Final Report

Spring 2024

Team #06

Honor Brogden - hobr2220@colorado.edu
Megha Gupta - megu2186@colorado.edu
Nathan Kareithi - naka6894@colorado.edu
Jackson Miers - jami1731@colorado.edu
Maddie Wallace - mawa9164@colorado.edu
Fizza Zaidi - fiza2251@colorado.edu

Business Understanding

[redacted] HR team is dedicated to advancing its data analysis methodology by integrating predictive analytics and machine learning techniques. This transformative endeavor is designed to elevate both the efficiency and precision in handling and interpreting employee data. By transitioning from conventional approaches to more refined analytics, the HR team is poised to accurately forecast employee turnover and proactively plan for future staffing requirements. This initiative not only aims to tackle the complexities of disparate data systems and operational inefficiencies but also aims to harness predictive insights to inform strategic workforce planning and decision-making processes.

Problem Statement

[redacted] HR team seeks to enhance its data analysis capabilities by incorporating predictive analytics and machine learning into their analysis procedures. They face challenges in integrating and analyzing employee data from multiple systems efficiently. Current methods, such as Excel, are time-consuming and lack repeatability. The team aims to apply predictive analytics and ML to identify drivers of voluntary churn and forecast the human capital needed for a 2-year hiring pipeline.

Overarching Objectives

- Incorporate predictive analytics and ML into data analysis within [redacted] HR
- Integrate, clean, and analyze employee data
- Improve efficiency and repeatability of the data analysis process
- Apply predictive analytics to identify key drivers of voluntary churn
- Forecast human capital required to build a hiring pipeline for the next two years

Data Cleaning Introduction

Our data refinement journey consisted of many specific steps. It began with basic exploratory analysis to gain a deep understanding of the dataset's structure and content. In this initial phase, we focused on using functions to gain insights into the data and refining column names for clarity and ease of use, ensuring a seamless experience throughout the data cleaning process. Following this, we took proactive measures to handle missing data, strategically dropping rows with null values to maintain data integrity. Following this, we delved into feature engineering, creating new meaningful features. Our attention to detail extended to data sanity checks, where we identified and rectified anomalies like outliers, ensuring data accuracy and reliability. We also created necessary data frames that will be of great use once we begin creating our ML models. Through a systematic and meticulous approach, our data refinement efforts culminated in a cleaner and more structured dataset, laying a robust groundwork for our subsequent analysis and modeling endeavors.

Data Cleaning Steps

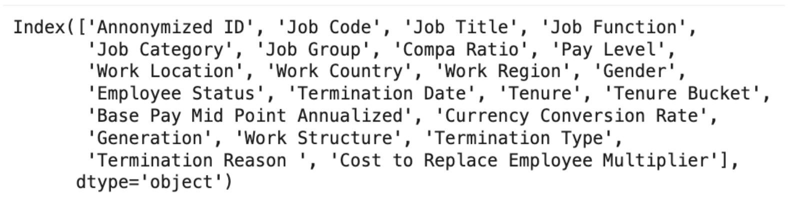
Step 1: Basic Analysis

Our data-cleaning journey started with a foundational step in which we performed a basic exploratory analysis of the dataset. This preliminary analysis involved utilizing functions to gain insights into the current state of our data, including summary statistics and data types of columns. The significance of this initial step lies in providing our team with a clear understanding of the starting point for our data-cleaning process. It serves as a crucial reference point that guides us in evaluating subsequent data cleaning steps effectively. Additionally, we employed the isnull() function to identify columns containing missing values and ascertain the extent of missing data. This exploration revealed that the dataset exhibits relatively clean data with only a few columns containing a small number of missing values. This introduction sets the stage for a systematic and informed approach to data cleaning and preparation for further analysis.

Step 2: Column Name Refinement

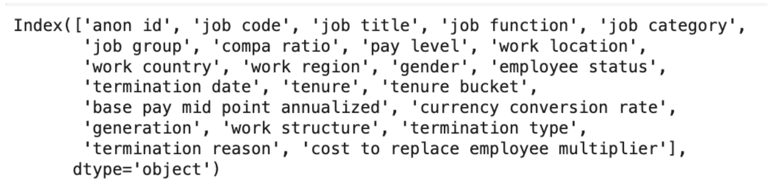
After conducting a thorough basic analysis of the dataset, our focus shifted toward refining column names for clarity and consistency. We began by standardizing all column names to lowercase, enhancing readability and reducing potential typing errors during data manipulation. Furthermore, we addressed naming inconsistencies and typos by renaming specific columns. We changed the column name "anonymized id" to "anon id", correcting a typo in the original name and creating a column name that is easier to work with without losing the value of its name.

The final column name we changed was for ease of future steps and model processing. We changed the column name "termination reason " to "termination reason". Notice here, the original column name contained a space after the word "reason". This extra space prevented Python from recognizing the column name if the extra space was not included.



```
Index(['Annonymized ID', 'Job Code', 'Job Title', 'Job Function',
       'Job Category', 'Job Group', 'Compa Ratio', 'Pay Level',
       'Work Location', 'Work Country', 'Work Region', 'Gender',
       'Employee Status', 'Termination Date', 'Tenure', 'Tenure Bucket',
       'Base Pay Mid Point Annualized', 'Currency Conversion Rate',
       'Generation', 'Work Structure', 'Termination Type',
       'Termination Reason ', 'Cost to Replace Employee Multiplier'],
      dtype='object')
```

Figure 1: Columns before column name editing



```
Index(['anon id', 'job code', 'job title', 'job function', 'job category',
       'job group', 'compa ratio', 'pay level', 'work location',
       'work country', 'work region', 'gender', 'employee status',
       'termination date', 'tenure', 'tenure bucket',
       'base pay mid point annualized', 'currency conversion rate',
       'generation', 'work structure', 'termination type',
       'termination reason', 'cost to replace employee multiplier'],
      dtype='object')
```

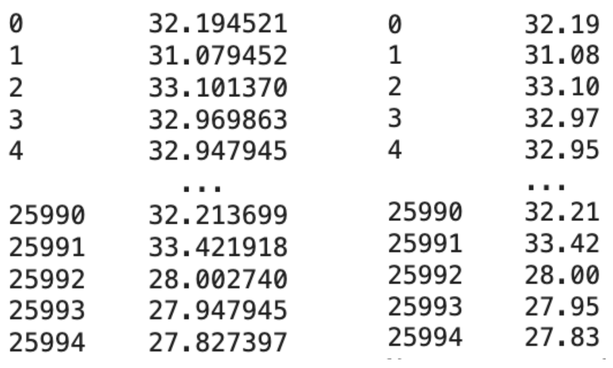
Figure 1.2: Columns after column name editing

Step 3: Dropping Rows

After changing the column names, we chose to drop NA's, otherwise known as nulls or null values. These are cells in the data missing values. The dataset provided to us did not contain many uncalled-for missing values, but we did examine the dataset further and determined some missing values would be better off being removed. After removing these null values, we dropped any duplicate rows to ensure our analytics would not be influenced by redundant data.

Step 4: Editing Columns for Clarity

We felt that for readability and ease of use, the "tenure" column would benefit from rounding. We chose to round the "tenure" column to two decimal places:



```
0        32.194521      0        32.19
1        31.079452      1        31.08
2        33.101370      2        33.10
3        32.969863      3        32.97
4        32.947945      4        32.95
           ...                   ...
25990    32.213699      25990    32.21
25991    33.421918      25991    33.42
25992    28.002740      25992    28.00
25993    27.947945      25993    27.95
25994    27.827397      25994    27.83
```

Figure 2: 'tenure' before and after editing

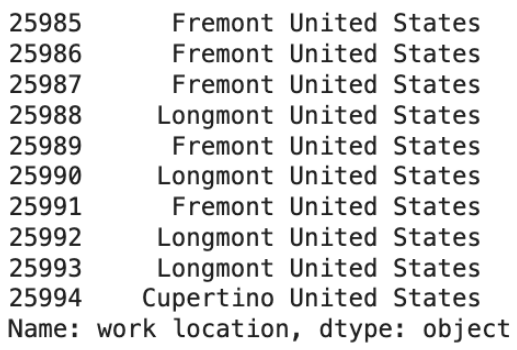
This cleaning step did change the value counts for each "tenure" count because of the rounding. However, for our purposes, we deemed this okay, as it will be much cleaner and easier to analyze.

Step 5: Feature Engineering

Feature engineering is essential in machine learning as it enhances model performance by providing relevant and informative features, improving data representation, and reducing dimensionality. By incorporating domain knowledge and handling missing values effectively, feature engineering ensures robust and accurate model predictions while also increasing model interpretability. Once we had edited columns and rows to our satisfaction, we were ready to begin feature engineering. This is the process of creating new features using existing features.

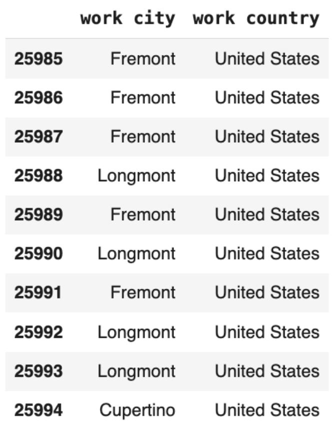
- Editing Work City, Work Country, and Work Structure:
  - "work location" column contained information in the form of [city country]. Having the country name in this column proved redundant since the dataset already contained a "work country" column. Because of this, we removed any country name that appeared in the "work location" column. Then we renamed the "work location" column to "work city".After completing this step, we were

left with the following clean columns regarding work location: “work city” and “work country”.



```
25985      Fremont United States
25986      Fremont United States
25987      Fremont United States
25988     Longmont United States
25989      Fremont United States
25990     Longmont United States
25991      Fremont United States
25992     Longmont United States
25993     Longmont United States
25994    Cupertino United States
Name: work location, dtype: object
```

Figure 3: work location before editing



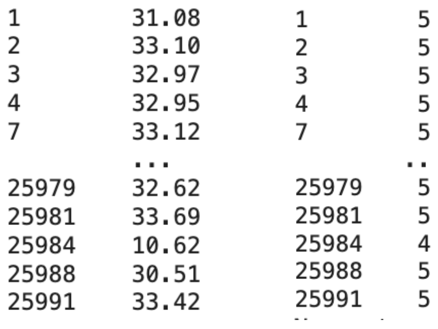
	work city	work country
25985	Fremont	United States
25986	Fremont	United States
25987	Fremont	United States
25988	Longmont	United States
25989	Fremont	United States
25990	Longmont	United States
25991	Fremont	United States
25992	Longmont	United States
25993	Longmont	United States
25994	Cupertino	United States

Figure 3.1: work city and country after editing

- Updated 'work city' names for clarity and consistency. For example, standardized names like 'remote - british columbia' to 'british columbia'.
- Filled in any null values in ‘work structure’ with ‘Onsite’ per client clarification.

➔ Tenure Binning:

- Binned the 'tenure' column into categories based on predefined bins (e.g., [0, 1, 5, 10, 20]) and labels (e.g., ['<1', '1-5', '5-10', '10-20', '20+']).



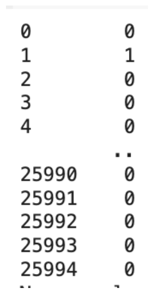
```
1        31.08        1        5
2        33.10        2        5
3        32.97        3        5
4        32.95        4        5
7        33.12        7        5
         ...                   ..
25979    32.62        25979    5
25981    33.69        25981    5
25984    10.62        25984    4
25988    30.51        25988    5
25991    33.42        25991    5
```

Figure 4: ‘tenure’ before and after binning

➔ Feature Engineering:

- Calculated the 'cost to replace employee' by multiplying 'base pay mid point annualized' with 'cost to replace employee multiplier'.
- Converted 'base pay mid point annualized' to USD using 'currency conversion rate'.
    - Dropped now unnecessary columns ('base pay mid point annualized', 'currency conversion rate').
- Created a binary ‘vol churn' column based on 'termination type'.

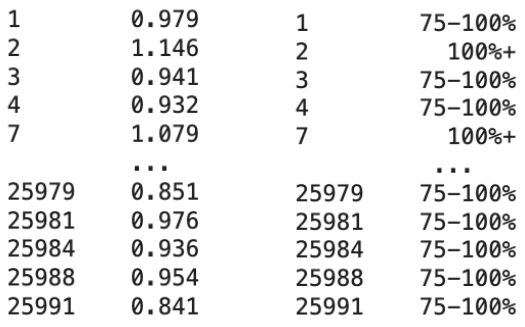
- This inserted a new column that contained a '1' if the employee voluntarily churned, and a 0 otherwise.



```
0        0
1        1
2        0
3        0
4        0
        ..
25990    0
25991    0
25992    0
25993    0
25994    0
```

Figure 5: 'vol_churn' column

➔ Handling Compa Ratio:
- ◆ Binned the 'compa ratio' column into categories based on predefined bins (e.g., [0, 0.25, 0.5, 0.75, 1]). Defined labels for each bin range.



```
1        0.979      1        75-100%
2        1.146      2          100%+
3        0.941      3        75-100%
4        0.932      4        75-100%
7        1.079      7          100%+
         ...                    ...
25979    0.851      25979    75-100%
25981    0.976      25981    75-100%
25984    0.936      25984    75-100%
25988    0.954      25988    75-100%
25991    0.841      25991    75-100%
```

Figure 6: 'compa ratio' before and after binning

Step 6: Sanity Checking

The next step we determined was necessary for cleaning the data was sanity-checking our values. This means checking to make sure that values are within the ranges they should be. One place we found an error here was within the "compa ratio". When we created a histogram for this feature this is what we saw:

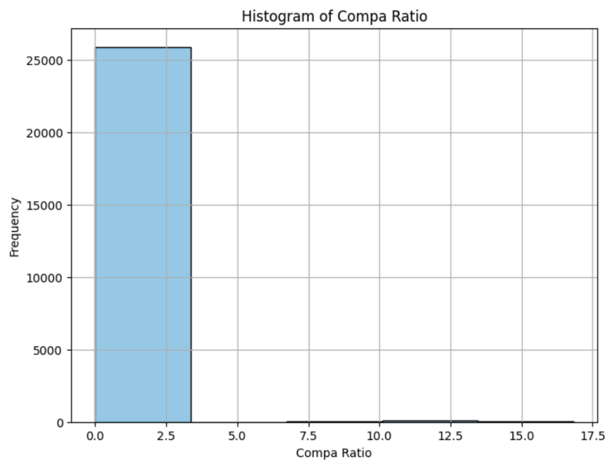


Figure 7: compa ratio before

While this might not seem obvious at first glance, we noticed the very small bars to the right of the large bar, indicating some rather large values in the dataset. Given that the “compa ratio” is calculated as the employee’s current salary divided by the current market rate, it does not make sense that some values should be as high as 7.5 and above. A value of 1 (100%) means the employee is making what the industry average is, so having a value of 7.5 would mean that an employee is said to be making 750% of the industry average. We assumed these values were incorrect due to a change in the frequency of payment, and therefore filtered out any rows with compa ratios of over 2.0 and below .25. This cleaning process has left us with a much more accurate and normalized depiction of the spread of values of “compa ratio”.

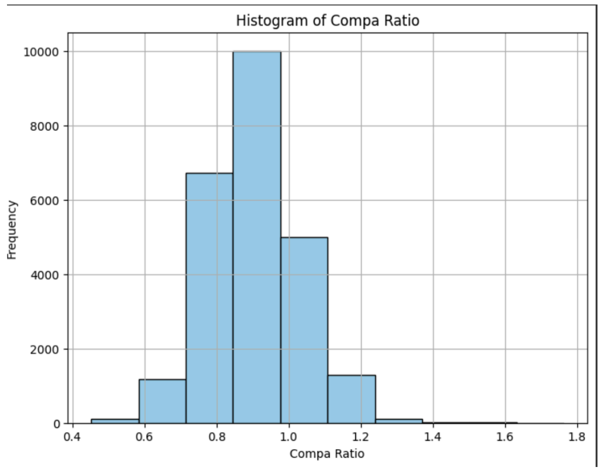
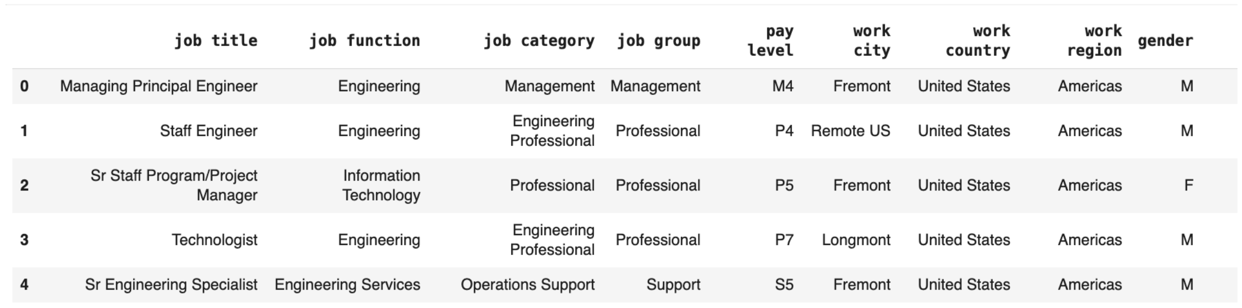


Figure 8: compa ratio after

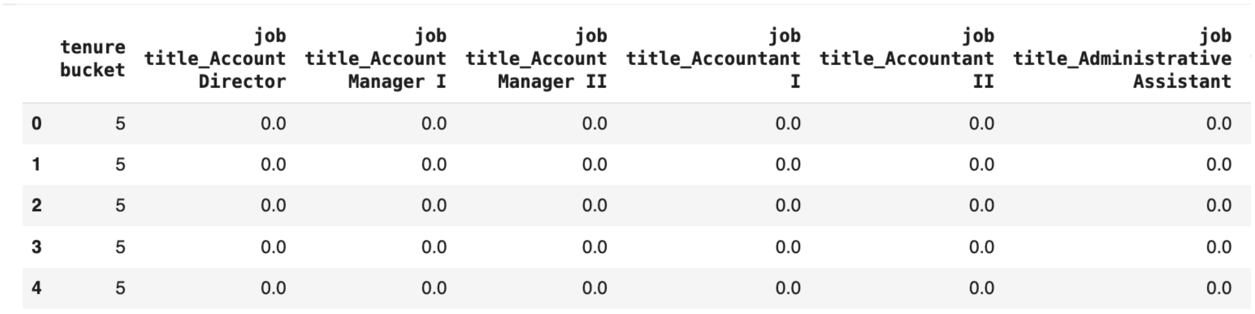
Step 7: Encoding of Categorical Features

We used the one-hot encoding feature from the pandas library. This process converts categorical variables into binary vectors, with each category represented by a separate binary column. This transformation is important because many machine learning algorithms can only operate on numerical data, so categorical variables need to be converted into a numerical format.



	job title	job function	job category	job group	pay level	work city	work country	work region	gender
0	Managing Principal Engineer	Engineering	Management	Management	M4	Fremont	United States	Americas	M
1	Staff Engineer	Engineering	Engineering Professional	Professional	P4	Remote US	United States	Americas	M
2	Sr Staff Program/Project Manager	Information Technology	Professional	Professional	P5	Fremont	United States	Americas	F
3	Technologist	Engineering	Engineering Professional	Professional	P7	Longmont	United States	Americas	M
4	Sr Engineering Specialist	Engineering Services	Operations Support	Support	S5	Fremont	United States	Americas	M

Figure 9: Dataframe before categorical one hot encoding



	tenure bucket	job title_Account Director	job title_Account Manager I	job title_Account Manager II	job title_Accountant I	job title_Accountant II	job title_Administrative Assistant
0	5	0.0	0.0	0.0	0.0	0.0	0.0
1	5	0.0	0.0	0.0	0.0	0.0	0.0
2	5	0.0	0.0	0.0	0.0	0.0	0.0
3	5	0.0	0.0	0.0	0.0	0.0	0.0
4	5	0.0	0.0	0.0	0.0	0.0	0.0

Figure 9.1: Dataframe after categorical one hot encoding. A 1 will be present in place of a 0 to indicate a true response.

Step 8: Recursive Feature Engineering (RFE)

RFE is a feature engineering technique used commonly in machine learning when datasets have many features, such as ours did after we encoded them. We used RFE to determine the top 100 most relevant features for one of our high-performing models and used those 100 features as our X variables for the remaining models. After completing this step, we saw increases in our model values. We did remove six other features upon testing with active employees and realizing certain features had no active employees within them, bringing the final feature number to 94.

Step 9: Dataframe Creation

To create a useable dataset for training and testing, we filtered out rows that included employees who involuntarily churned, or those with termination types of ‘Retired’, ‘Retirement (Rehired)’, ‘Deceased’, ‘Involuntary Termination’, ‘Other Termination’, and ‘ Release Termination’, leaving us with a data frame of voluntarily churned and active employees.

Step 10: Converting to CSV

Lastly, we converted the Excel file to CSV format to streamline our workflow and enhance efficiency when importing the data into Python. By utilizing CSV, we leverage its simplicity and reduced file size, resulting in faster loading times and improved processing speed.

This conversion enables smoother integration with our model, facilitating a more seamless and timely analysis of the data.

Splitting the Data

After preprocessing, feature engineering, and RFE we used a total of 94 columns and 17,528 instances for training and testing data sets.

The dataset was randomly split into training and testing sets, using sklearn's train_test_split, ensuring that the classes are balanced between the two sets, shuffling the data, and setting a random seed for reproducibility. We did not use a holdout set, as we did not feel it was necessary for this project.

Splitting Percentages

We chose to adhere to industry standards when dividing the data into training and validation sets. Training sets typically have more data than validation sets because training is much more comprehensive and requires more data for the model to learn and understand the data.

- Training Set: 80%
    - Number of training instances: 14,022
- Testing Set: 20%
    - Number of instances: 3,506

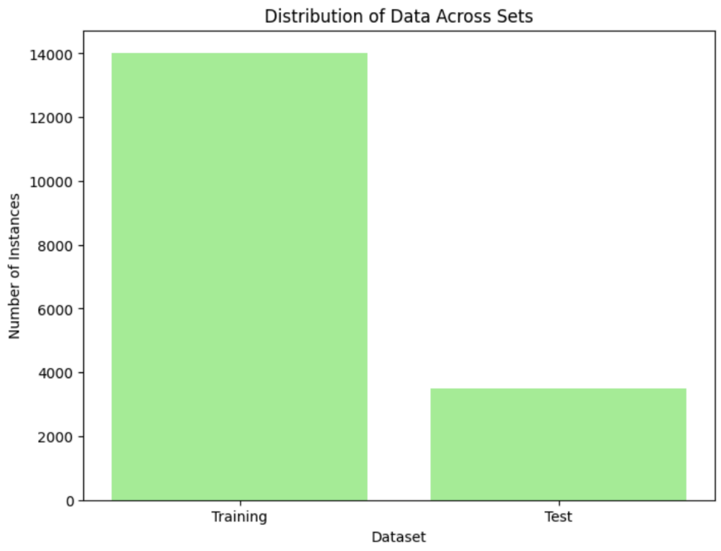


Figure 10: Distribution of data across training and testing sets

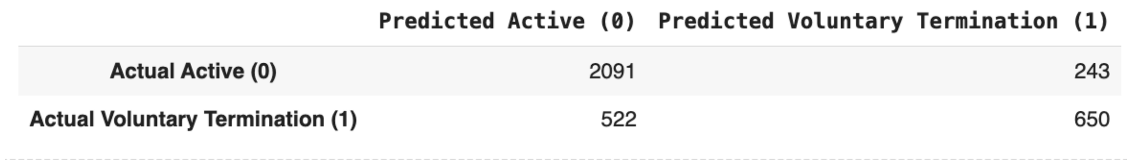
Data Cleaning Conclusion

In conclusion, our data refinement journey was a meticulous and systematic endeavor aimed at enhancing the quality, consistency, and usability of our dataset. Beginning with basic exploratory analysis, we gained valuable insights into the dataset's structure and characteristics, laying a solid foundation for subsequent cleaning and processing steps. By refining column names, handling missing data, and conducting feature engineering, we ensured that our dataset is well-prepared for in-depth analysis and modeling tasks. Additionally, our attention to detail during data sanity checks allowed us to identify and rectify anomalies, ensuring data accuracy and reliability. Lastly, we created ideal data frames for use in models we plan to make in the future. Through these efforts, we have transformed our dataset into a cleaner, more structured, and actionable resource, poised to support informed decision-making and drive meaningful insights for [redacted] HR team.

Models Used

Gradient Boosting Classifier (model 0)

- ➔ We attempted to implement the Gradient Boosting Classifier (GBM), a powerful ensemble learning method used for classification tasks. GBM sequentially builds decision trees, each correcting errors made by the previous trees, and optimizes a loss function using gradient descent to improve prediction accuracy. This approach often yields high performance and robustness to noisy data, making GBM a popular choice for classification tasks.
- ➔ Accuracy: About 78% of the predictions made by this model are correct.
- ➔ F1-score: The overall accuracy of this model, considering both precision and recall, is around 63%.
- ➔ Recall: This model correctly identifies about 55% of all actual positive instances.
- ➔ AUC: The model's ability to distinguish between positive and negative classes is approximately 72%.



	Predicted Active (0)	Predicted Voluntary Termination (1)
Actual Active (0)	2091	243
Actual Voluntary Termination (1)	522	650

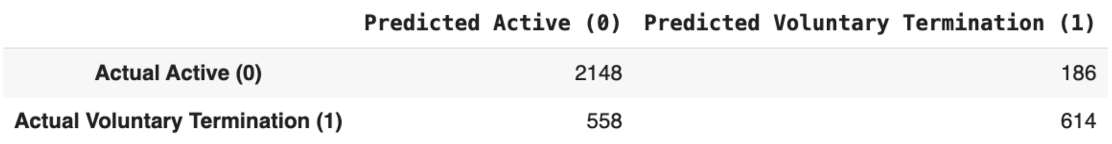
Figure 11: Confusion matrix for GBM model

- ➔ The Gradient Boosting Classifier achieves moderate performance across all metrics, with a relatively high accuracy and AUC. It effectively balances precision and recall. Overall, this model has good performance but the low recall is something that cannot be ignored.

Support Vector Machine (model 1)

- ➔ We also attempted a Support Vector Classifier (SVC) model, a popular algorithm for classification tasks. SVC aims to find the hyperplane that best separates the classes in the feature space, maximizing the margin between them. It is effective in handling high-dimensional data and is robust to overfitting.

- ➔ Accuracy: About 79% of the predictions made by this model are correct.
- ➔ F1-score: The overall accuracy of this model, considering both precision and recall, is around 62%.
- ➔ Recall: This model correctly identifies about 52% of all actual positive instances.
- ➔ AUC: The model's ability to distinguish between positive and negative classes is approximately 72%.



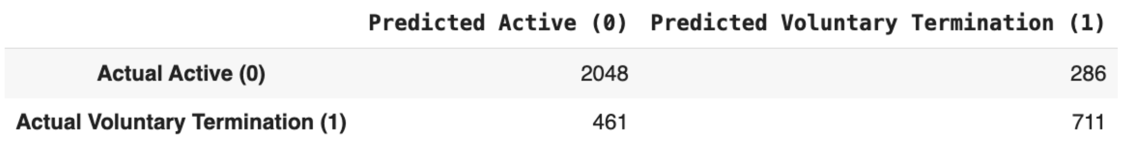
	Predicted Active (0)	Predicted Voluntary Termination (1)
Actual Active (0)	2148	186
Actual Voluntary Termination (1)	558	614

Figure 12: Confusion matrix for SVM model

- ➔ The Support Vector Classifier performs similarly to the Gradient Boosting Classifier, with slightly lower recall but comparable accuracy and AUC. Again, this model has a recall that is lower than desired and therefore isn't a good option for our task.

Random Forest Classifier (model 2)

- ➔ We also explored the Random Forest Classifier, a widely used ensemble learning technique for classification tasks. Random Forest Classifier constructs multiple decision trees during training, each contributing to the final prediction through a voting mechanism. This approach helps mitigate overfitting and improves generalization. Random Forest is particularly effective in handling high-dimensional data and capturing complex relationships within the dataset.
- ➔ Accuracy: Around 79% of the predictions made by this model are correct.
- ➔ F1-score: The overall accuracy of this model, considering both precision and recall, is approximately 66%.
- ➔ Recall: This model correctly identifies about 79% of all actual positive instances.
- ➔ AUC: The model's ability to distinguish between positive and negative classes is roughly 83%.



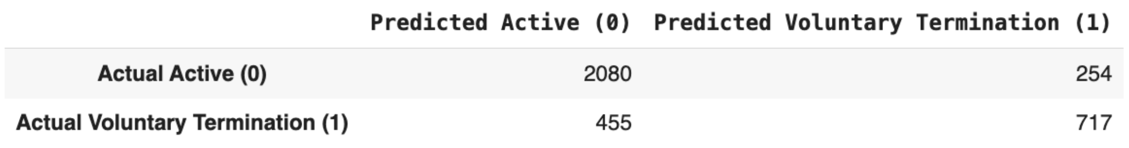
	Predicted Active (0)	Predicted Voluntary Termination (1)
Actual Active (0)	2048	286
Actual Voluntary Termination (1)	461	711

Figure 13: Confusion matrix for random forest classifier model

- ➔ The Random Forest Classifier achieves the highest recall among the models, indicating its strength in correctly identifying positive instances. It also demonstrates good overall performance with high accuracy and AUC.

Extreme Gradient Boosting Classifier (model 3)

- ➔ We employed the XGBoost Classifier, a highly efficient and advanced gradient boosting method used for classification tasks. XGBoost builds on the concept of boosting decision trees sequentially, each correcting errors made by the previous one, and uses a gradient descent optimization algorithm to minimize loss, enhancing predictive accuracy. This method is favored for its robustness and superior handling of varied data types, making it an excellent choice for complex classification challenges.
- ➔ Accuracy: About 80% of the predictions made by this model are correct.
- ➔ F1-score: The overall accuracy of this model, considering both precision and recall, is approximately 67%.
- ➔ Recall: This model correctly identifies about 61% of all actual positive instances.
- ➔ AUC: The model's ability to distinguish between positive and negative classes is roughly 85%.



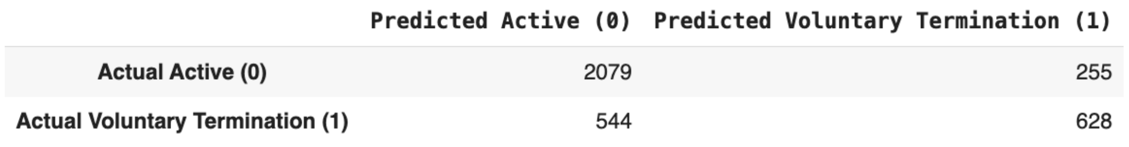
	Predicted Active (0)	Predicted Voluntary Termination (1)
Actual Active (0)	2080	254
Actual Voluntary Termination (1)	455	717

Figure 14: Confusion matrix for XGB model

- ➔ The XGBoost Classifier outperforms other models in terms of accuracy, F1-score, and AUC. It strikes a good balance between precision and recall, with a particularly high AUC indicating its effectiveness in ranking instances.

Logistic Regression (model 4)

- ➔ We also implemented Logistic Regression, a fundamental algorithm for binary classification tasks. Logistic Regression aims to estimate the probability that a given instance belongs to a particular class by fitting a logistic function to the input features. Despite its simplicity, Logistic Regression is robust, interpretable, and suitable for scenarios with linearly separable data.
- ➔ Accuracy: Around 77% of the predictions made by this model are correct.
- ➔ F1-score: The overall accuracy of this model, considering both precision and recall, is approximately 61%.
- ➔ Recall: This model correctly identifies about 54% of all actual positive instances.
- ➔ AUC: The model's ability to distinguish between positive and negative classes is roughly 71%.



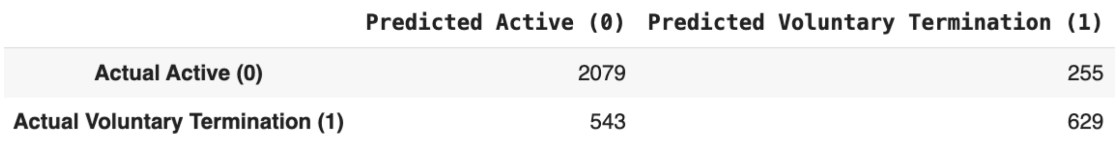
	Predicted Active (0)	Predicted Voluntary Termination (1)
Actual Active (0)	2079	255
Actual Voluntary Termination (1)	544	628

Figure 15: Confusion matrix for logistic regression model 4

- ➔ The first instance of Logistic Regression shows moderate performance across all metrics, similar to other models.

Logistic Regression (model 5)

- ➔ Accuracy: About 77% of the predictions made by this model are correct.
- ➔ F1-score: The overall accuracy of this model, considering both precision and recall, is approximately 61%.
- ➔ Recall: This model correctly identifies about 54% of all actual positive instances.
- ➔ AUC: The model's ability to distinguish between positive and negative classes is roughly 71%.

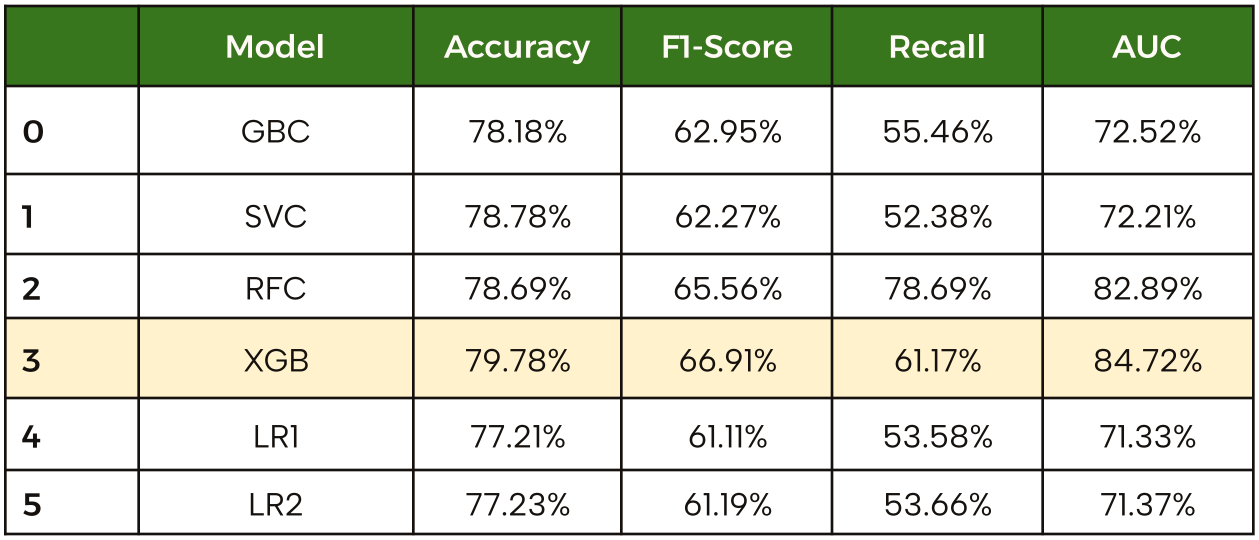


	Predicted Active (0)	Predicted Voluntary Termination (1)
Actual Active (0)	2079	255
Actual Voluntary Termination (1)	543	629

Figure 16: Confusion matrix for logistic regression model 4

- ➔ The second instance of Logistic Regression performs similarly to the first instance, with consistent performance across metrics.

Model Comparisons



	Model	Accuracy	F1-Score	Recall	AUC
0	GBC	78.18%	62.95%	55.46%	72.52%
1	SVC	78.78%	62.27%	52.38%	72.21%
2	RFC	78.69%	65.56%	78.69%	82.89%
3	XGB	79.78%	66.91%	61.17%	84.72%
4	LR1	77.21%	61.11%	53.58%	71.33%
5	LR2	77.23%	61.19%	53.66%	71.37%

Figure 17: Comparison between all models tested

All models were trained using the split data and subsequently utilized to predict the target variable on the test set. Performance metrics such as accuracy, F1-score, recall, and Area Under the ROC Curve (AUC) were computed to evaluate the model's effectiveness in accurately classifying instances. Figure 17 displays all the metrics used in an easy-to-understand table. In evaluating the performance of the model, several metrics were considered. Accuracy was assessed to gauge the overall effectiveness across all classes, indicating the model's ability to make correct predictions. F1 score was utilized as it offers a balanced measure, taking into account both precision and recall, thus providing insight into the model's ability to minimize false positives and false negatives. Recall was examined to understand how well the model identifies actual positive instances among the total positives. Additionally, the Area Under the Curve (AUC) was analyzed to ascertain the model's capability to distinguish between positive and negative classes; a higher AUC value indicates superior discrimination performance. These metrics collectively influenced the decision-making process regarding whether to proceed with each model, reflecting its efficacy and reliability for the intended task.

When comparing the performance of different models based on the provided table, several observations can be made. Firstly, in terms of accuracy, the XGB (Extreme Gradient Boosting) model outperforms the others with an accuracy of 79.78%, closely followed by the SVC (Support Vector Classifier) at 78.78%. However, when considering the F1-score, which balances precision and recall, XGB still leads with 66.91%, but RFC (Random Forest Classifier) closely follows with 65.56%. Moreover, RFC demonstrates the highest recall rate at 78.69%, indicating its effectiveness in capturing positive instances. In terms of the AUC (Area Under the Curve), XGB again exhibits the best performance at 84.72%, showcasing its superior ability to distinguish between positive and negative classes. Overall, while RFC also stands out for its high recall rate, XGB consistently performs well across multiple metrics.

Final Suggestion

Based on our comprehensive evaluation, we recommend the adoption of the Extreme Gradient Boosting (XGBoost) classifier due to its outstanding performance in terms of accuracy, F1-score, and AUC. This model demonstrates superior predictive capability in identifying potential voluntary churn among employees. By leveraging the XGBoost classifier, [redacted] can optimize recruitment strategies and make data-driven decisions to mitigate attrition risks effectively.

The accuracy of the predictions stands at approximately 80%, indicating that the model predicts outcomes correctly in the majority of cases. When considering both precision and recall, the F1-score, reflecting overall accuracy, hovers around 67%, showcasing a balanced performance. Regarding recall, the model accurately identifies approximately 61% of all actual positive instances. Furthermore, the model's AUC, signifying its ability to distinguish between positive and negative classes, reaches around 85%, indicating a strong capability in this aspect.

You can see in figure 18 that the model demonstrates superior capability in predicting and classifying potential voluntary churn, as shown via the actual versus predicted responses, and see the model's abilities.

	Predicted Active	Predictive Voluntary Termination
Actual Active	2080	254
Predicted Voluntary Termination	455	717

Figure 18: Actual versus predicted response counts from the model

You can also see the benefits of adopting the classifier, like optimizing recruitment strategies based on the feature importance, in other words, what things are most likely to predict whether an employee is likely to churn voluntarily. In your case, someone being an engineering specialist II with a compa bucket of 50-75% is very likely to churn voluntarily, as shown in Figure 19.

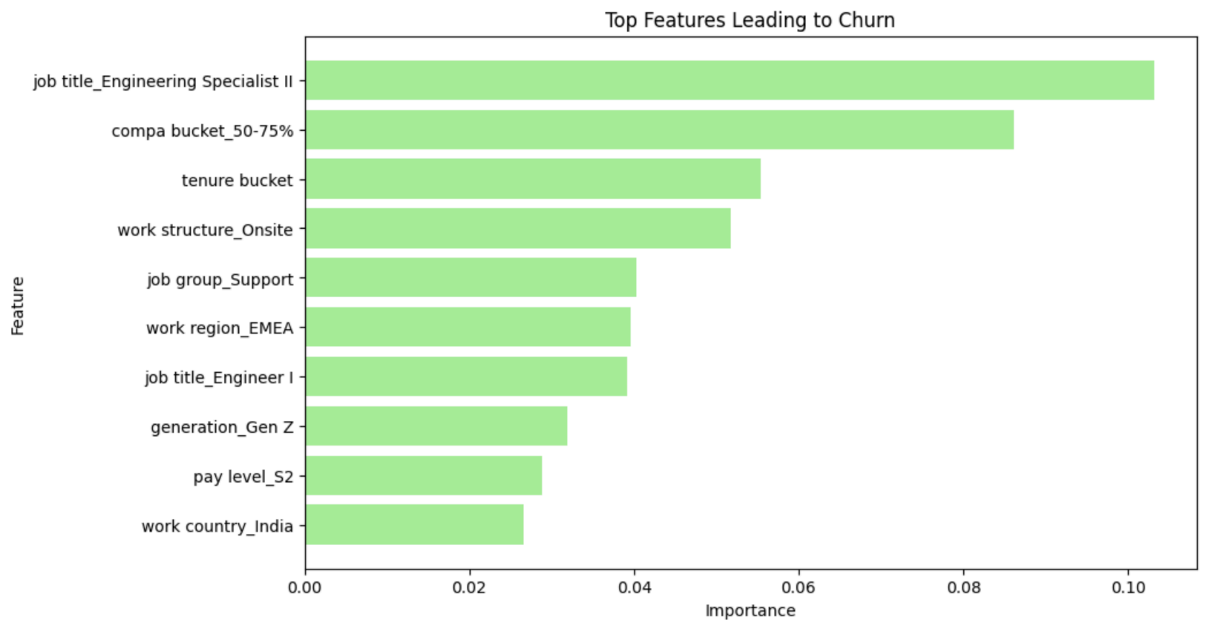


Figure 19: top features leading to voluntary churn

By adopting this model, [redacted] HR team can identify and address the key drivers of churn more proactively, potentially reducing turnover rates and improving employee satisfaction. By delving deeper into the features that lead to churn and requesting feedback from employees who fall under those features, [redacted] can potentially address the reasons that employees in these sectors are churning. Additionally, the insights gained from this model will assist in optimizing recruitment strategies to fill the talent pipeline effectively, aligning with the forecasted human capital needs over the next two years.

Two Year Pipeline

In the presented data in figure 20, we observe the projected number of employees predicted to voluntarily churn from the current active workforce. Additionally, we have compiled a data frame containing all active employees alongside their respective probabilities of churning. Employing a threshold of 70% or higher predicted probability, individuals are classified as potential churners. These numbers were calculated by running a dataframe of only active employees through our model.

	Likely to Churn
False	11,431
True	237

Anon ID	Churn Probability	Likely to Churn
1113	02.32%	False
1115	13.43%	False
1116	21.88%	False
1120	01.47%	False

Figure 20: Count of employees likely to churn and chart showing churn probability by employee anon ID

When considering [redacted] annual headcount dynamics, we found that the organization experiences an average attrition rate of 5.27%, encompassing all forms of terminations. To sustain a consistent headcount each year, it is recommended that [redacted] recruits an additional workforce equivalent to 6% of their current annual headcount. This figure accommodates potential losses incurred from newly hired employees within the same period. It's noteworthy that this recommendation assumes a non-weighted average; employing a weighted average calculation would result in a higher percentage for recruitment needs.

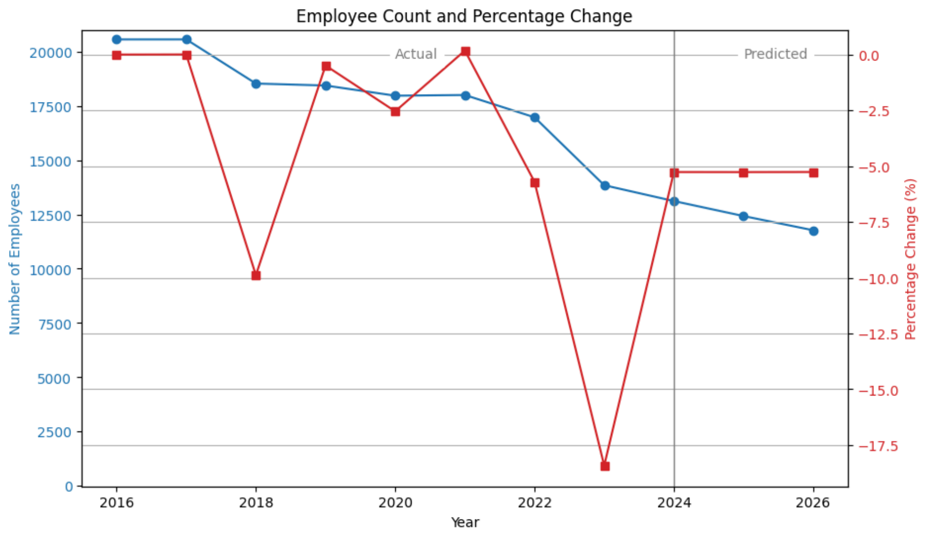


Figure 21: two-year pipeline shown in graph form

Conclusion

In conclusion, the integration of predictive analytics and machine learning techniques into [redacted] HR data analysis procedures marks a transformative endeavor poised to elevate efficiency, precision, and strategic decision-making. Through meticulous data refinement processes encompassing basic analysis, column name refinement, feature engineering, and model evaluation, the CU team has crafted a robust dataset primed for actionable insights. By leveraging our Extreme Gradient Boosting Classifier (XGBoost), [redacted] stands to gain significant advantages in identifying and mitigating employee churn risks while optimizing recruitment strategies for future staffing needs. The adoption of such methodologies would not only enhance HR's operational capabilities but also underscores [redacted] commitment to data-driven decision-making and proactive workforce planning. Moving forward, the continuous refinement of these processes will be instrumental in extracting maximum value from HR data assets and driving sustainable organizational growth.