utilized to assess its performance on unseen data. For instance, the analyst should split the dataset in the proportion 80/20 (training/testing).
3. **Step 3-Algorithm Training:** Business analysts should proceed to import the XG-Boost library and initiate the XG-Boost model object. Subsequently, the analyst should use suitable hyperparameters such as the learning rate, tree depth, number of trees, and regularization.
4. **Step 4- Model Evaluation:** Afterwards, the analyst utilizes the trained algorithm to make forecasts on the testing set utilizing the system. They should assess the algorithm's performance on the testing set by employing metrics such as Precision, accuracy, AUC, and recall. Libraries such as sci-kit-learn provide platforms for these calculation**s.**
5. **Step 5-Model Deployment:** Once the business analyst is satisfied with the algorithm's performance, they can preserve the trained algorithms and use them to forecast new, churning data. The strategic benefits of utilizing XG-Boost are its scalability, speed, and capability to manage large, complex datasets effectively.

5. Conclusion

This study aimed to explore the deployment of machine learning algorithms in assessing employee performance. The principal objective of employee performance analysis is to assess an employee's achievement during a specific time frame. The dataset for this research revolved around the leadership team of a global retailer's specific store level in the USA, extending over 18 months. The dataset for this study was subjected to Python programming software for intensive and comprehensive data analysis as well as for visualization purposes. From the experiment design, it was evident that XG-Boost seems to be the best-performing model overall. In particular, it had the greatest AUC for both holdout and training data (0.86 and 0.88, respectively), and it has a relatively low runtime (16 minutes) and maximum memory utilization (12%). By contrast, Random Forest displayed an average AUC for training data (0.79) but a lesser AUC for holdout data (0.51), which indicates that it may be overfitting the training data; besides, it had a longer runtime than XG-Boost. On the other hand, Naive Bayes and Logistic Regression have lower AUCs than XG-Boost, but they are also much faster to train and use less memory.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] Mourad, Z. (2022, May 18). *Towards a new method for classifying employee performance using machine learning algorithms*. IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9806118

[2] Nayem, Z., & Uddin, M. A. (2024). Unbiased employee performance evaluation using machine learning. *Journal of Open Innovation*, 100243. https://doi.org/10.1016/j.joitmc.2024.100243

[3] Patel, K., Sheth, K., Mehta, D., Tanwar, S., Florea, B., Ţarălungă, D. D., Altameem, A., Altameem, T., & Sharma, R. (2022). RANKEr: An AI-Based Employee-Performance Classification scheme to rank and identify low performers. *Mathematics*, *10*(19), 3714. https://doi.org/10.3390/math10193714

[4] Pro-AI-Rokibul. (2024). *Employee-Performance-Analysis/Model/Employee_Performacne.ipynb at main · proAIrokibul/Employee-Performance-Analysis*. GitHub. https://github.com/proAIrokibul/Employee-Performance-Analysis/blob/main/Model/Employee_Performacne.ipynb

[5] Punnoose, R. (2022). Prediction of Employee Turnover in Organizations using Machine Learning Algorithms. *www.academia.edu*. https://www.academia.edu/76526150/Prediction_of_Employee_Turnover_in_Organizations_using_Machine_Learning_Algorithms?sm=b

[6] Tanasescu, L. G., Vines, A., Bologa, A. R., & Vîrgolici, O. (2024). Data Analytics for optimizing and predicting employee performance. *Applied Sciences*, *14*(8), 3254. https://doi.org/10.3390/app14083254

[7] Turukmane, A. (2022). Experimental performance analysis of machine learning algorithms. *www.academia.edu*. https://www.academia.edu/113952890/Experimental_Performance_Analysis_of_Machine_Learning_Algorithms?sm=b