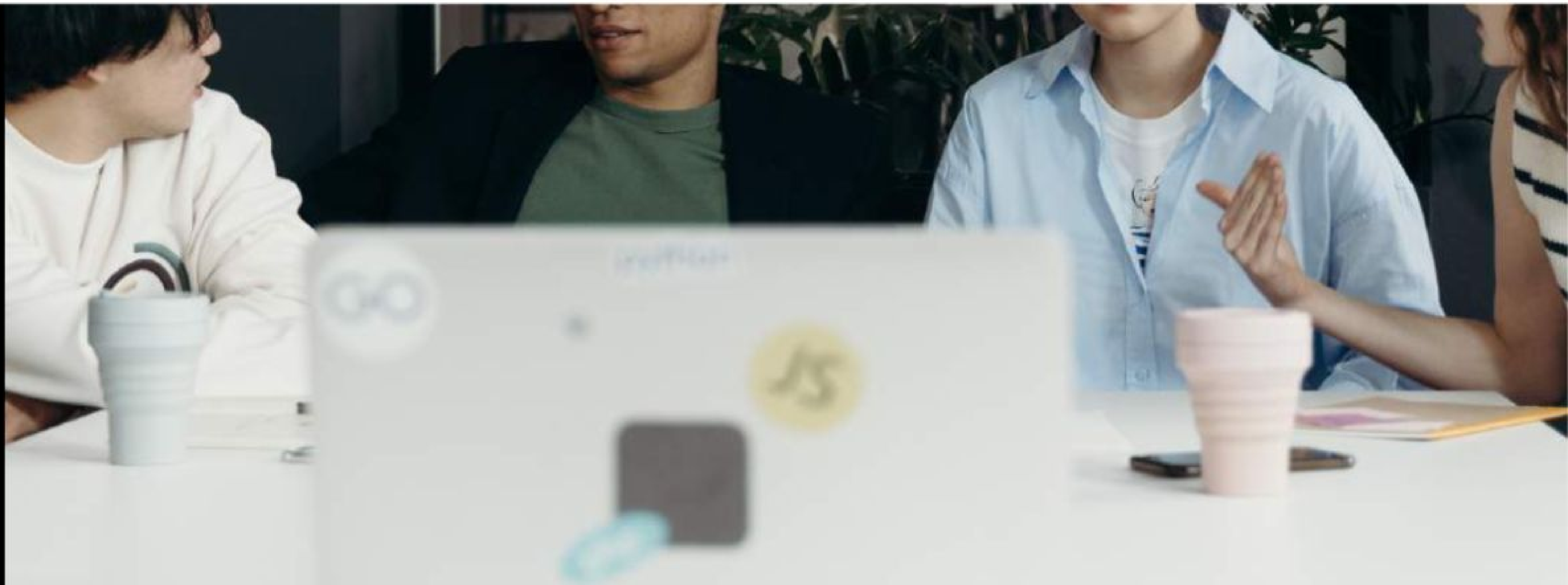
SALIFORT MOTORS

Predicting Employee Leaves

Human Resource Department



by : Ana Farida

Project HR

The HR department at Salifort Motors is focused on improving employee satisfaction, detecting employees who are likely to leave, and understanding the factors behind their decisions.

With better employee retention, Salifort Motors can save time and resources spent on hiring and benefit from a more stable workforce.

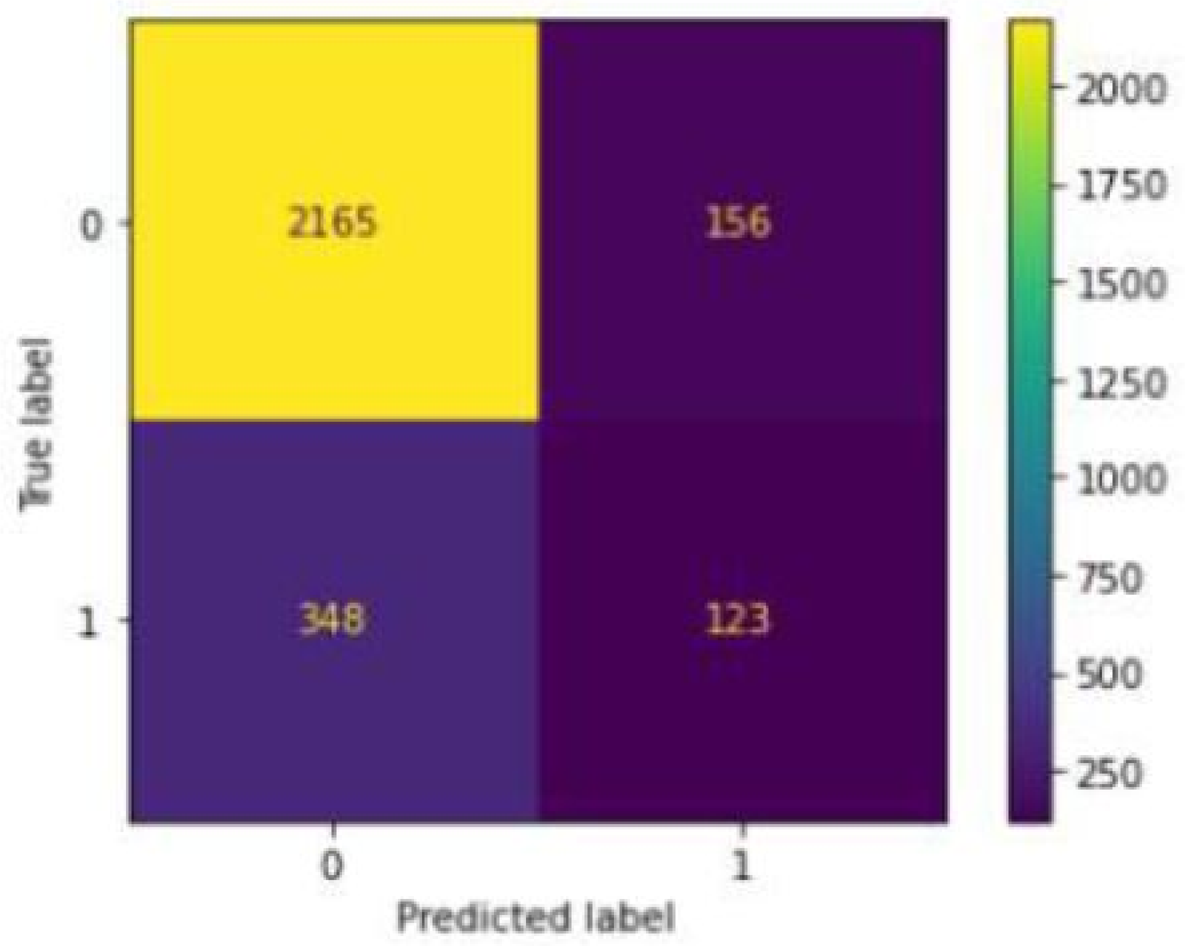
01 Predicting Employee Quits

02 Key Predictors of Employee Quits

03 Insights and Recommendations

Predicting Employee Quits

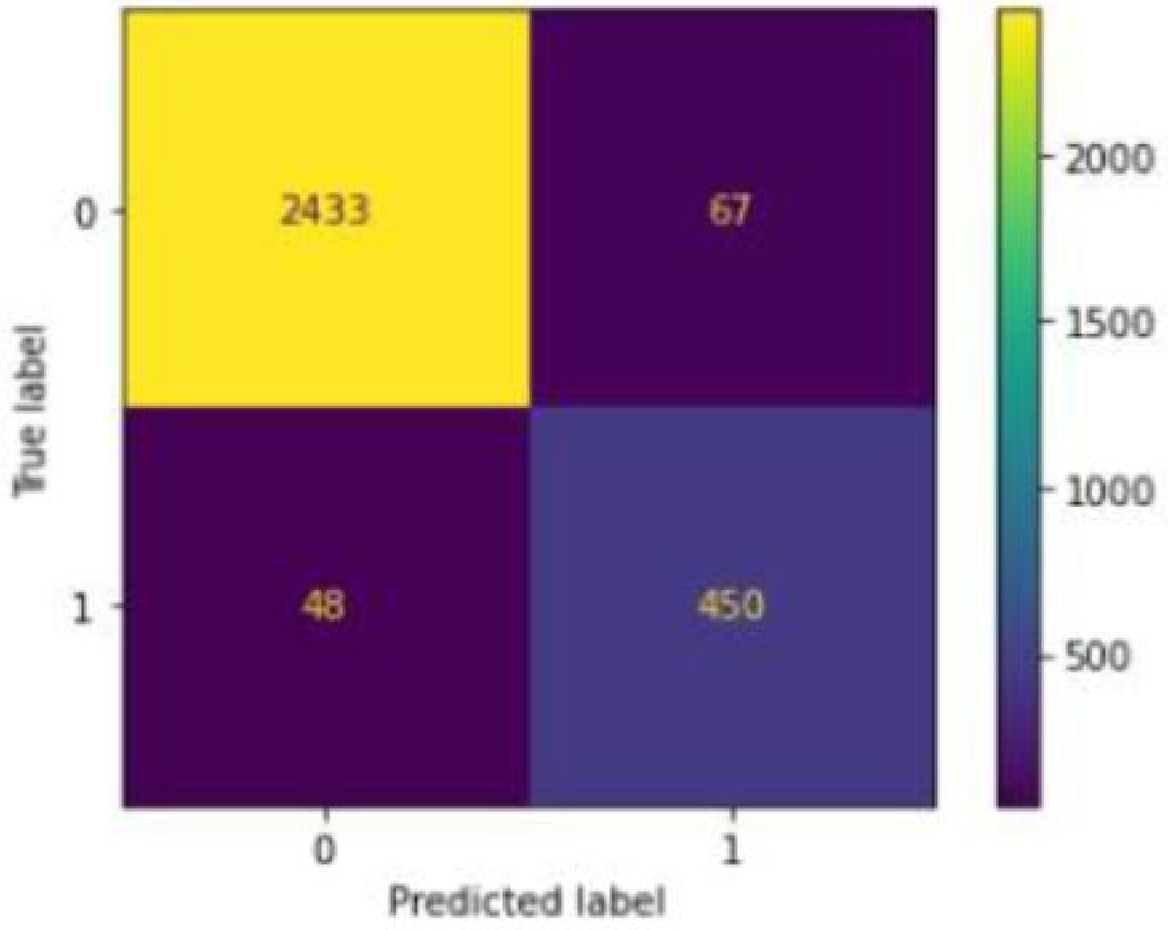
A. Logistic Regression Model



	precision	recall	f1-score	support
weighted avg	0.79	0.82	0.80	2792

- The logistic regression model, trained with a 75% training and 25% testing split, a random state of 42, and a maximum of 500 iterations, achieved a precision of 79%, a recall of 82%, an F1-score of 80% (all weighted averages), and an accuracy of 82%.
- However, the scores are significantly lower when focusing on predicting employees who leave.

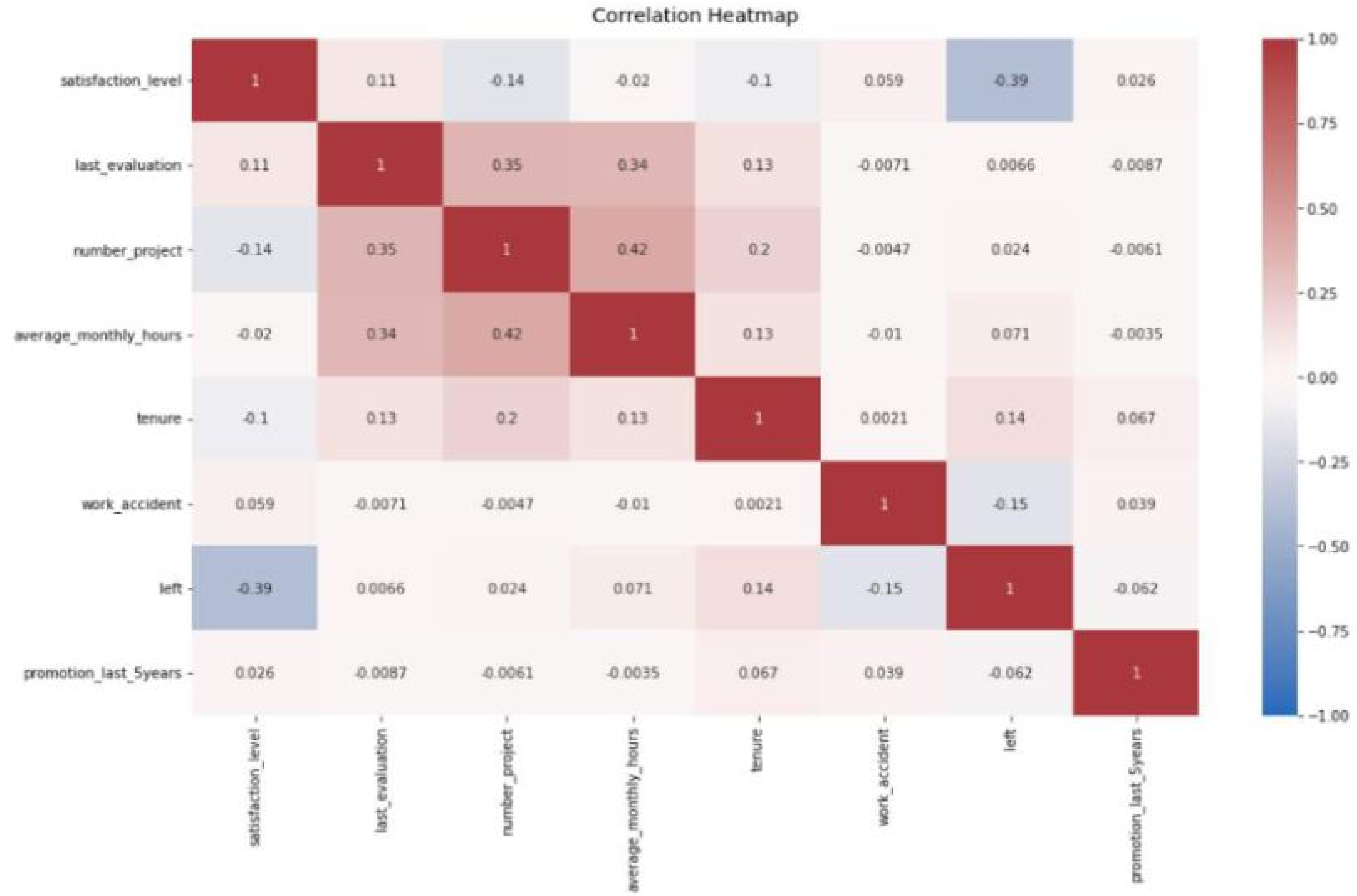
B. Random Forest Model (with Feature Engineering)

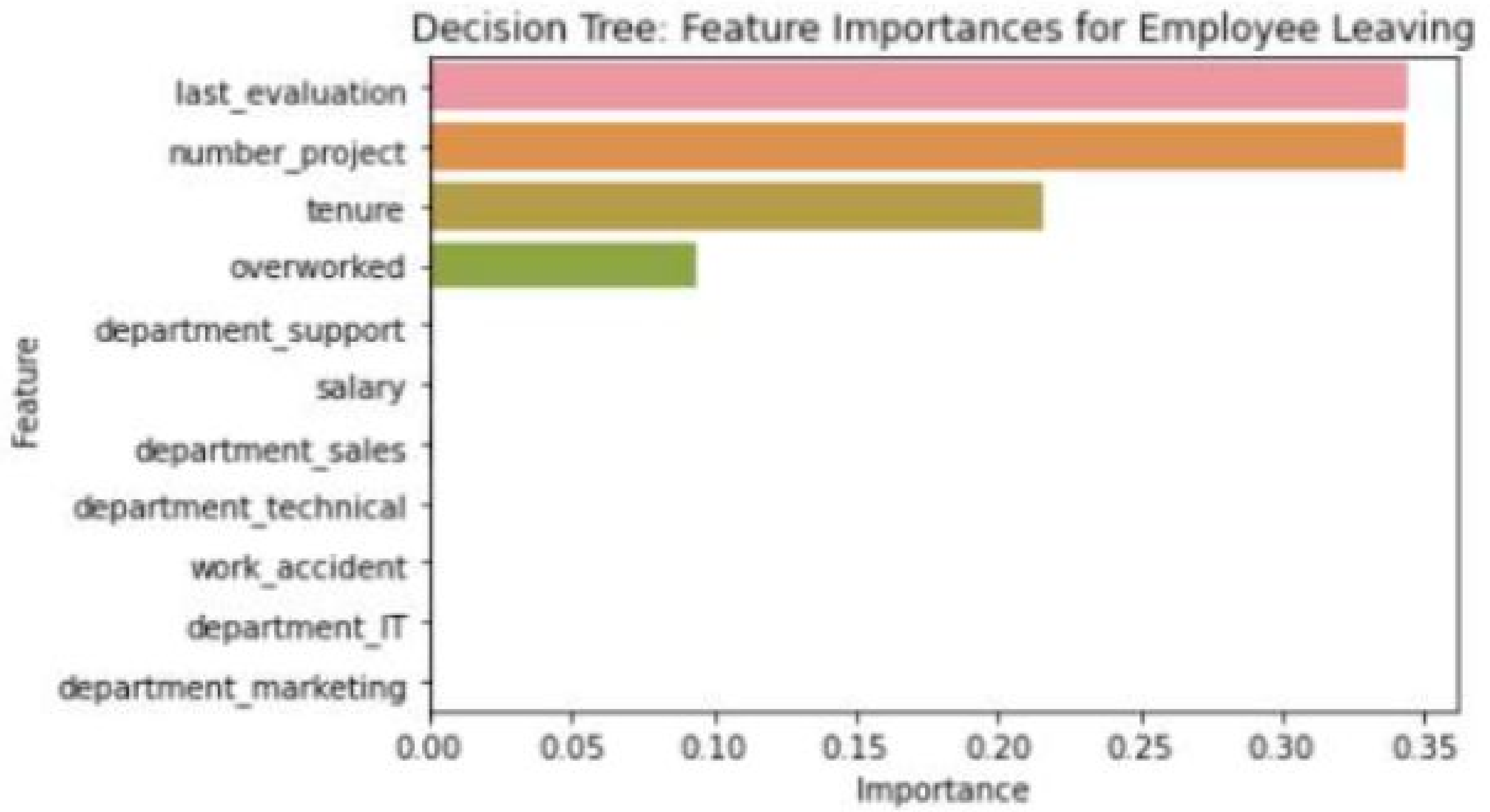


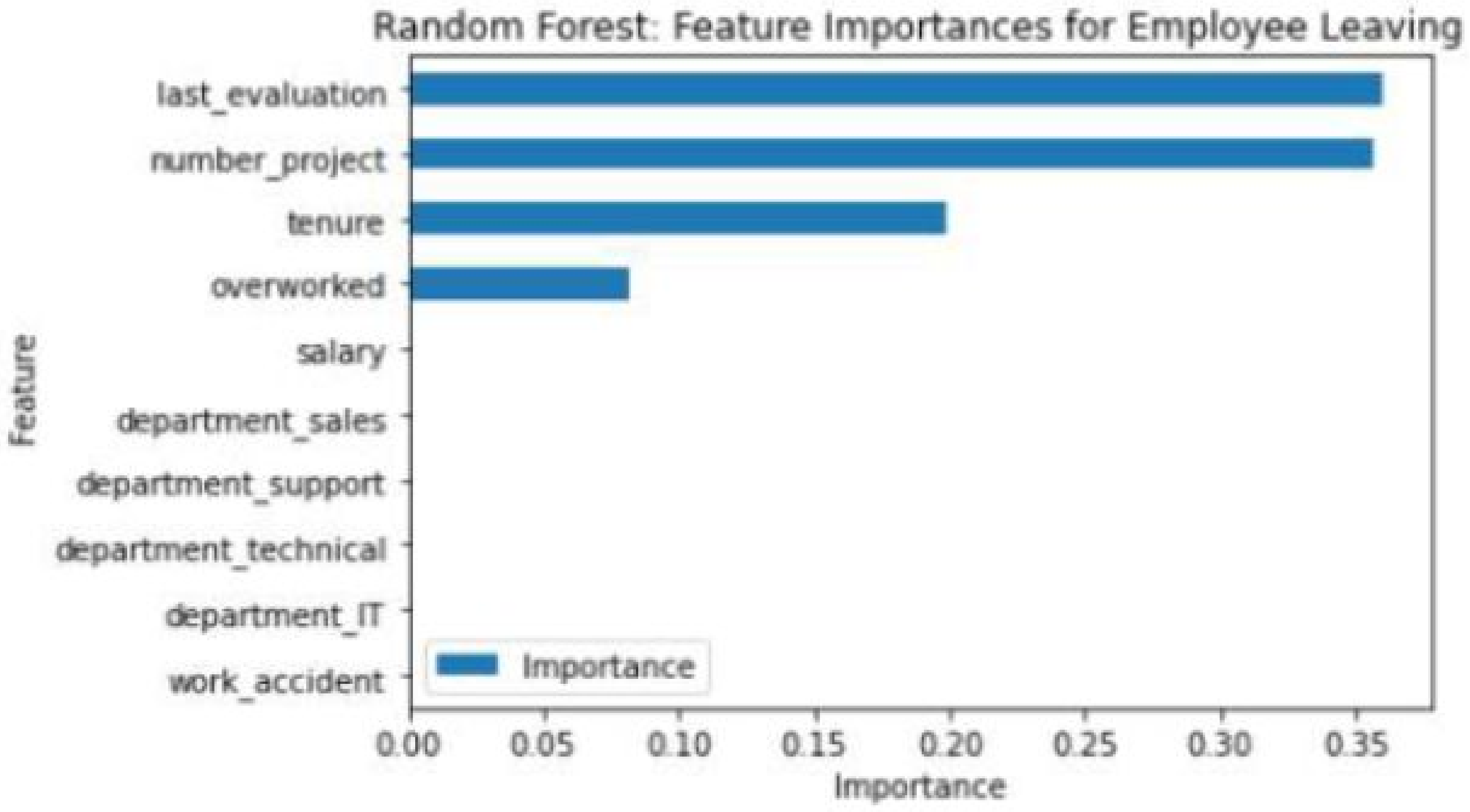
model	precision	recall	F1	accuracy	auc
decision tree2 cv	0.856693	0.903553	0.878882	0.958523	0.958675
model	precision	recall	F1	accuracy	auc
random forest2 cv	0.866758	0.878754	0.872407	0.957411	0.96481

- The random forest2 model was trained using 75% of the data for training and 25% for testing, with cross-validation set to 4, a random state of 0, a maximum depth of 5, maximum features set to 1.0, maximum samples set to 0.7, minimum samples per leaf set to 2, minimum samples per split set to 2, and 300 estimators.
- **Random forest2 achieved an AUC score of 0.964, slightly lower than forest's 0.982 due to using fewer features, but it still outperforms decision tree2 (0.957) based on the AUC metric.**
- The model predicts more false positives than false negatives, which means that some employees may be identified as at risk of quitting or getting fired, when that's actually not the case. But this is still a strong model.

Key Predictors of Employee Quits

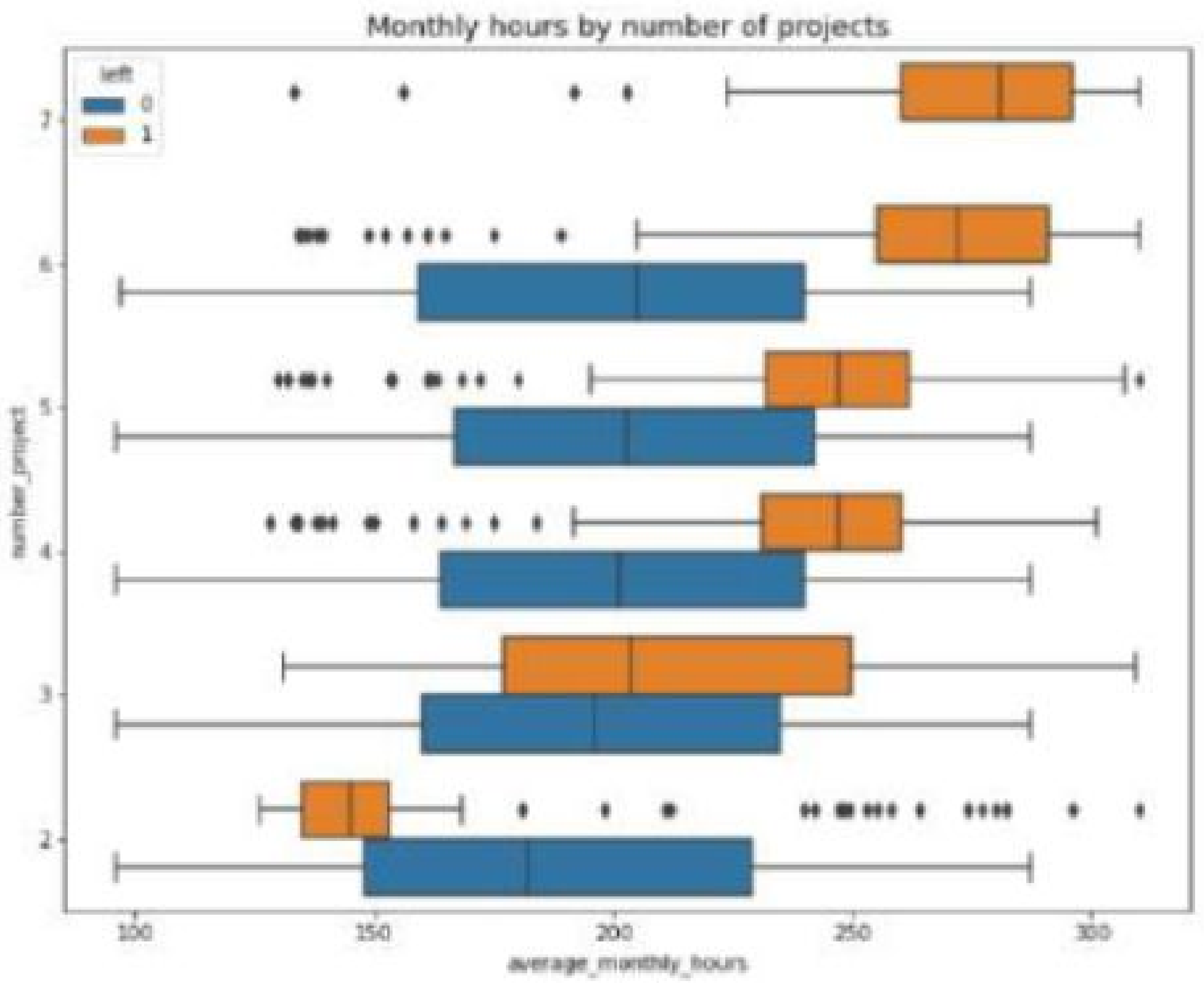


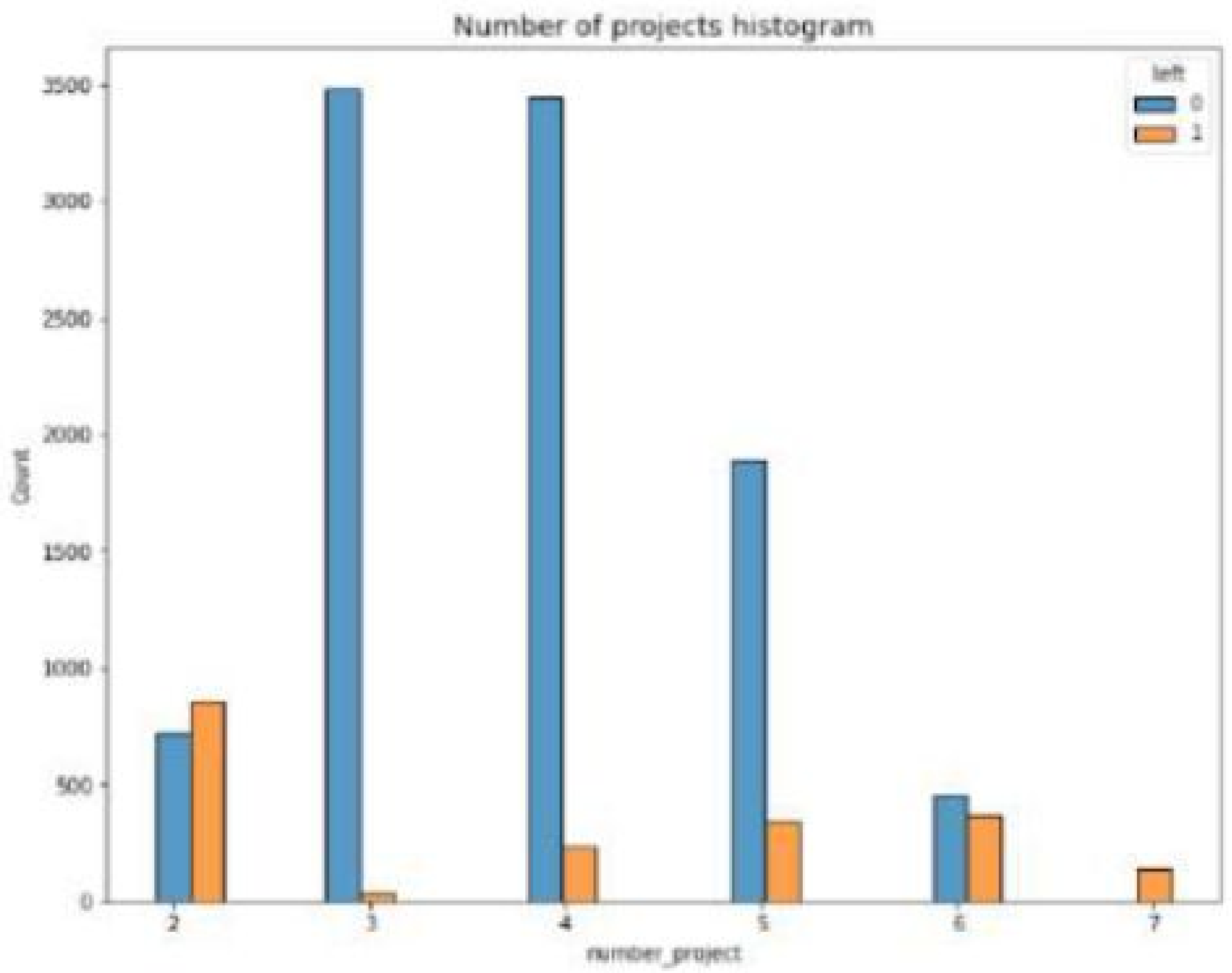




- The correlation heatmap shows that tenure, average monthly hours, number of projects, and last evaluation are related to left, whether an employee leaves.
- Feature importance from both decision tree and random forest models confirms that the **key predictors are last evaluation, number of projects, tenure, and being overworked-working more than 166.67 hours per month.** This is also in line with the results obtained from the correlation heatmap during exploratory data analysis

Exploratory Data Analysis - A



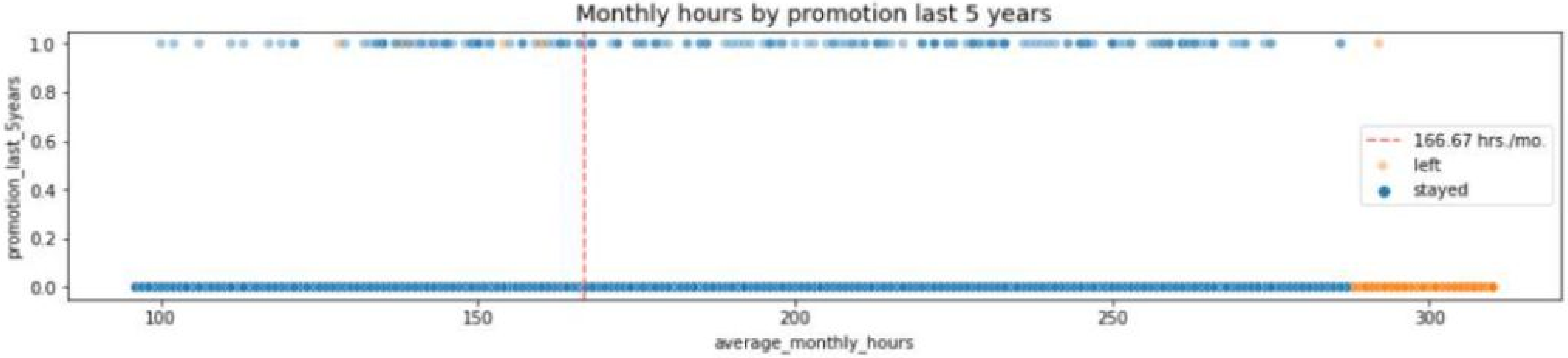


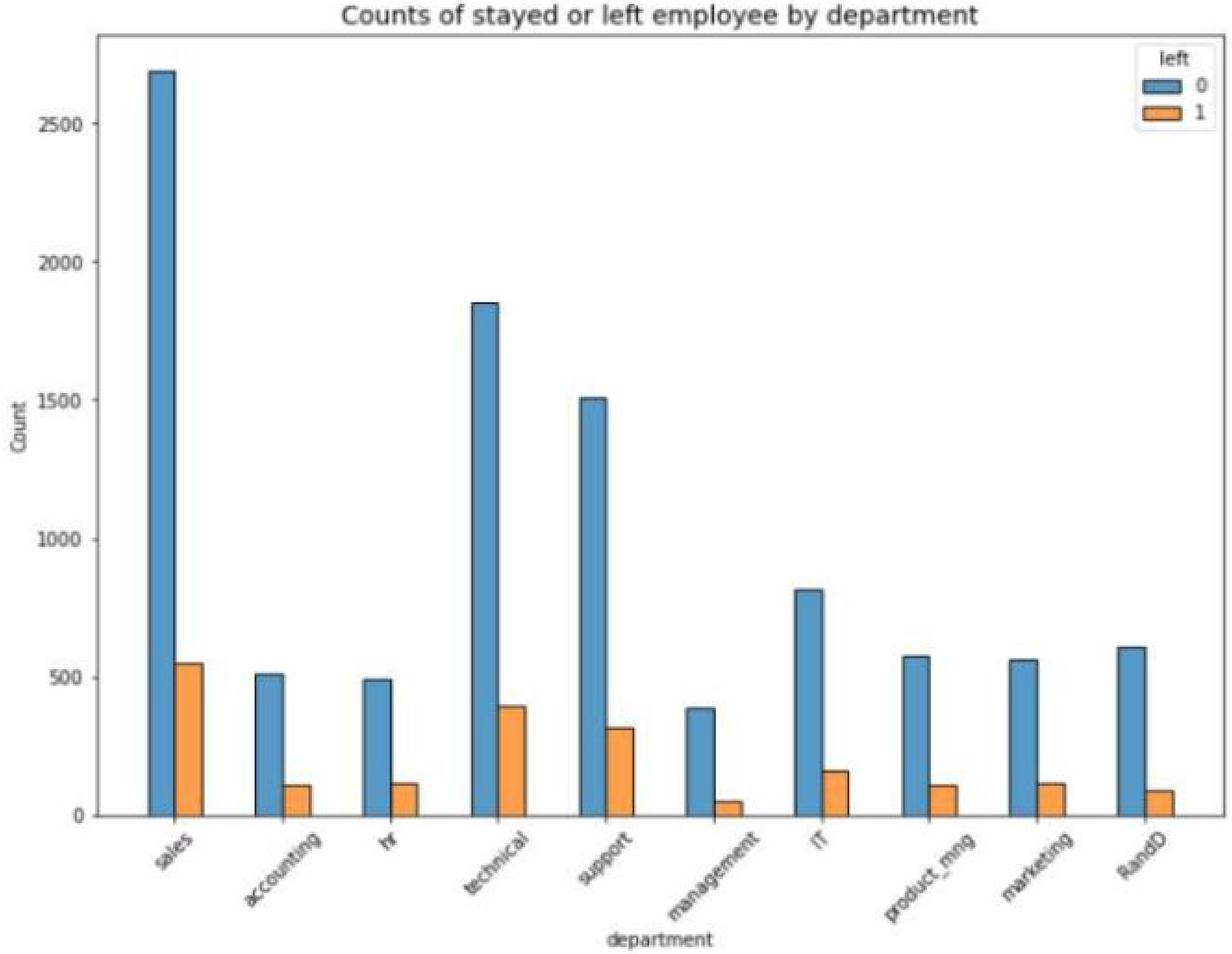
- Assuming a 40-hour workweek and two weeks of vacation a year, employees who work Monday through Friday average 166.67 hours per month, calculated on 50 weeks × 40 hours ÷ 12 months. **Employees who leave the company can be grouped as**:

- **High last_evaluation**: Employees in this category work **more, over 200 hours a month, and handle more than 3 projects.**

- **Low last_evaluation:** Employees with **less hours than the average** of 166.67 hours per month

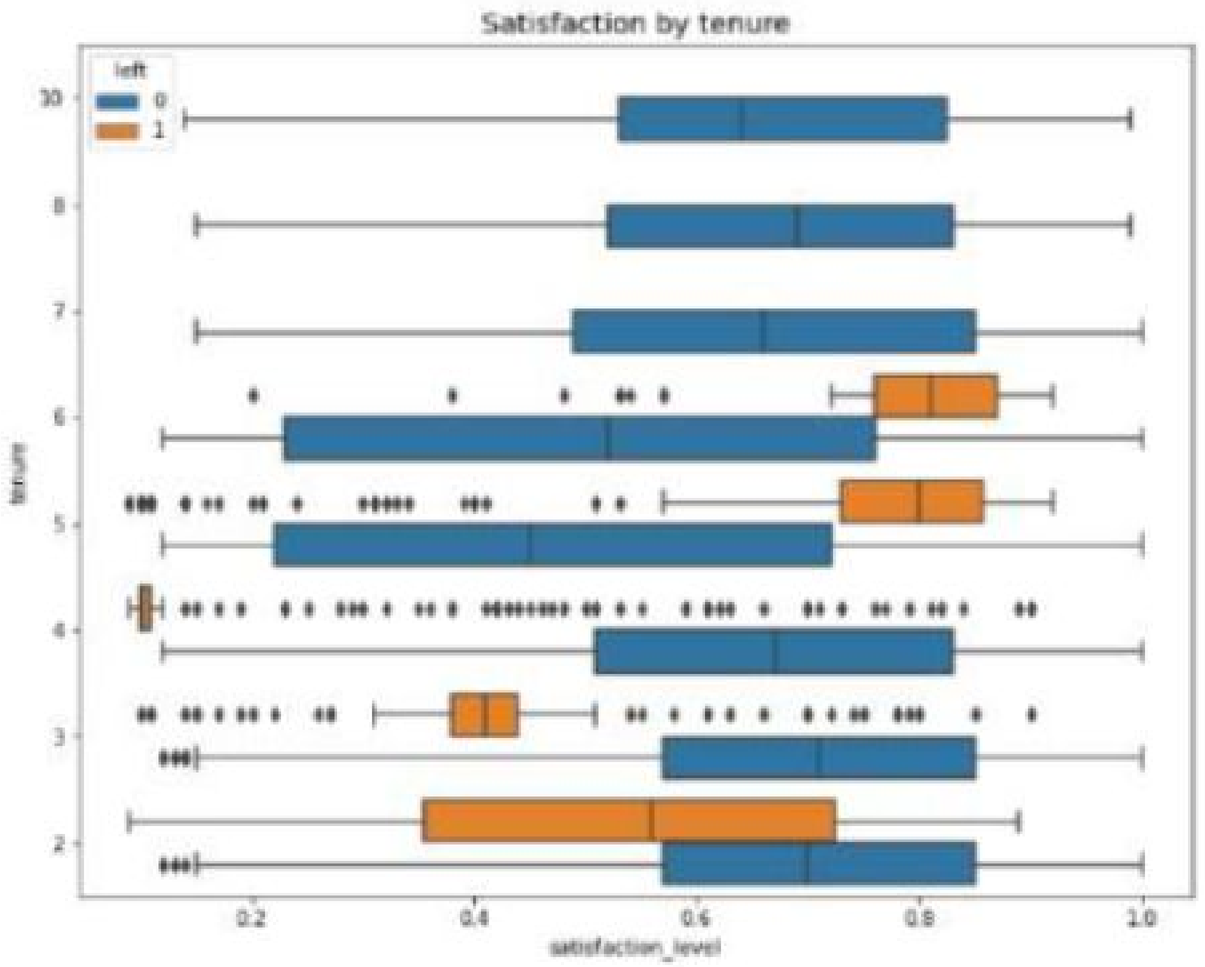
Exploratory Data Analysis - B

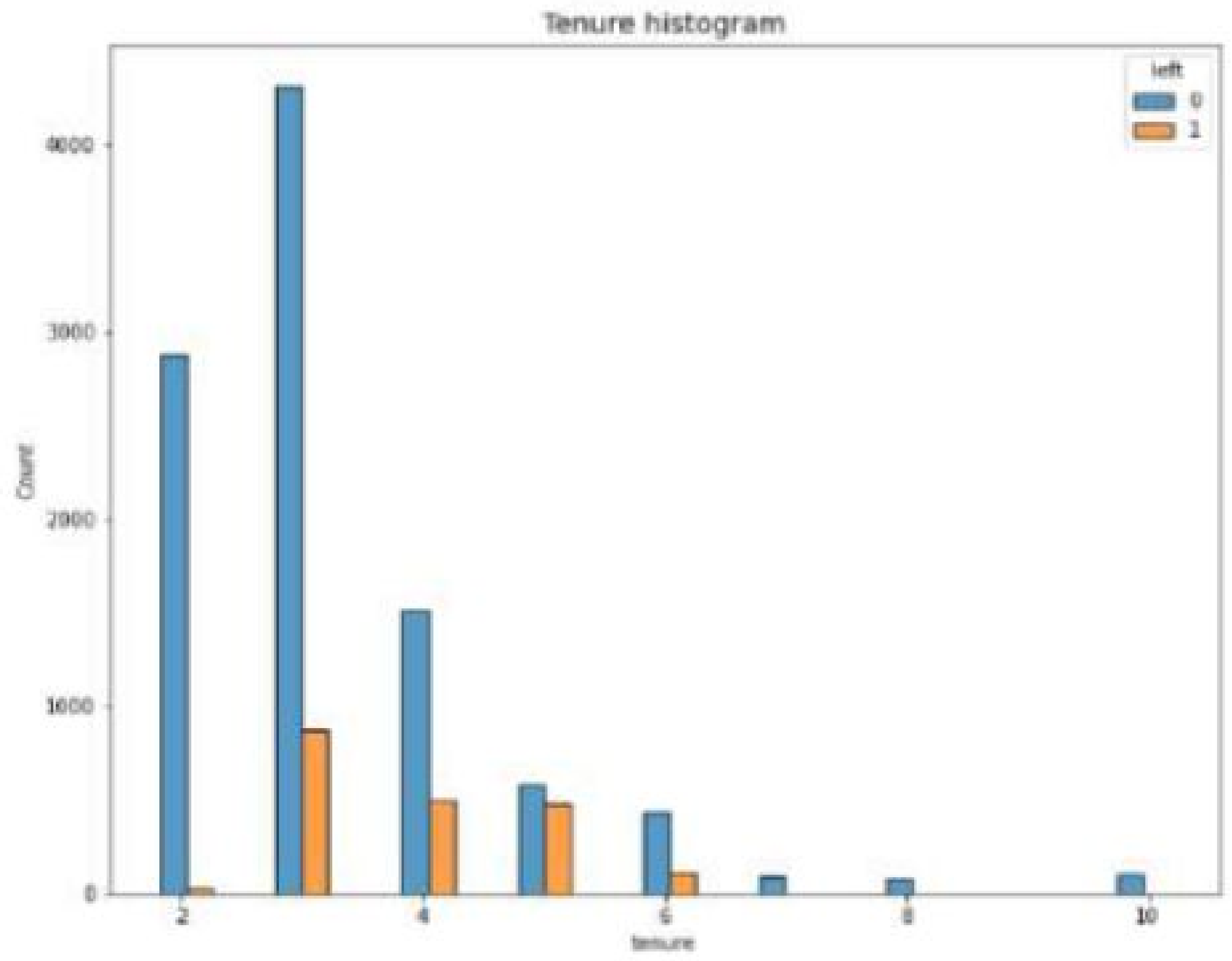


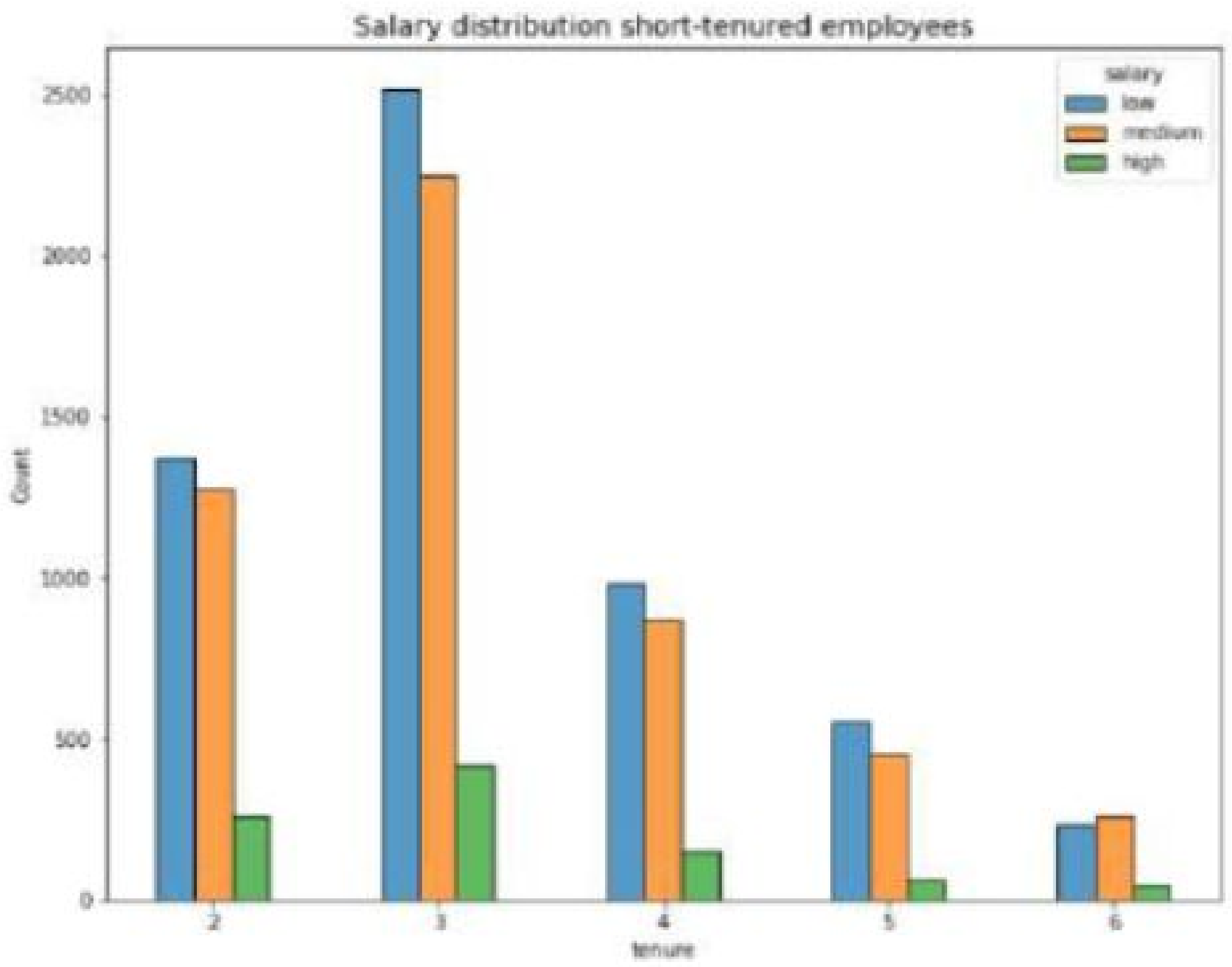


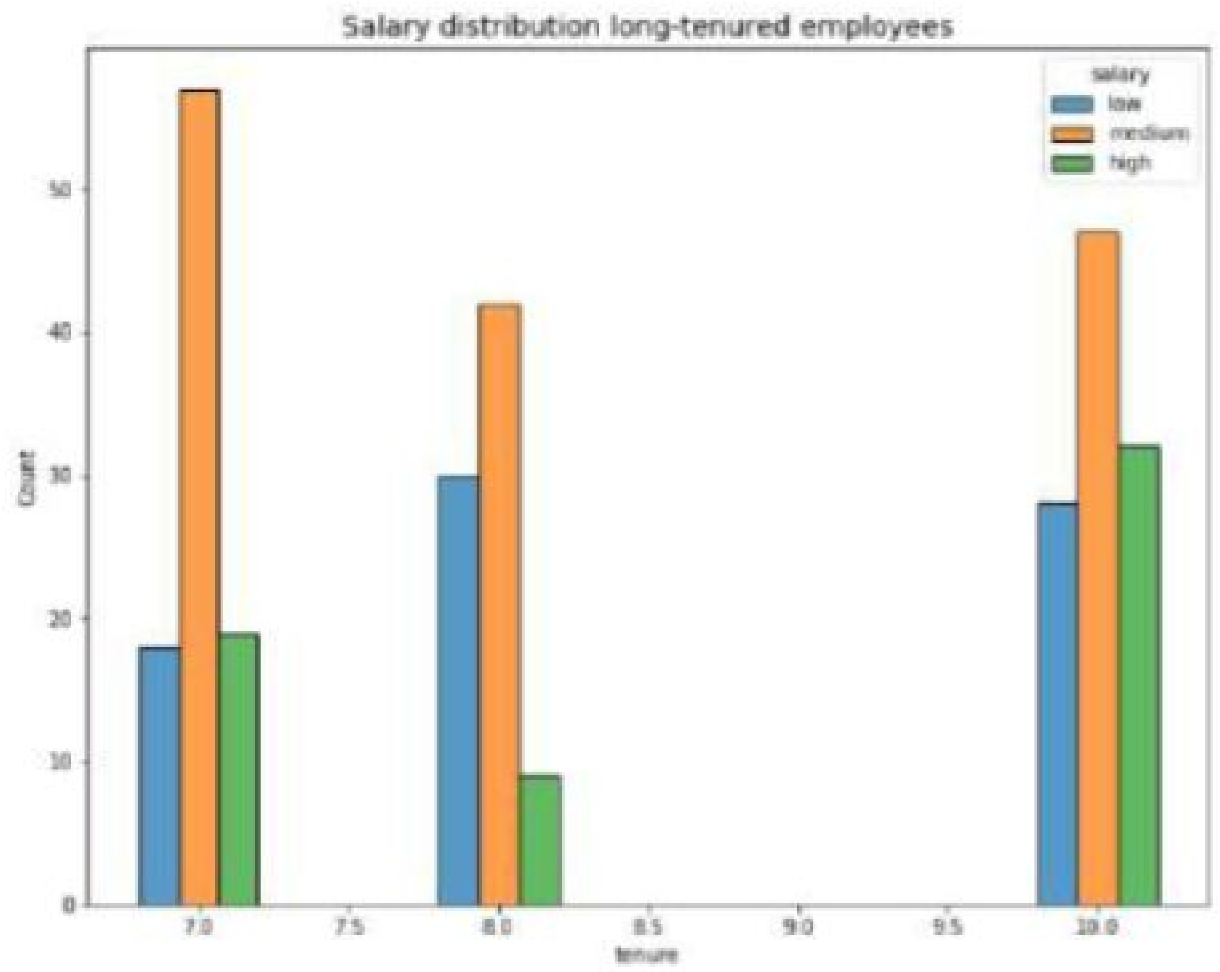
- Employees who **work over 270 hours a month but have not received a promotion_last_5years tend to leave the company.**
- The highest number of employees who left are from the sales, technical, and support departments.

Exploratory Data Analysis - C









- Employees with seven years at the company or more are categorized as long-tenured, and those with six years or less are short-tenured.
- Long-tenured employees, who are **employees with more than six years at the company, are less likely to leave the company.**
- **Among short-tenured employees, the number with low salaries is higher** compared to those with medium or high salaries.
- **The average satisfaction level of employees who left the company is 44%, which is lower** than the 67% average for employees who stayed.

	mean	median
left		
0	0.667365	0.69
1	0.440271	0.41

Insights and Recommendations

General Insights:

- Employees leaving the company appear to be a result of poor management.
- The turnover can be related to a connection of longer hours at work, handling of too many projects, and generally low levels of satisfaction.
- Long hours without promotions or good evaluations tend to be unsatisfactory and could have caused a burnout for the group of employees in the company.
- However, people who have spent over six years with the company are less likely to quit.
- The models and feature importance analysis confirm that employees are overworked.

For improving retention, the following recommendations are advised:

- Limit the number of projects that can be taken on by an employee.
- Promote employees who have at least four years of tenure or probe further into why they are feeling dissatisfied at this stage.
- Reward employees for working longer hours, or ensure they do not have to without remuneration.
- Clearly communicate overtime policies and set explicit expectations about workload and time off.
- Facilitate company-wide and team-level discussions to understand and improve the work culture.
- Reassess the evaluation criteria so that high scores are not only given to employees who work more than 200 hours a month. Apply a justified scale to reward effort and contribution in proportion.
- Besides, data leakage might still be an issue. It could be interesting to check the predictions performance after removing the last_evaluation feature as it may not occur frequently. Since the scores in an evaluation determine whether or not someone leaves, it would probably be more useful to know how to predict a person's performance score. Same with the satisfaction score.

DETAILS

Exploratory Data Analysis - 1

Variable	Description
satisfaction_level	Employee-reported job satisfaction level [0–1]
last_evaluation	Score of employee's last performance review [0–1]
number_project	Number of projects employee contributes to
average_monthly_hours	Average number of hours employee worked per month
time_spend_company	How long the employee has been with the company (years)
Work_accident	Whether or not the employee experienced an accident while at work
left	Whether or not the employee left the company
promotion_last_5years	Whether or not the employee was promoted in the last 5 years
Department	The employee's department
salary	The employee's salary (U.S. dollars)

```
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64
 3   average_montly_hours   14999 non-null  int64
 4   time_spend_company     14999 non-null  int64
 5   Work_accident          14999 non-null  int64
 6   left                   14999 non-null  int64
 7   promotion_last_5years  14999 non-null  int64
 8   Department             14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
```

- There are 14,999 rows and 10 columns.
- All columns are numeric data type (float and integer), except for the department and salary columns (object).

```
df0 = df0.rename(columns={'average_montly_hours': 'average_monthly_hours',
                          'time_spend_company': 'tenure',
                          'Work_accident': 'work_accident',
                          'Department': 'department'
                          })
```

```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_monthly_hours', 'tenure', 'work_accident', 'left',
       'promotion_last_5years', 'department', 'salary'],
      dtype='object')
```

- As a data cleaning step, rename the 'satisfaction_level', 'last_evaluation', 'number_project', 'average_montly_hours', 'time_spend_company', 'Work_accident', 'left', 'promotion_last_5years', 'Department', 'salary' columns.
- Standardize the column names so that they are all in `snake_case`, correct any column names that are misspelled, and make column names more concise as needed.

```
df0.isna().sum()
```

```
satisfaction_level       0
last_evaluation          0
number_project           0
average_monthly_hours    0
tenure                   0
work_accident            0
left                     0
promotion_last_5years    0
department               0
salary                   0
dtype: int64
```

```
df0.duplicated().sum()
```

```
3008
```

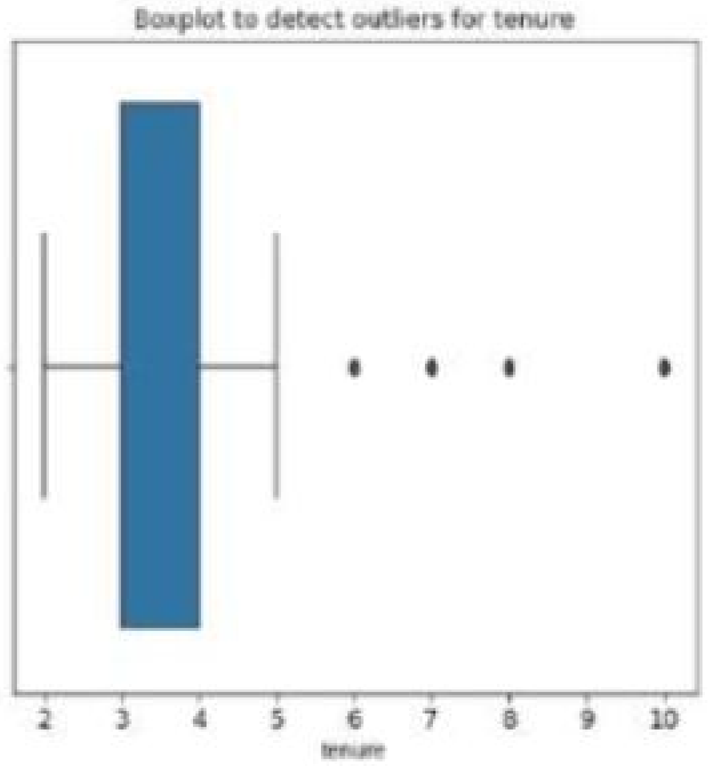
```
df1 = df0.drop_duplicates(keep='first')
```

- There are no missing values in the data.
- There are 3,008 duplicate entries, which need to be deleted.

Exploratory Data Analysis - 2

- Descriptive statistics:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610	0.238083	0.021268
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.425924	0.144281
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.000000	0.000000
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000	0.000000	0.000000
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000	0.000000	0.000000
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000	1.000000	1.000000




```
percentile25 = df1["tenure"].quantile(0.25)
percentile75 = df1["tenure"].quantile(0.75)
iqr = percentile75 - percentile25

upper_limit = percentile75 + (1.5 * iqr)
lower_limit = percentile25 - (1.5 * iqr)
print(f'Upper limit: {upper_limit}')
print(f'Lower limit: {lower_limit}')

outliers = df1[(df1["tenure"] > upper_limit) | (df1["tenure"] < lower_limit)]
print(f'Number of rows in the data containing outliers in tenure: {len(outliers)}')
```

```
Upper limit: 5.5
Lower limit: 1.5
Number of rows in the data containing outliers in tenure: 824
```

- The upper limit for tenure is 5.5, and the lower limit is 1.5. Based on this reference, all values outside this range are considered outliers, with 824 outliers identified in the tenure column.

```
print(df1["left"].value_counts())
print()
print(df1["left"].value_counts(normalize=True))
```

```
0    10000
1     1991
Name: left, dtype: int64

0    0.833959
1    0.166041
Name: left, dtype: float64
```

- The dataset originally had 14,999 entries, but after removing 3,008 duplicates, it now has 11,991 entries.
- **The dataset is imbalanced, with 83% (10,000) of employees staying and only 17% (1,991) leaving.**

Logistic Regression - 1

A. Construct Model

```
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64
 3   average_montly_hours   14999 non-null  int64
 4   time_spend_company     14999 non-null  int64
 5   Work_accident          14999 non-null  int64
 6   left                   14999 non-null  int64
 7   promotion_last_5years  14999 non-null  int64
 8   Department             14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
```

- Out of 14,999 entries, 3,008 duplicate entries and 824 outliers were removed, resulting in 11,167 entries available for modeling.
- These are the columns of the data that will be used for modeling: satisfaction_level, last_evaluation, number_project, average_monthly_hours, tenure, work_accident, left, promotion_last_5years, salary, department.

- Encode categorical variables: department and salary, both of type object datatype.
- The salary variable is categorical and ordinal; that is, its categories have hierarchy. Instead of creating dummy variables, map the levels to numeric values, such as 0 for 'low,' 1 for 'medium,' and 2 for 'high.'
- The department variable is a categorical variable without hierarchy, so it can be encoded using dummy variables for modeling.

- The dataset is imbalanced, with 83% (9,285) of people staying and only 17% (1,882) leaving.

```
df0.duplicated().sum()

3008
```

```
print(f'Number of rows in the data containing outliers in tenure: {len(outliers)}')

Number of rows in the data containing outliers in tenure: 824
```

```
df_enc["salary"] = (df_enc["salary"].astype('category').cat.set_categories(["low", "medium", "high"]).cat.codes)
df_enc = pd.get_dummies(df_enc, drop_first=False)
```

```
df_logreg = df_enc[(df_enc['tenure'] >= lower_limit) & (df_enc['tenure'] <= upper_limit)]
```

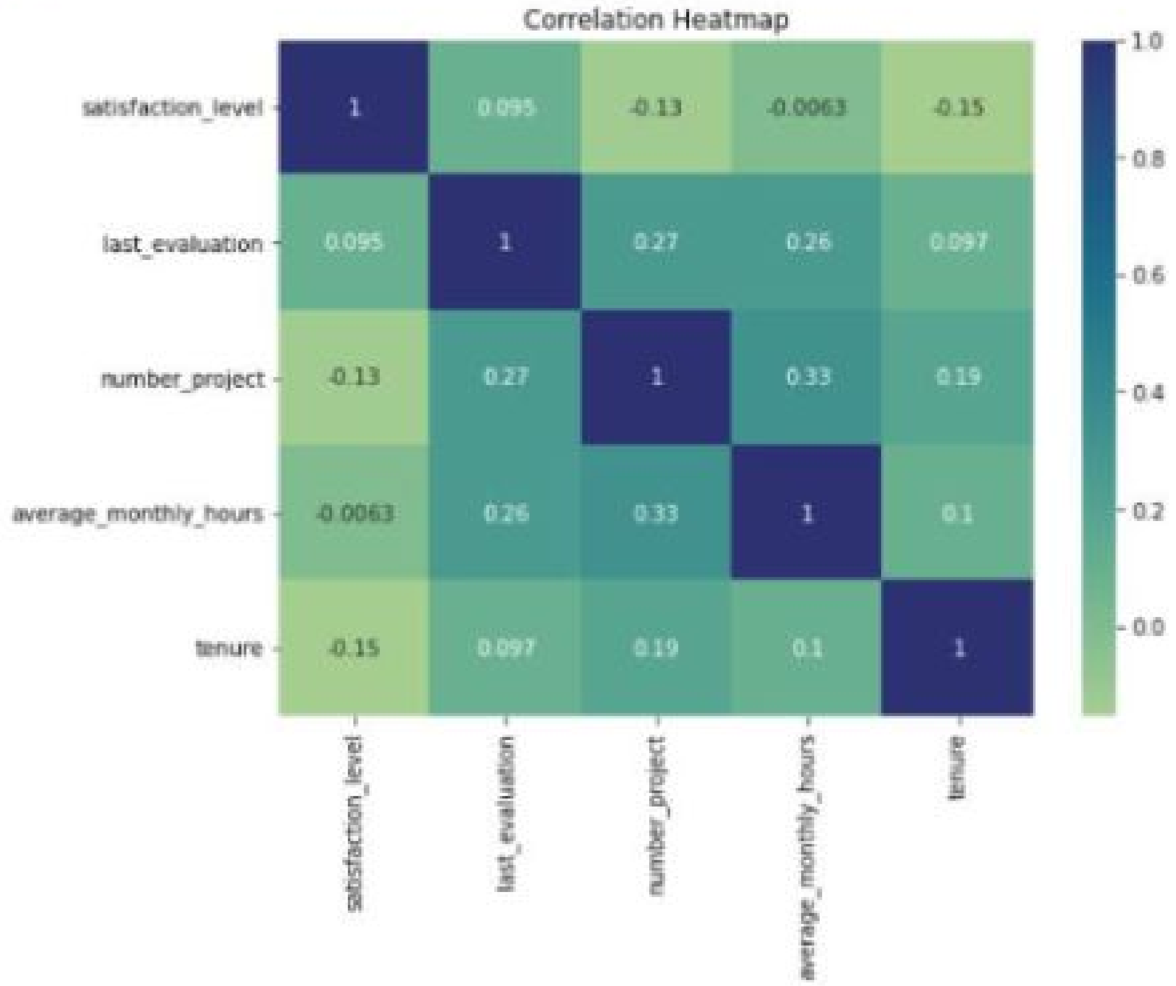
```
df_logreg['left'].value_counts(normalize=True)

0    0.831468
1    0.168532
Name: left, dtype: float64
```

left	count
0	9285
1	1882

```
Index: 11167 entries, 0 to 11999
Data columns (total 19 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   satisfaction_level      11167 non-null  float64
 1   last_evaluation         11167 non-null  float64
 2   number_project          11167 non-null  int64
 3   average_monthly_hours   11167 non-null  int64
 4   tenure                  11167 non-null  int64
 5   work_accident           11167 non-null  int64
 6   left                    11167 non-null  int64
 7   promotion_last_5years   11167 non-null  int64
 8   salary                  11167 non-null  int8
 9   department_IT           11167 non-null  bool
 10  department_RandD        11167 non-null  bool
 11  department_accounting   11167 non-null  bool
 12  department_hr           11167 non-null  bool
 13  department_management   11167 non-null  bool
 14  department_marketing    11167 non-null  bool
 15  department_product_mng  11167 non-null  bool
 16  department_sales        11167 non-null  bool
 17  department_support      11167 non-null  bool
 18  department_technical    11167 non-null  bool
dtypes: bool(10), float64(2), int64(6), int8(1)
```

Logistic Regression - 2



- Among the predictors, last_evaluation, number of projects, and average monthly hours are closely related to each other.

B. Performance Model

```python
y = df_logreg['left']
X = df_logreg.drop('left', axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, stratify=y, random_state=42)

log_clf = LogisticRegression(random_state=42, max_iter=500).fit(X_train, y_train)

y_pred = log_clf.predict(X_test)

log_cm = confusion_matrix(y_test, y_pred, labels=log_clf.classes_)

log_disp = ConfusionMatrixDisplay(confusion_matrix=log_cm,
                                  display_labels=log_clf.classes_)
log_disp.plot(values_format='')
plt.show();
```

	precision	recall	f1-score	support
Predicted would not leave	0.86	0.93	0.90	2321
Predicted would leave	0.44	0.26	0.33	471
accuracy			0.82	2792
macro avg	0.65	0.60	0.61	2792
weighted avg	0.79	0.82	0.80	2792

- The logistic regression model with data train vs test size 75% vs 25% and random state 42, achieved a precision of 79%, recall of 82%, f1-score of 80% (all weighted averages), and accuracy of 82%.
- However, if it's most important to predict employees who leave, then the scores are significantly lower.

Tree Based - 1

A. Decision Tree Model

```
tree = DecisionTreeClassifier(random_state=0)

cv_params = {'max_depth':[4, 6, 8, None],
             'min_samples_leaf': [2, 5, 1],
             'min_samples_split': [2, 4, 6]
             }

scoring = {'accuracy', 'precision', 'recall', 'f1', 'roc_auc'}

tree1 = GridSearchCV(tree, cv_params, scoring=scoring, cv=4, refit='roc_auc')
```

```
    print(f'Best patrameters: {tree1.best_params_}')
    print(f'Best AUC score on CV: {tree1.best_score_}')
```

```
Best patrameters: {'max_depth': 4, 'min_samples_leaf': 2, 'min_samples_split': 2}
Best AUC score on CV: 0.9743823751317063
```

- Strong AUC score (0.974), which shows that this model can predict employees who will leave very well.

B. Random Forest Model

```
rf = RandomForestClassifier(random_state=0)

cv_params = {'max_depth': [3,5, None],
             'max_features': [1.0],
             'max_samples': [0.7, 1.0],
             'min_samples_leaf': [1,2,3],
             'min_samples_split': [2,3,4],
             'n_estimators': [300, 500],
             }

scoring = {'accuracy', 'precision', 'recall', 'f1', 'roc_auc'}

rf1 = GridSearchCV(rf, cv_params, scoring=scoring, cv=4, refit='roc_auc')
```

```
    print(f'Best AUC score on CV: {rf1.best_score_}')
    print(f'Best params: {rf1.best_params_}')
```

```
Best AUC score on CV: 0.9818158627884357
Best params: {'max_depth': 5, 'max_features': 1.0, 'max_samples': 0.7, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 300}
```

```
rf1_cv_results = make_results('random forest cv', rf1, 'auc')
print(tree1_cv_results)
print(rf1_cv_results)
```

model	precision	recall	F1	accuracy	auc
decision tree cv	0.955522	0.91497	0.934765	0.978508	0.974382

model	precision	recall	F1	accuracy	auc
random forest cv	0.970653	0.91497	0.941924	0.981015	0.981816

- The evaluation score of random forest model performs better than the decision tree model, showing it is generally more effective.

Tree Based - 2

Feature Engineering

- The company may not keep on record the satisfaction level for every employee. Another point to consider is that the column average_monthly_hours could lead to data leakage since employees who intend to leave or are soon to be fired by the management will work fewer hours.

```
df2 = df_enc.drop('satisfaction_level', axis=1)
df2['overworked'] = df2['average_monthly_hours']

print('Max hours:', df2['overworked'].max())
print('Min hours:', df2['overworked'].min())
```

```
Max hours: 310
Min hours: 96
```

- Satisfaction_level can be dropped and a new binary feature, overworked, can be created that indicates whether an employee is overworking.
- Define overworked as working more than 175 hours per month. Convert True to 1 and False to 0.

```
df2['overworked'] = (df2['overworked'] > 175).astype(int)

df2 = df2.drop('average_monthly_hours', axis=1)
```

```
y = df2['left']
X = df2.drop('left', axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, stratify=y, random_state=0)
```

- The tree-based model was trained using 75% of the data for training and 25% for testing, with a random state set to 0.

Tree Based - 3

A. Decision Tree Model (with Feature Engineering)

```
tree = DecisionTreeClassifier(random_state=0)

cv_params = {'max_depth':[4, 6, 8, None],
             'min_samples_leaf': [2, 5, 1],
             'min_samples_split': [2, 4, 6]
             }

scoring = {'accuracy', 'precision', 'recall', 'f1', 'roc_auc'}

tree2 = GridSearchCV(tree, cv_params, scoring=scoring, cv=4, refit='roc_auc')
```

```
print(f'Best patrameters: {tree2.best_params_}')

# Check best AUC score on CV
print(f'Best AUC score on CV: {tree2.best_score_}')
```

```
Best patrameters: {'max_depth': 6, 'min_samples_leaf': 2, 'min_samples_split': 6}
Best AUC score on CV: 0.9586752505340426
```

model	precision	recall	F1	accuracy	auc
decision tree cv	0.955522	0.91497	0.934765	0.978508	0.974382

model	precision	recall	F1	accuracy	auc
decision tree2 cv	0.856693	0.903553	0.878882	0.958523	0.958675

- tree2 scores dropped compared to tree because it used fewer features, but the performance remains strong.

B. Random Forest Model (with Feature Engineering)

```
rf = RandomForestClassifier(random_state=0)

cv_params = {'max_depth': [3,5, None],
             'max_features': [1.0],
             'max_samples': [0.7, 1.0],
             'min_samples_leaf': [1,2,3],
             'min_samples_split': [2,3,4],
             'n_estimators': [300, 500],
             }

scoring = {'accuracy', 'precision', 'recall', 'f1', 'roc_auc'}

rf2 = GridSearchCV(rf, cv_params, scoring=scoring, cv=4, refit='roc_auc')
```

```
print(f'Best AUC score on CV: {rf2.best_score_}')
print(f'Best params: {rf2.best_params_}')
```

```
Best AUC score on CV: 0.9648100662833985
Best params: {'max_depth': 5, 'max_features': 1.0, 'max_samples': 0.7, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 300}
```

model	precision	recall	F1	accuracy	auc
random forest cv	0.970653	0.91497	0.941924	0.981015	0.981816

model	precision	recall	F1	accuracy	auc
decision tree2 cv	0.856693	0.903553	0.878882	0.958523	0.958675

model	precision	recall	F1	accuracy	auc
random forest2 cv	0.866758	0.878754	0.872407	0.957411	0.96481

- Random forest2 scores (0.964) dropped slightly because it used fewer features than forest (0.982), but it still outperforms the decision tree2 (0.957) based on the AUC metric.