

HR Analytics: Understanding Employee Satisfaction and Attrition

One of the primary uses of such a dataset is to analyze factors that contribute to employee turnover or attrition. By studying the relationships between variables like satisfaction level, number of projects, salary, and whether an employee has left the company, you can identify key drivers of attrition. This analysis can help the company take proactive measures to improve employee retention.

It includes features like the number of projects, average monthly hours, and overtime hours. Analyzing these variables can provide insights into employee productivity and workload. For example, you can determine whether employees with more projects or longer working hours are more likely to leave or if there's an optimal workload for higher job satisfaction. You can assess whether employees with higher salaries or recent promotions are more likely to stay with the company.

Understanding the relationship between commute distance and employee satisfaction or turnover can help the company make decisions about office locations or remote work policies. You can build predictive models to forecast which employees are most likely to leave the company in the future. This can be used for proactive retention strategies.

The analysis can inform data-driven HR and management decisions. For example, it can help in designing employee engagement programs, setting appropriate salary levels, and identifying departments or teams that may require special attention.

Overall, the analysis of this dataset can provide actionable insights to HR departments and company management to improve employee satisfaction, retention, and overall organizational performance. It can also help in reducing the costs associated with recruiting and training new employees due to high turnover.

There are 12 columns and 2121 rows

Column Names	Data Types	Values	Description
number_project	int64	Numerical	number of projects an employee is involved in.
average_monthly_hours	Object	Categorical	The average number of monthly work hours for an employee.
salary	float64	Numerical	salary level of an employee, categorized as "low," "medium," or "high."
satisfaction_level	int64	Numerical	satisfaction level of an employee, which could be a measure of job

			satisfaction.
work_accident	int64	Numerical	Indicates whether the employee has had a work-related accident (1 for yes, 0 for no).
left	int64	Numerical	Indicates whether the employee has left the company (1 for yes, 0 for no), which could be a target variable for prediction.
time_spend_company	int64	Numerical	number of years the employee has spent at the company.
sales	Object	Categorical	The department or sales sector where the employee works.
promotion_last_5years	int64	Numerical	Indicates whether the employee has been promoted in the last 5 years (1 for yes, 0 for no).
Training Hours	int64	Numerical	number of hours the employee has received training.
commute_distance	int64	Numerical	measure of the distance the employee commutes to work.
overtime_hours	int64	Numerical	The number of hours the employee works overtime.

This dataset is used to study employee behavior and factors that may influence their decision to leave the company. The dataset includes both categorical and numerical features, and it seems to be suitable for building predictive models to understand and predict employee turnover or attrition.

Tools and Techniques Utilized:

- Development Platform: Google Colab (Colaboratory)
- Python Libraries: Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn
- Data Exploration: Descriptive Statistics, Data Visualization (Histograms, Scatter Plots, Bar Plots, Box Plots, Heatmaps, Pie Charts)
- Data Preprocessing: Handling Missing Values, Removing Duplicate Rows, Label Encoding
- Statistical Analysis: Correlation Matrix
- Machine Learning Models: Linear Regression, Random Forest Regressor, Gradient Boosting Regressor, Logistic Regression, Random Forest Classifier, Gradient Boosting Classifier
- Model Evaluation: Mean Squared Error, R-squared, Accuracy Score, Confusion Matrix

CODE AND OUTPUT

Install and import necessary libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LogisticRegression
from random import sample
```

Load the HRM dataset

```
hrm = pd.read_csv('/content/hr_analysis.csv')
```

	number_project	average_monthly_hours	salary	satisfaction_level	work_accident	left	time_spend_company	sales	promotion_last_5years	Training Hours	com
0	5	164	medium	0.41	1	0	3	sales	0	16	
1	5	200	medium	0.43	1	0	1	support	0	18	
2	6	225	medium	0.32	1	1	2	sales	0	16	
3	5	198	medium	0.38	1	1	1	hr	0	15	
4	3	153	high	0.62	0	0	3	management	0	25	

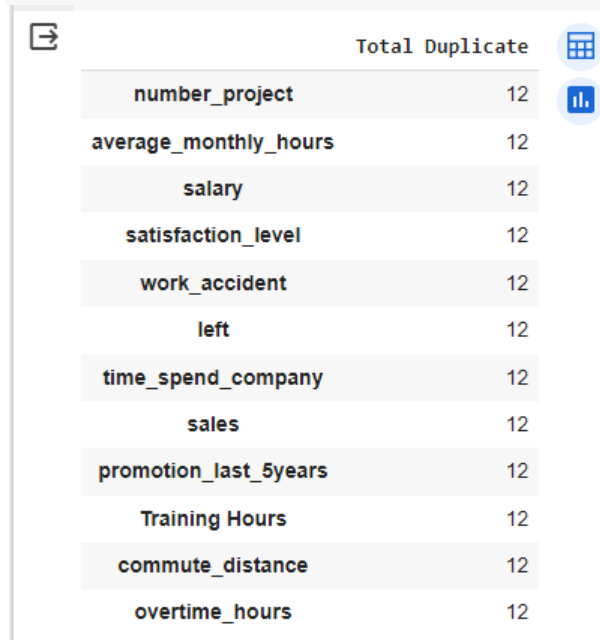
Check the structure of the dataset

```
print(hrm.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2121 entries, 0 to 2120
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  -
 0   number_project        2121 non-null  int64  
 1   average_monthly_hours 2121 non-null  int64  
 2   salary                2121 non-null  object  
 3   satisfaction_level     2121 non-null  float64 
 4   work_accident         2121 non-null  int64  
 5   left                  2121 non-null  int64  
 6   time_spend_company    2121 non-null  int64  
 7   sales                 2121 non-null  object  
 8   promotion_last_5years 2121 non-null  int64  
 9   Training Hours        2121 non-null  int64  
10   commute_distance      2121 non-null  int64  
11   overtime_hours        2121 non-null  int64  
dtypes: float64(1), int64(9), object(2)
memory usage: 199.0+ KB
```

Finding Duplicate Value

```
data= pd.DataFrame(hrm.loc[hrm.duplicated()].count())
data.columns = ['Total Duplicate']
data
```



The image shows a Jupyter Notebook interface with a table titled 'Total Duplicate'. The table has two columns: the column names from the original dataset and the count of duplicates for each. All counts are 12. To the right of the table are two icons: a calculator and a bar chart.

	Total Duplicate
number_project	12
average_monthly_hours	12
salary	12
satisfaction_level	12
work_accident	12
left	12
time_spend_company	12
sales	12
promotion_last_5years	12
Training Hours	12
commute_distance	12
overtime_hours	12

Drop duplicate values

```
hrm.drop_duplicates(inplace=True)
data= pd.DataFrame(hrm.loc[hrm.duplicated()].count())
data.columns = ['Total Duplicate']
data
```



Total Duplicate



number_project	0
average_monthly_hours	0
salary	0
satisfaction_level	0
work_accident	0
left	0
time_spend_company	0
sales	0
promotion_last_5years	0
Training Hours	0
commute_distance	0
overtime_hours	0

Check for missing values

```
print(hrm.isna().sum())
```



```
number_project      0
average_monthly_hours  0
salary              0
satisfaction_level   0
work_accident        0
left                0
time_spend_company   0
sales               0
promotion_last_5years 0
Training Hours       0
commute_distance     0
overtime_hours       0
dtype: int64
```

Summary statistics

```
print(hrm.describe())
```

	number_project	average_monthly_hours	satisfaction_level	\
count	2109.000000	2109.000000	2109.000000	
mean	3.903272	180.467046	0.498122	
std	1.340556	29.465905	0.178828	
min	2.000000	130.000000	0.100000	
25%	3.000000	154.000000	0.370000	
50%	4.000000	181.000000	0.500000	
75%	5.000000	206.000000	0.630000	
max	6.000000	242.000000	0.900000	

	work_accident	left	time_spend_company	promotion_last_5years	\
count	2109.000000	2109.000000	2109.000000	2109.000000	
mean	0.370792	0.427691	2.339497	0.083926	
std	0.483131	0.494861	1.266058	0.277343	
min	0.000000	0.000000	1.000000	0.000000	
25%	0.000000	0.000000	1.000000	0.000000	
50%	0.000000	0.000000	2.000000	0.000000	
75%	1.000000	1.000000	3.000000	0.000000	
max	1.000000	1.000000	5.000000	1.000000	

	Training_Hours	commute_distance	overtime_hours
count	2109.000000	2109.000000	2109.000000
mean	18.011380	26.905168	21.284969
std	4.451815	8.402331	8.674532
min	12.000000	7.000000	6.000000
25%	15.000000	21.000000	19.000000
50%	18.000000	29.000000	23.000000
75%	20.000000	33.000000	27.000000
max	25.000000	40.000000	38.000000

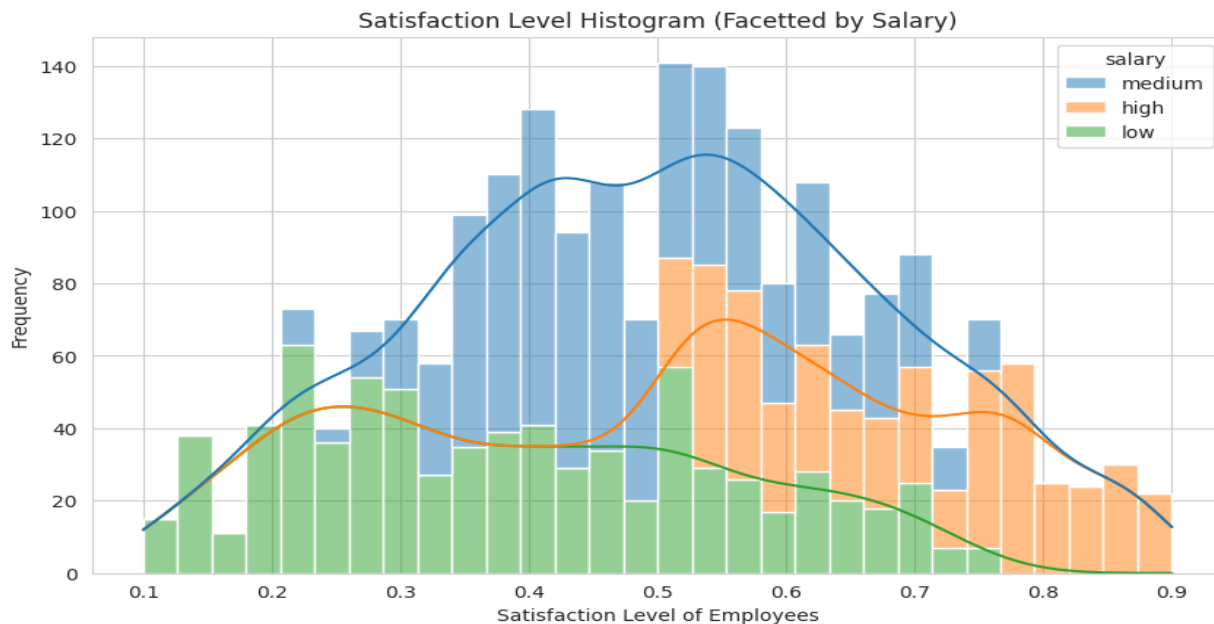
Satisfaction level statistics splitted by salary ranges

```
print(hrm.groupby('salary')['satisfaction_level'].describe())
```

	count	mean	std	min	25%	50%	75%	max
salary								
high	509.0	0.686621	0.114891	0.51	0.58	0.69	0.78	0.90
low	768.0	0.390065	0.166618	0.10	0.25	0.38	0.52	0.75
medium	832.0	0.482548	0.120961	0.21	0.39	0.47	0.57	0.75

Facetted histograms by salary

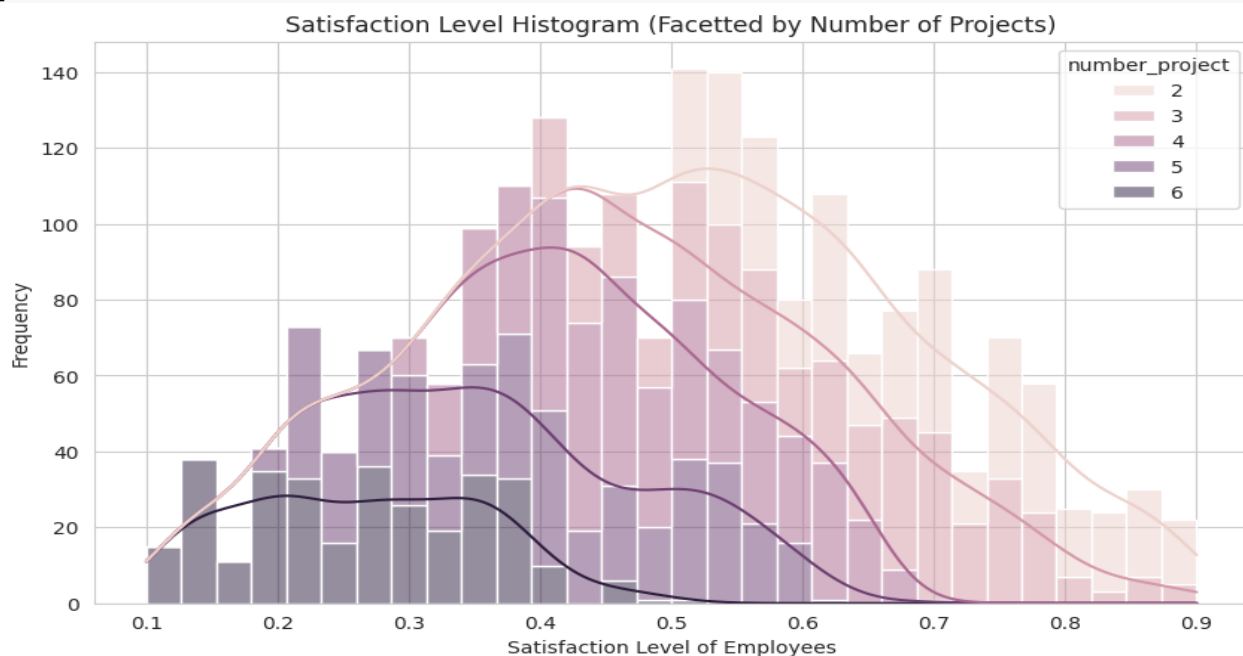
```
plt.figure(figsize=(10, 6))
sns.histplot(data=hrm, x='satisfaction_level', bins=30, color='red', kde=True,
hue='salary', multiple='stack')
plt.title("Satisfaction Level Histogram (Facetted by Salary)")
plt.xlabel("Satisfaction Level of Employees")
plt.ylabel("Frequency")
plt.show()
```



The above graph indicates that employee with high salary have higher satisfaction level and vice versa. So hike in their salary may make them more satisfied.

Facetted histograms by number of projects

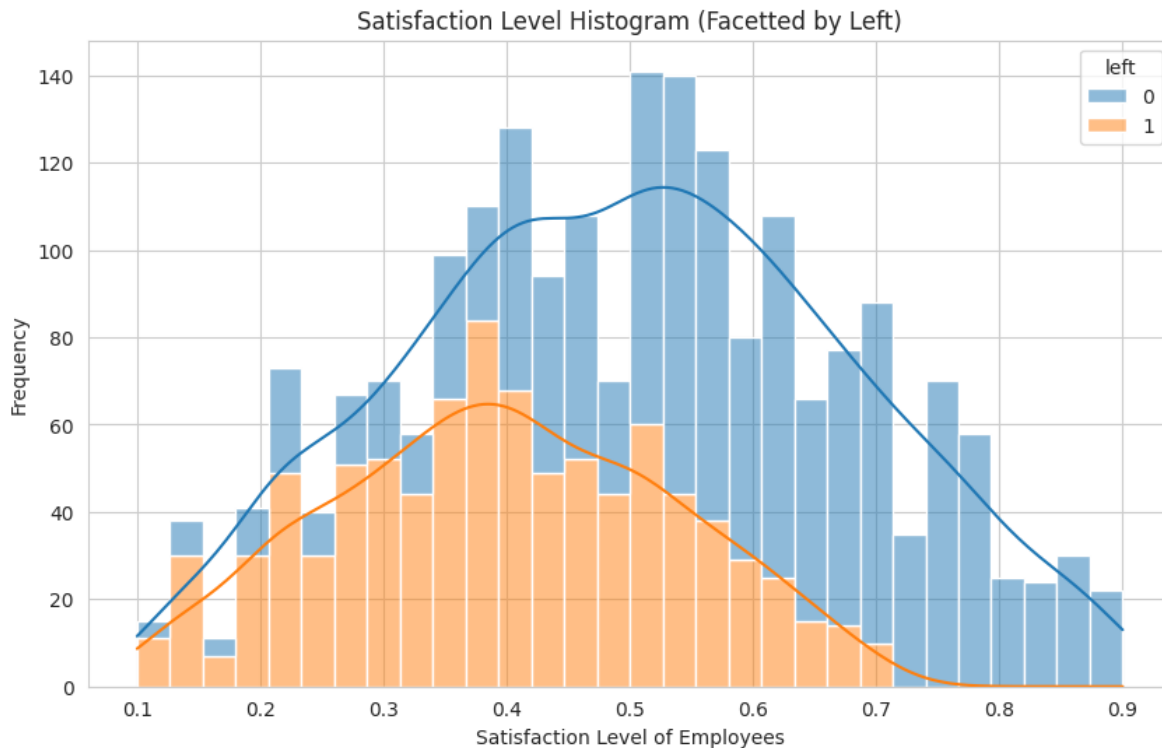
```
plt.figure(figsize=(10, 6))
sns.histplot(data=hrm, x='satisfaction_level', bins=30, kde=True,
hue='number_project', multiple='stack')
plt.title("Satisfaction Level Histogram (Facetted by Number of Projects)")
plt.xlabel("Satisfaction Level of Employees")
plt.ylabel("Frequency")
plt.show()
```



The above graph indicates that employee with max no.of projects have lower satisfaction level and vice versa

Facetted histograms by left or not

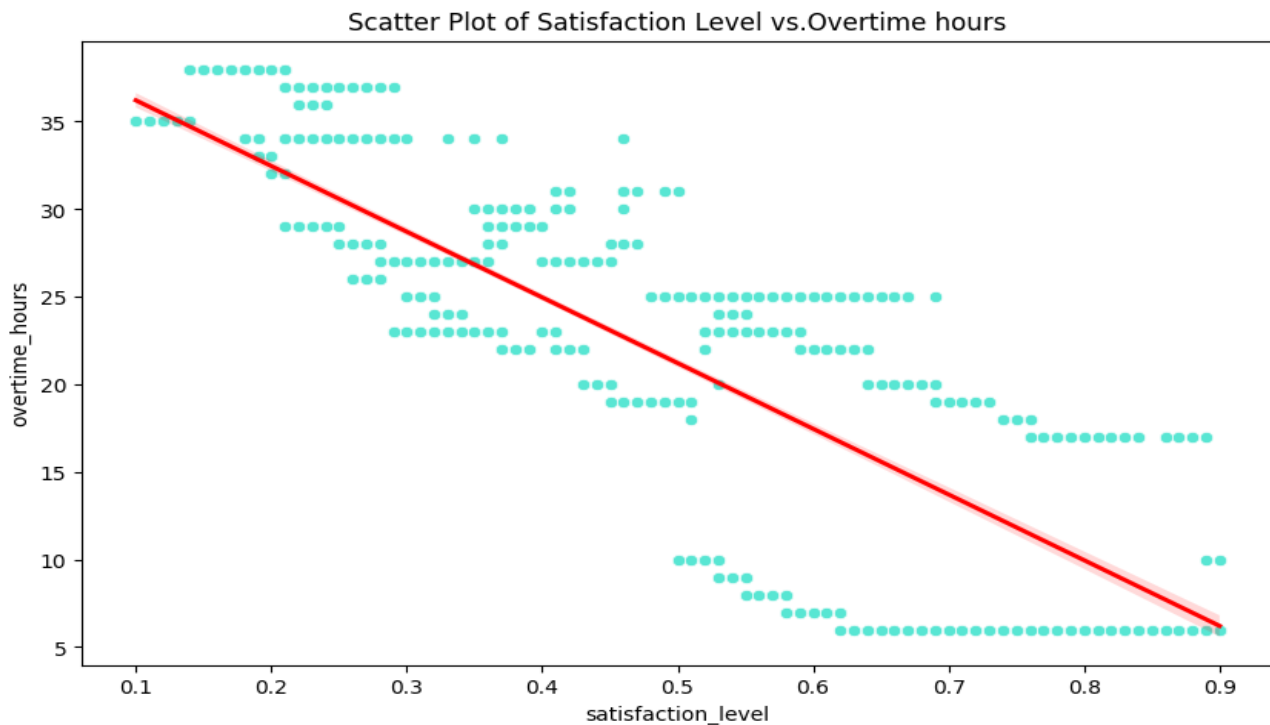
```
plt.figure(figsize=(10, 6))
sns.histplot(data=hrm, x='satisfaction_level', bins=30, color='red', kde=True,
hue='left', multiple='stack')
plt.title("Satisfaction Level Histogram (Facetted by Left)")
plt.xlabel("Satisfaction Level of Employees")
plt.ylabel("Frequency")
plt.show()
```



This graph indicates that employee with higher satisfaction level stayed in the company and with lower satisfaction level left the company

Scatter Plot of Satisfaction Level vs. Overtime hours

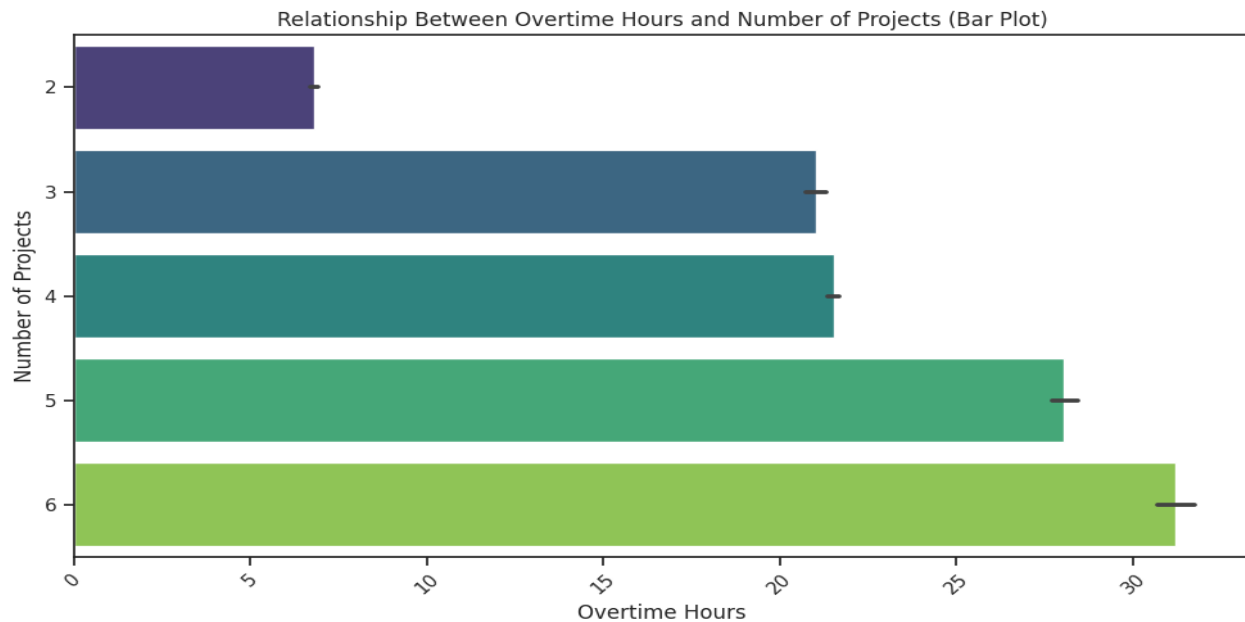
```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=hrm['satisfaction_level'],
y=hrm['overtime_hours'], c="#58E6D3")
plt.xlabel("Satisfaction Level")
plt.ylabel("Overtime hours")
plt.title("Scatter Plot of Satisfaction Level vs.Overtime hours")
sns.regplot(x=hrm['satisfaction_level'], y=hrm['overtime_hours'], scatter=False,
color='r')
plt.show()
```

Employees doing overtime due to no. of projects given to them makes them less satisfied to continue in the company

Bar plot to visualize the relationship between overtime_hours and number_project

```
plt.figure(figsize=(10, 6))
sns.barplot(x="overtime_hours", y="number_project", data=hrm, palette="viridis")
plt.xlabel("Overtime Hours")
plt.ylabel("Number of Projects")
plt.title("Relationship Between Overtime Hours and Number of Projects (Bar Plot)")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



Employees with higher no.of projects have to do overtime.

salary vs left by Salary Ranges & number of project vs left by Salary Ranges

Cross-tabulation of left vs salary

```
print(pd.crosstab(hrm['left'], hrm['salary']))
```

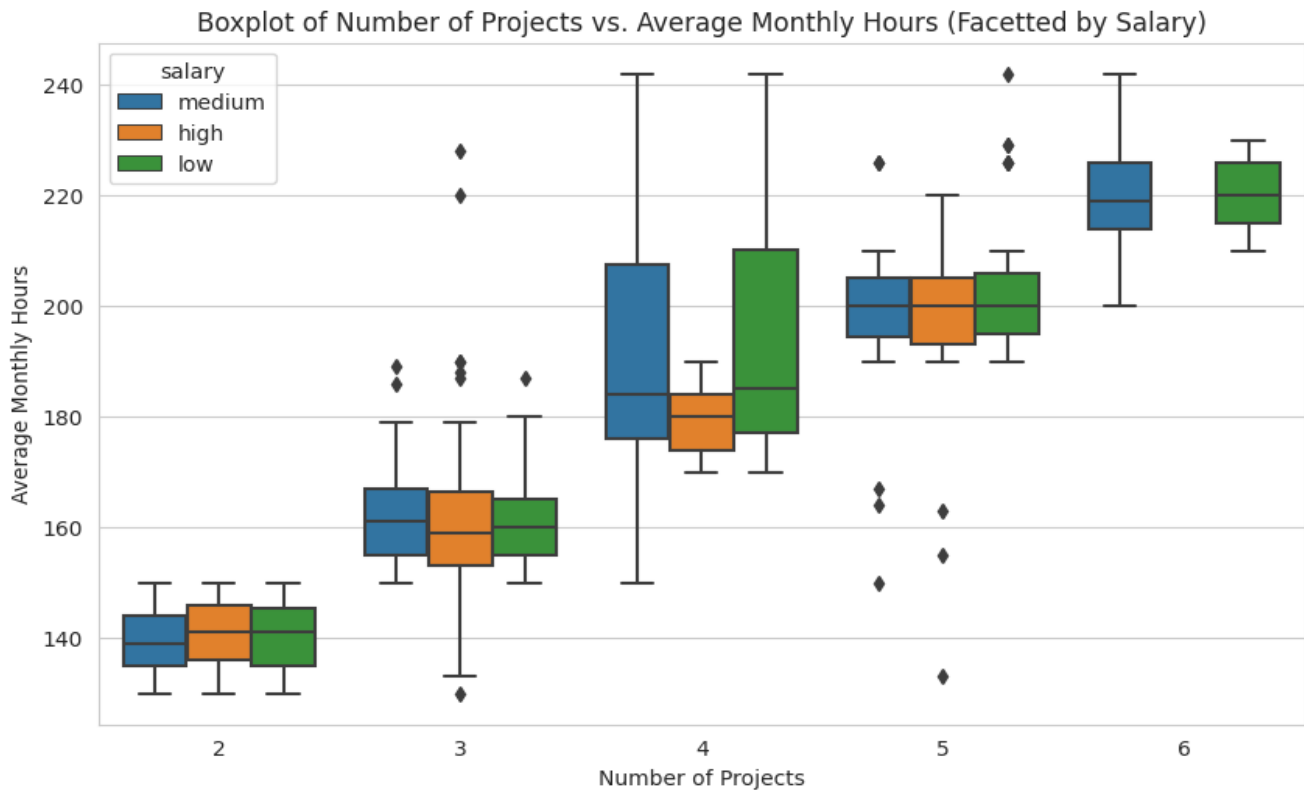
Cross-tabulation of left vs number_project

```
print(pd.crosstab(hrm['left'], hrm['number_project']))
```

```
salary high low medium
left
0      407 363   437
1      102 405   395
number_project 2  3  4  5  6
left
0           421 411 163 134 78
1           0  18 343 305 236
```

Boxplot of number of projects vs Average monthly hours at workplace of employees facettted by salary

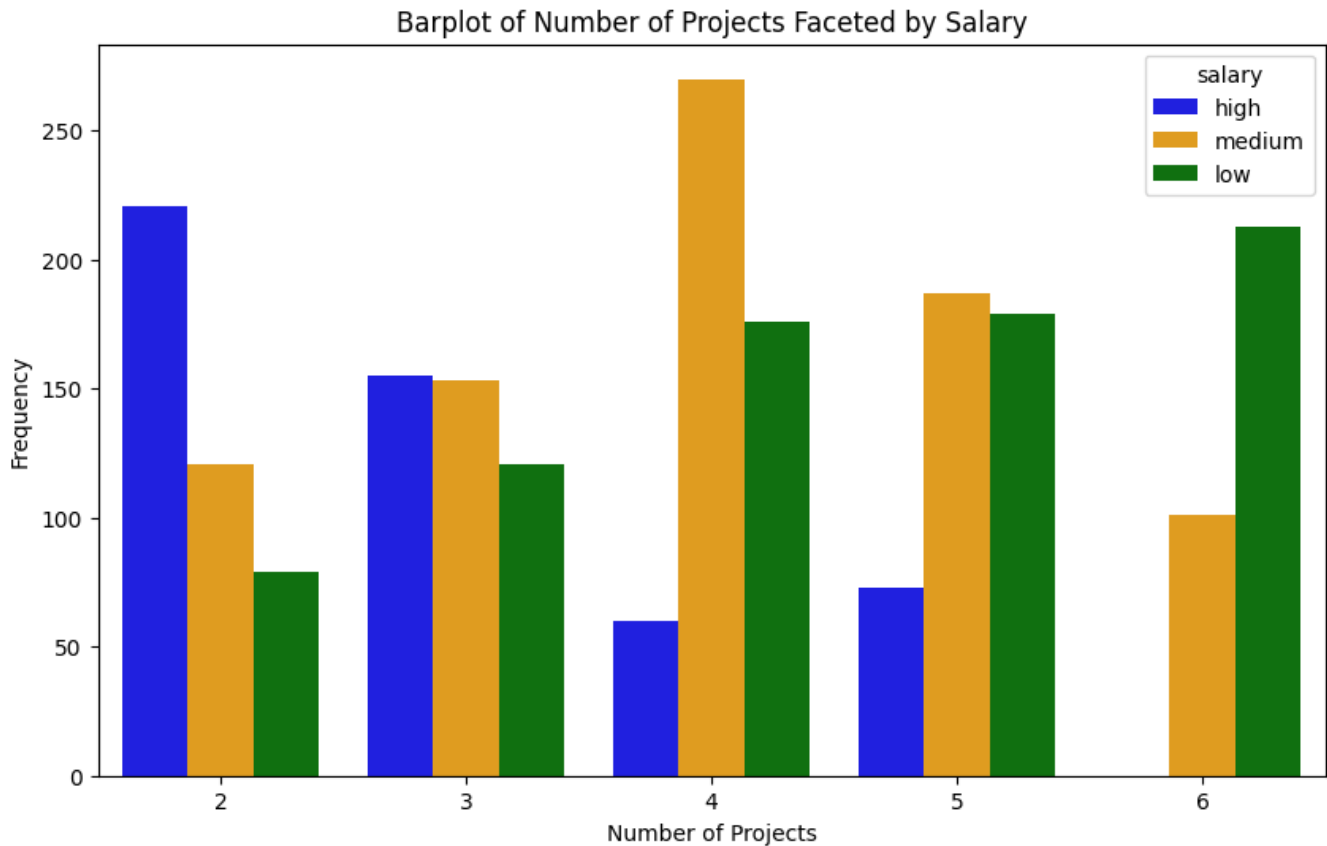
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=hrm, x='number_project', y='average_monthly_hours',
hue='salary')
plt.xlabel("Number of Projects")
plt.ylabel("Average Monthly Hours")
plt.title("Boxplot of Number of Projects vs. Average Monthly Hours (Facettted by Salary)")
plt.show()
```



We can see that increase in number of projects takes employee to work for more time regardless of salary structure. So distributing the projects equally may help the employees feel less pressurized.

Facetted barplots by salary

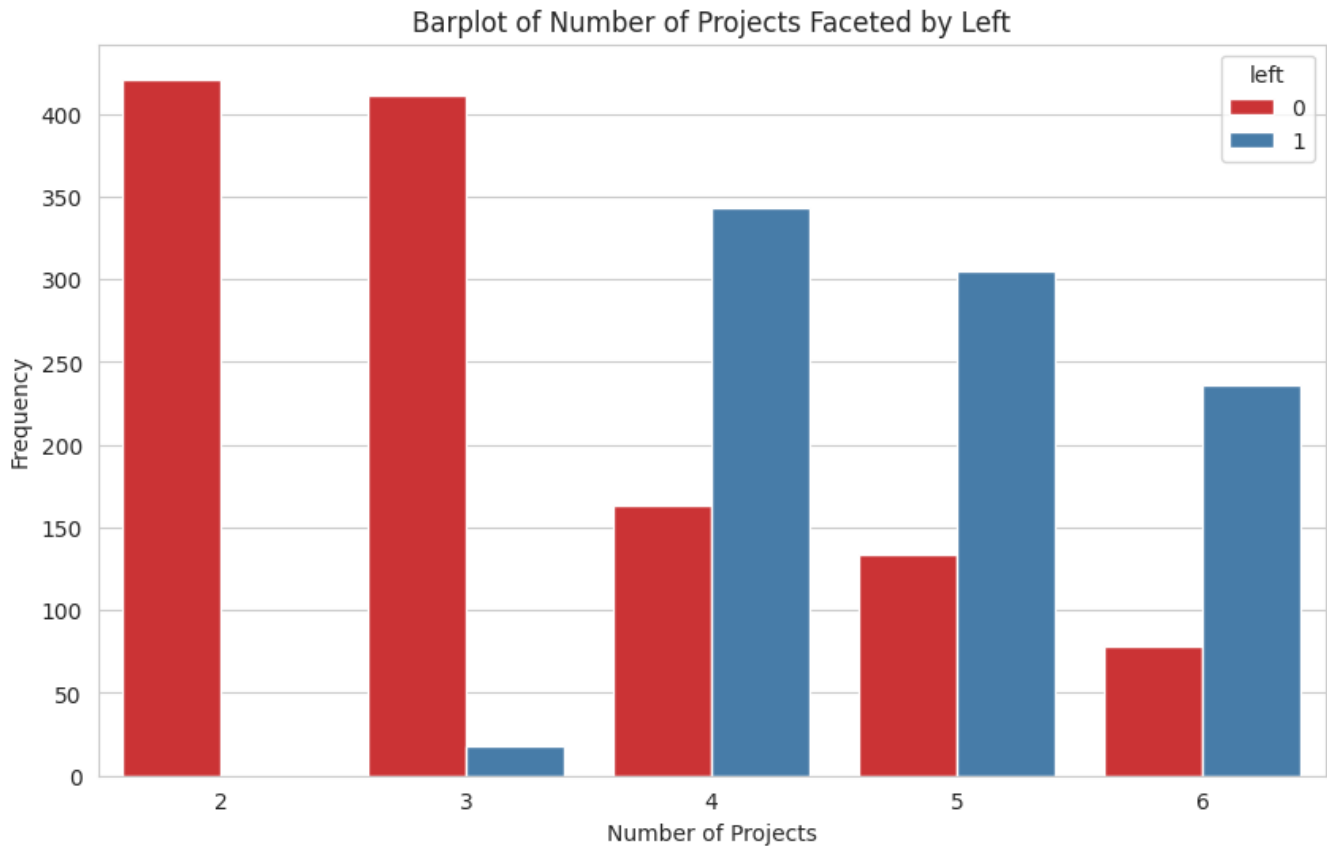
```
plt.figure(figsize=(10, 6))
sns.countplot(data=hrm, x='number_project', hue='salary', palette=['blue',
'orange', 'green'])
plt.xlabel("Number of Projects")
plt.ylabel("Frequency")
plt.title("Barplot of Number of Projects Faceted by Salary")
plt.show()
```



Salary: "high": 1, "medium": 2, "low": 3. Employees with higher salary gets less number of projects and vice versa

Facetted barplots by left or not

```
plt.figure(figsize=(10, 6))
sns.countplot(data=hrm, x='number_project', hue='left' )
plt.xlabel("Number of Projects")
plt.ylabel("Frequency")
plt.title("Barplot of Number of Projects Faceted by Left")
plt.show()
```



Employees left the company when they are given higher no. of projects so we should minimize the no. of projects so that we can control the employee leaving the company

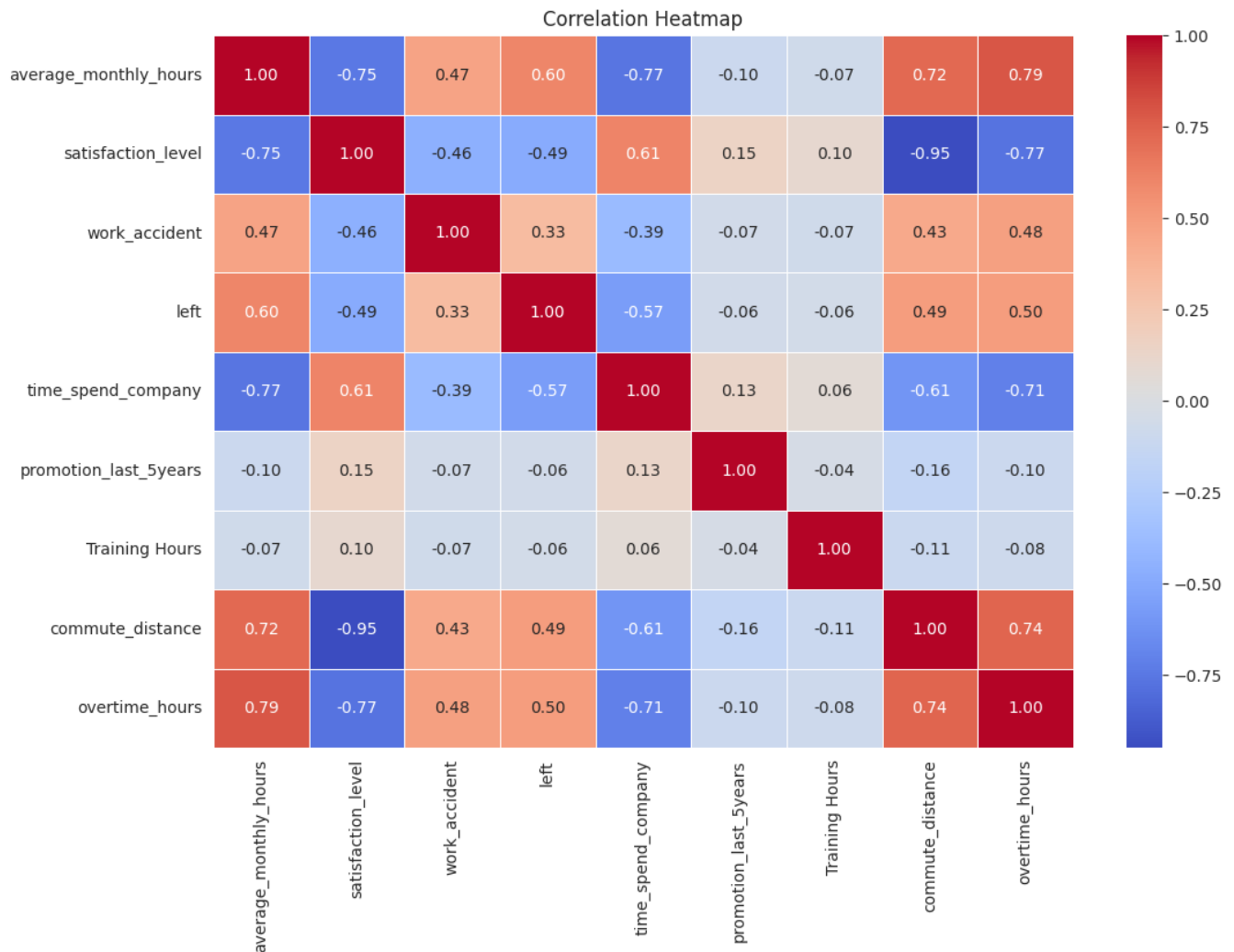
Correlation matrix and Heatmap

```
correlation_matrix = hrm.corr()
print(correlation_matrix)
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",
            linewidths=.5)
plt.title("Correlation Heatmap")
plt.show()
```

<ipython-input-550-e156cb1a535b>:1: FutureWarning: The default value of numeric_only in

```
correlation_matrix = hrm.corr()
          average_monthly_hours  satisfaction_level \
average_monthly_hours          1.000000          -0.747189
satisfaction_level          -0.747189          1.000000
work_accident                0.465382          -0.462054
left                         0.598276          -0.488168
time_spend_company          -0.766306           0.607178
promotion_last_5years        -0.100347           0.148852
Training Hours              -0.073290           0.098728
commute_distance             0.723897          -0.950506
overtime_hours              0.789808          -0.773040

          work_accident    left  time_spend_company \
average_monthly_hours    0.465382  0.598276          -0.766306
satisfaction_level      -0.462054 -0.488168           0.607178
work_accident           1.000000  0.326488          -0.390480
left                    0.326488  1.000000          -0.568805
time_spend_company      -0.390480 -0.568805           1.000000
promotion last 5years    -0.065957 -0.061183           0.128223
```



We can see correlation values for each column lying between 1 to -1.

'average_monthly_hours' vs. 'left'

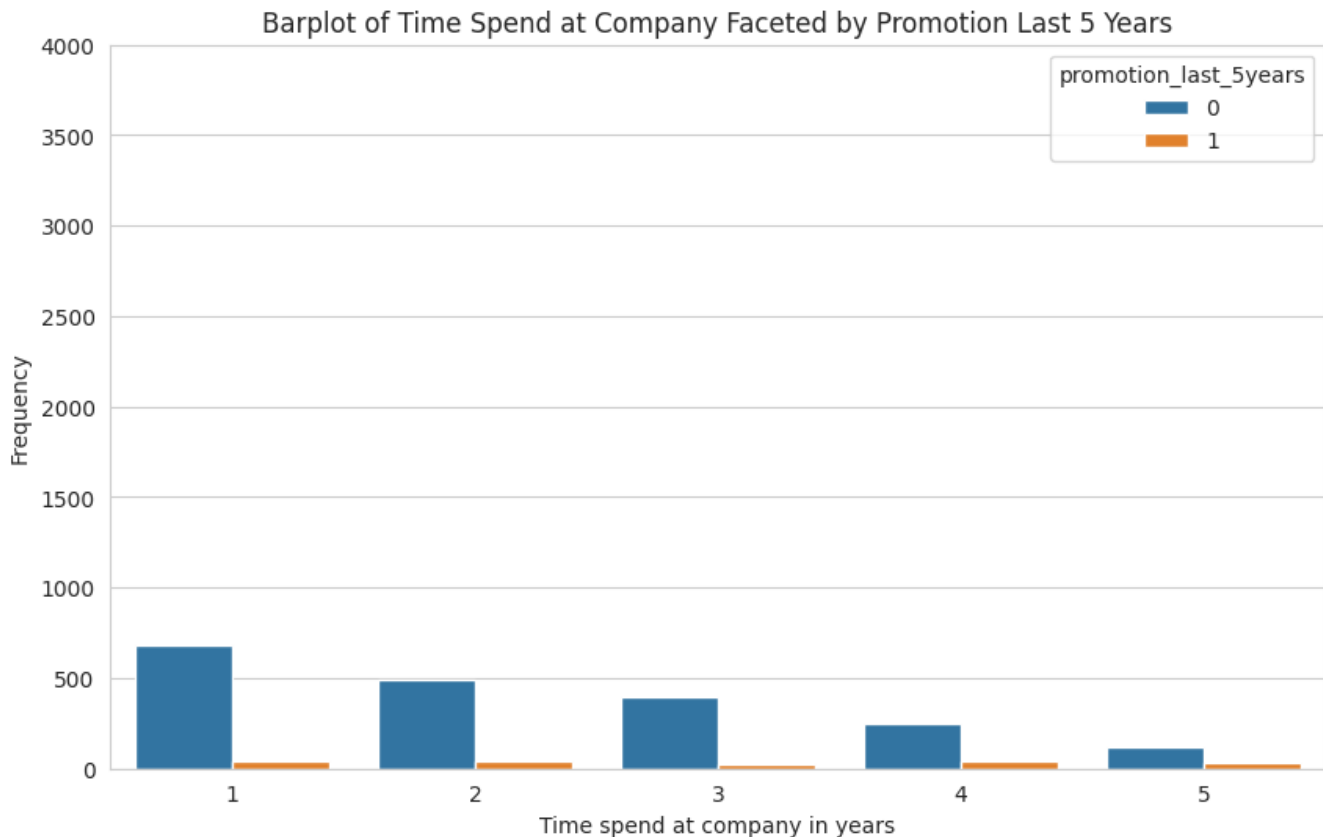
```
hrm.groupby('left')['average_monthly_hours'].describe()
```

	count	mean	std	min	25%	50%	75%	max
left								
0	1207.0	165.231152	27.188558	130.0	144.00	158.0	182.0	242.0
1	902.0	200.854767	17.742891	160.0	186.25	201.0	216.0	242.0

With the above statistical information as the average working hours are increasing the count of employees leaving the company also increases.

Bar plot of 'time_spend_company' faceted by 'promotion_last_5years'

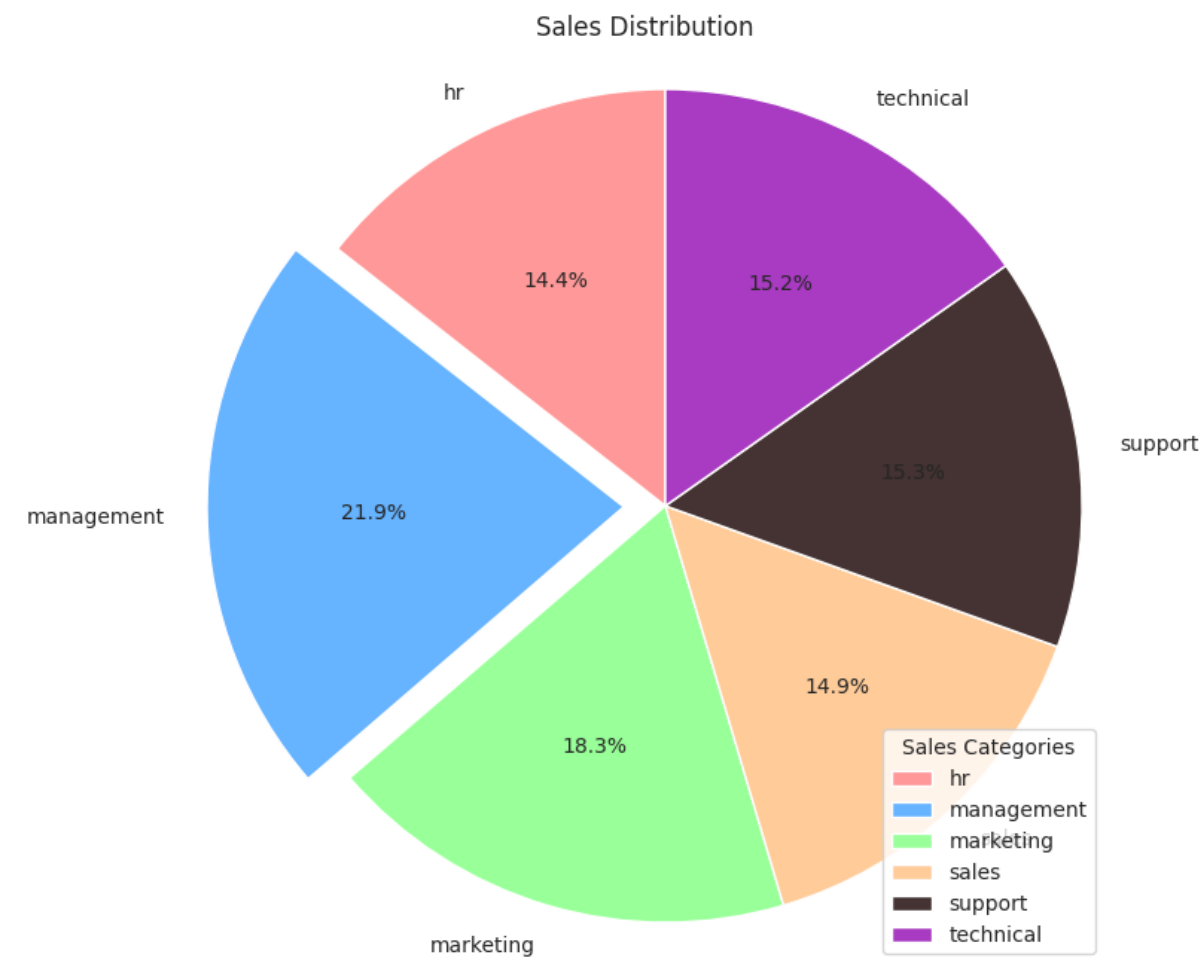
```
plt.figure(figsize=(10, 6))
sns.countplot(x='time_spend_company', data=hrm, hue='promotion_last_5years')
plt.xlabel("Time spend at company in years")
plt.ylabel("Frequency")
plt.title("Barplot of Time Spend at Company Faceted by Promotion Last 5 Years")
plt.ylim(0, 4000) # Set y-axis limits
plt.show()
```



Promotion given to the employees is very low. So to make the employees more satisfied Hr should include programs so that they can seek promotions.

Piechart of 'Sales Distribution'

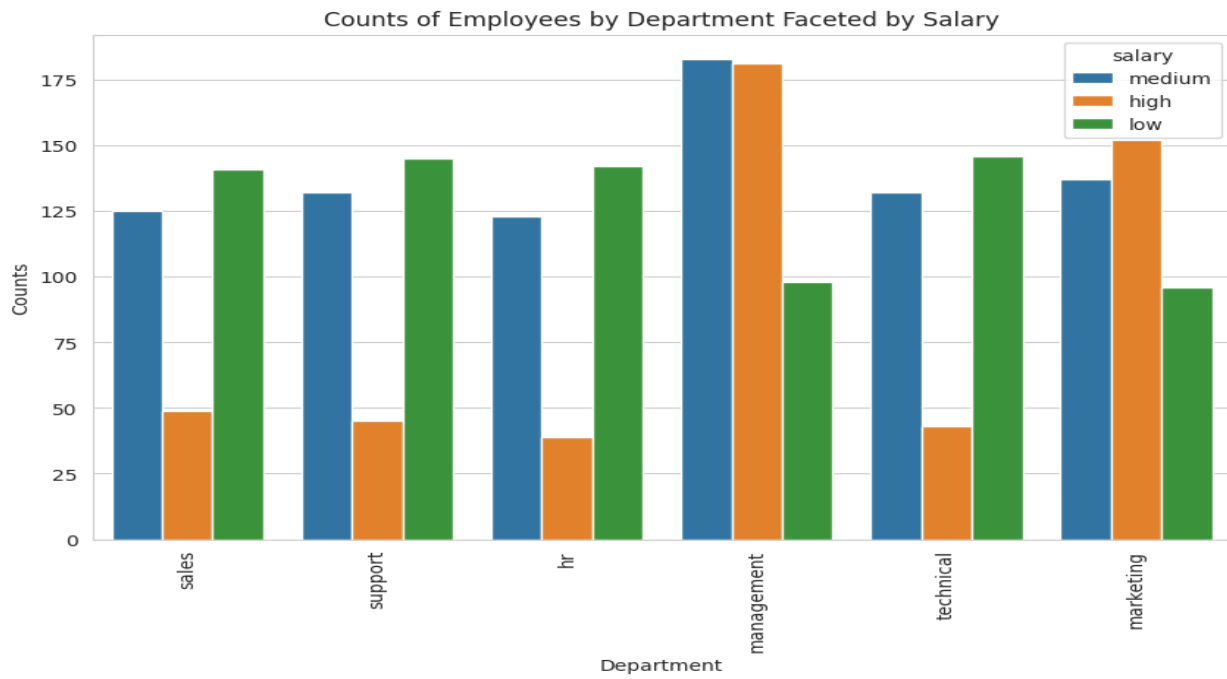
```
sales_data = hrm.groupby('sales').size()
plt.figure(figsize=(8, 8))
colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#443332', '#A83BC2']
explode = (0, 0.1, 0, 0, 0, 0)
plt.pie(sales_data, labels=sales_data.index, autopct='%1.1f%%', startangle=90,
        colors=colors, explode=explode)
plt.title("Sales Distribution")
plt.axis('equal')
plt.legend(sales_data.index, title="Sales Categories", loc='lower right')
plt.show()
```



Management department has the most no.of employees

Bar plot of 'sales' faceted by 'salary'

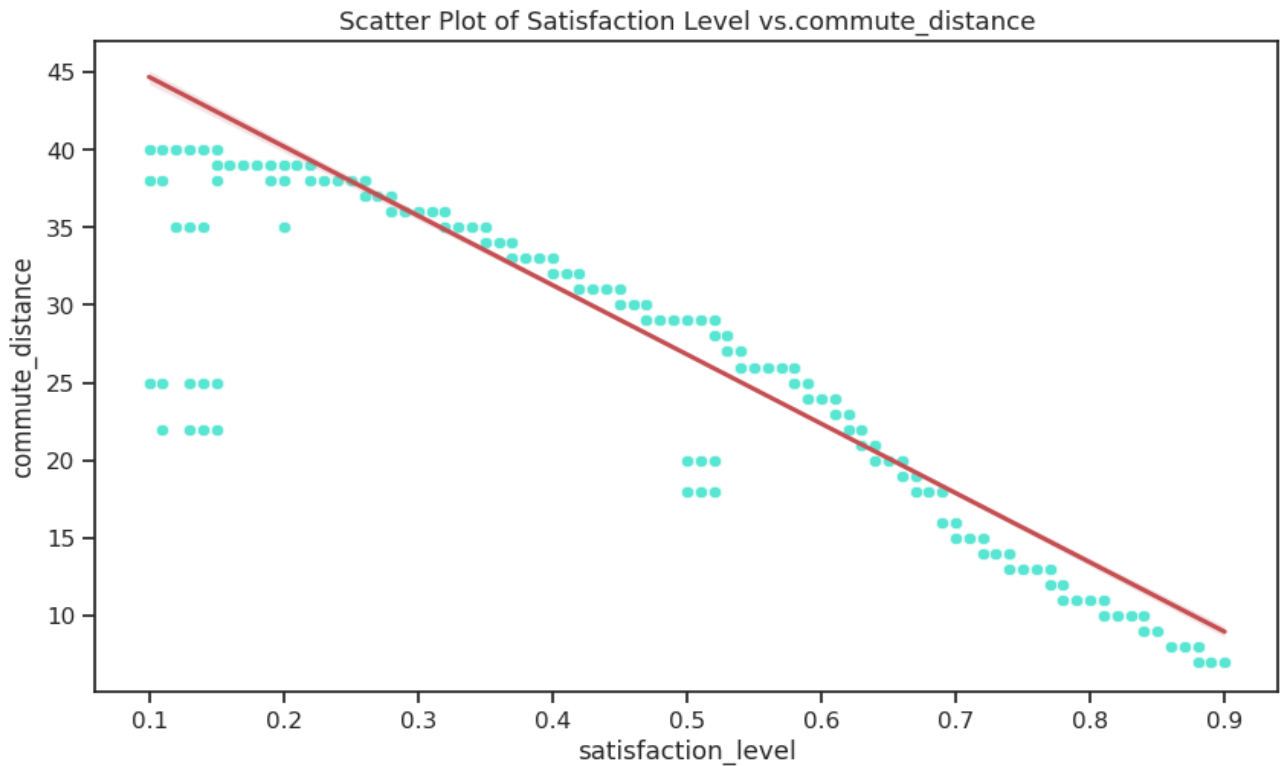
```
plt.figure(figsize=(10, 6))
sns.countplot(x='sales', data=hrm, hue='salary')
plt.xlabel("Department")
plt.ylabel("Counts")
plt.title("Counts of Employees by Department Faceted by Salary")
plt.xticks(rotation=90)
plt.show()
```

Management and Marketing department has the most no.of employees having high salary among other departments

Scatter Plot of Satisfaction Level vs.Commute_distance

```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=hrm['satisfaction_level'],
y=hrm['commute_distance'],c="#58E6D3")
plt.xlabel("Satisfaction Level")
plt.ylabel("commute_distance")
plt.title("Scatter Plot of Satisfaction Level vs.commute_distance")
sns.regplot(x=hrm['satisfaction_level'], y=hrm['commute_distance'],
scatter=False, color='r')
plt.show()
```



Employees coming from far areas to company are less satisfied so arranging remote working facility or work from home can make them more satisfied

Convert salary category to numeric datatype

```
salary_mapping = {"low": 1, "medium": 2, "high": 3}
hrm['salary'] = hrm['salary'].map(salary_mapping)
hrm.dtypes
```

```
number_project      category
average_monthly_hours  int64
salary                int64
satisfaction_level     float64
work_accident         int64
left                  int64
time_spend_company    int64
sales                 object
promotion_last_5years  int64
Training Hours        int64
commute_distance       int64
overtime_hours        int64
dtype: object
```

DATA MODELING

Dependent variable as satisfaction_level

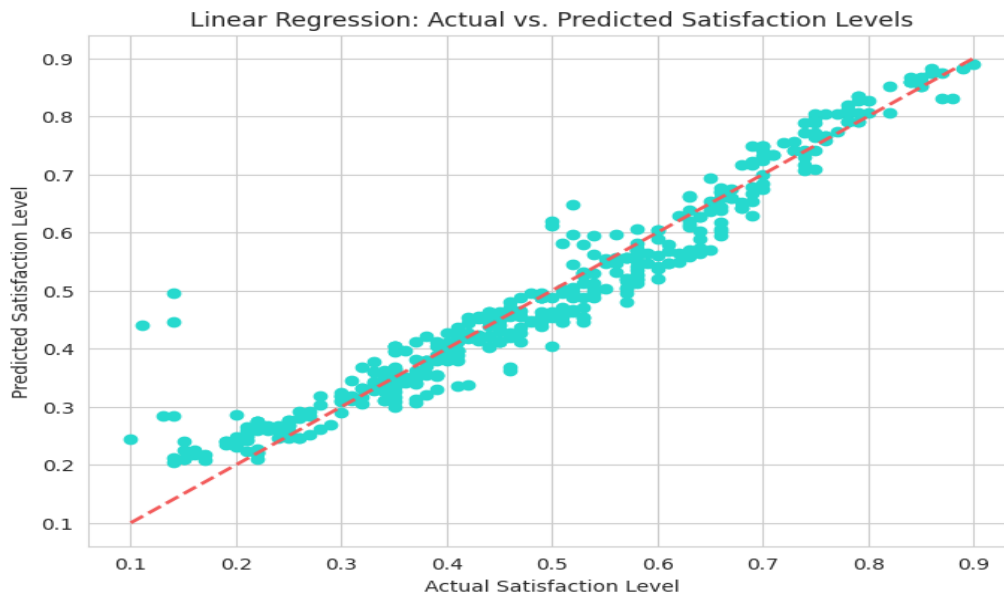
Linear regression

```
# Define features and target for Linear regression
X_reg = hrm.drop(columns=['satisfaction_level', 'sales', 'promotion_last_5years',
                          'Training Hours'])
y_reg = hrm['satisfaction_level']
# Split data into training and testing sets
X_train_reg, X_test_reg, y_train_reg, y_test_reg = train_test_split(X_reg,
                                                                    y_reg,
                                                                    test_size=0.2,
                                                                    random_state=42)
# Create and train the linear regression model
regression_model = LinearRegression()
regression_model.fit(X_train_reg, y_train_reg)
# Make predictions
y_pred_reg = regression_model.predict(X_test_reg)
# Evaluate the model
mse_reg = mean_squared_error(y_test_reg, y_pred_reg)
r_squared_reg = r2_score(y_test_reg, y_pred_reg)
print("Linear Regression Mean Squared Error:", round(mse_reg, 4))
print("Linear Regression R-squared:", round(r_squared_reg, 4))

Linear Regression Mean Squared Error: 0.0023
Linear Regression R-squared: 0.9333
```

The Linear Regression model achieved a Mean Squared Error (MSE) of 0.0023, indicating accurate predictions of employee satisfaction levels. The R-squared (R^2) value of 0.9333 implies that approximately 93.33% of the variance in satisfaction levels is explained by the chosen features, demonstrating the model's strong explanatory power and goodness of fit to the data.

```
# Plot the regression results
plt.figure(figsize=(8, 6))
plt.scatter(y_test_reg, y_pred_reg, c='#26D9CE')
plt.plot([min(y_test_reg), max(y_test_reg)], [min(y_test_reg), max(y_test_reg)],
         color='#F35B5B', linestyle='--', lw=2)
plt.xlabel("Actual Satisfaction Level")
plt.ylabel("Predicted Satisfaction Level")
plt.title("Linear Regression: Actual vs. Predicted Satisfaction Levels")
plt.show()
```



The scatter plot visually reinforces the strong performance of linear regression model. It shows that the model's predictions closely match the actual values, and the linear relationship between predictors and 'satisfaction_level' is captured effectively.

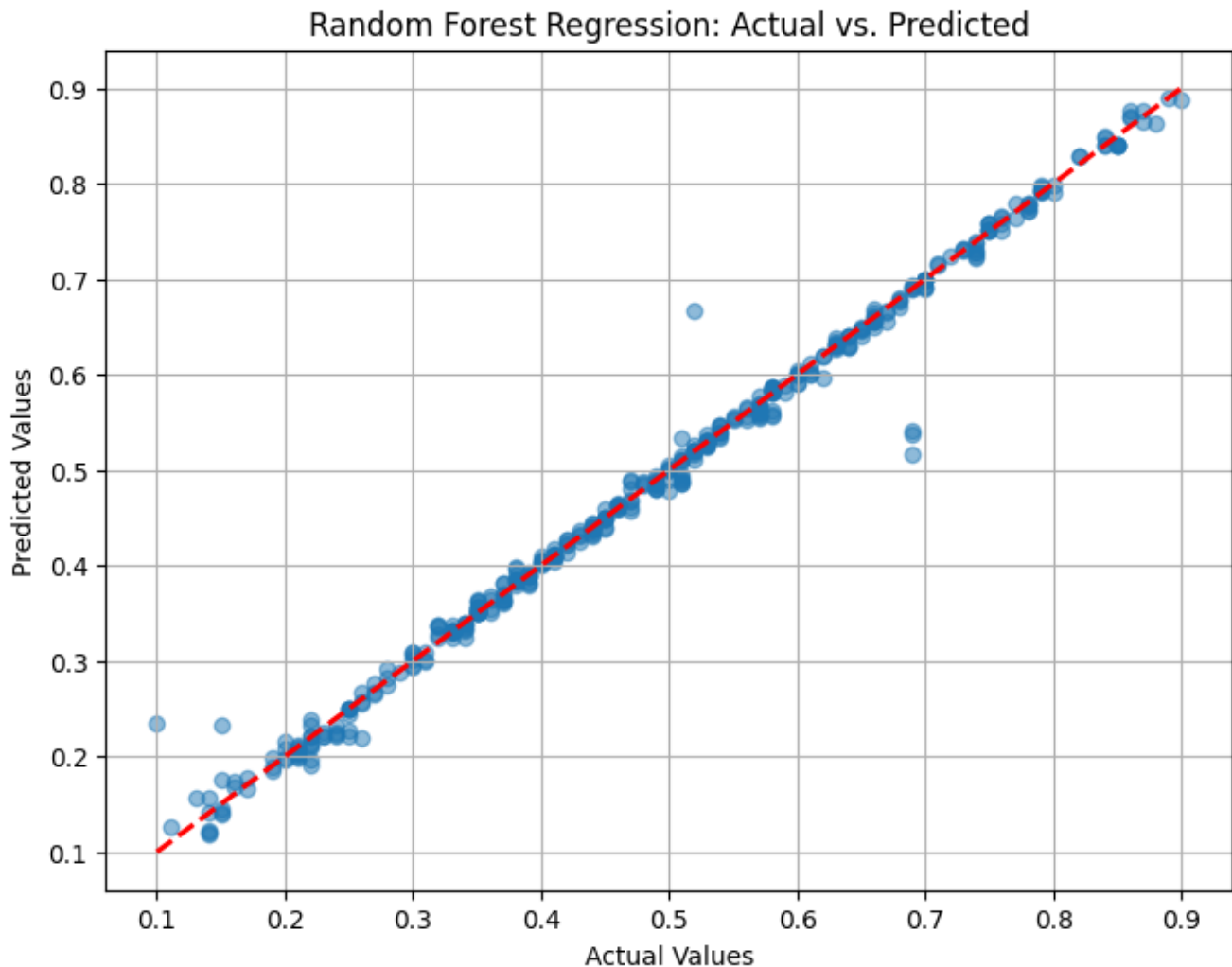
Random Forest Regression

```
X_train, X_test, y_train, y_test = train_test_split(X_reg, y_reg, test_size=0.2,
random_state=42)
# Create and train the Random Forest Regressor
rf_reg = RandomForestRegressor(n_estimators=100, random_state=42)
rf_reg.fit(X_train, y_train)
# Make predictions on the test set
rf_reg_predictions = rf_reg.predict(X_test)
# Evaluate the model
mse = mean_squared_error(y_test, rf_reg_predictions)
r2 = r2_score(y_test, rf_reg_predictions)
# Print the evaluation metrics
print(f"Random Forest Regression Mean Squared Error: {mse:.4f}")
print(f"Random Forest Regression R-squared: {r2:.4f}")
plt.figure(figsize=(8, 6))
plt.scatter(y_test, rf_reg_predictions, alpha=0.5)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',
linestyle='--', lw=2)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Random Forest Regression: Actual vs. Predicted')
plt.grid(True)
# Show the plot
plt.show()

Random Forest Regression Mean Squared Error: 0.0004
Random Forest Regression R-squared: 0.9896
```

Random Forest Regression model performed exceptionally well with a low Mean Squared Error (MSE) of 0.0004, signifying highly accurate predictions. Additionally, the high R-squared (R^2)

value of 0.9896 indicates that approximately 98.96% of the variance in the target variable is explained by the model, highlighting its strong predictive capabilities and ability to capture data patterns.

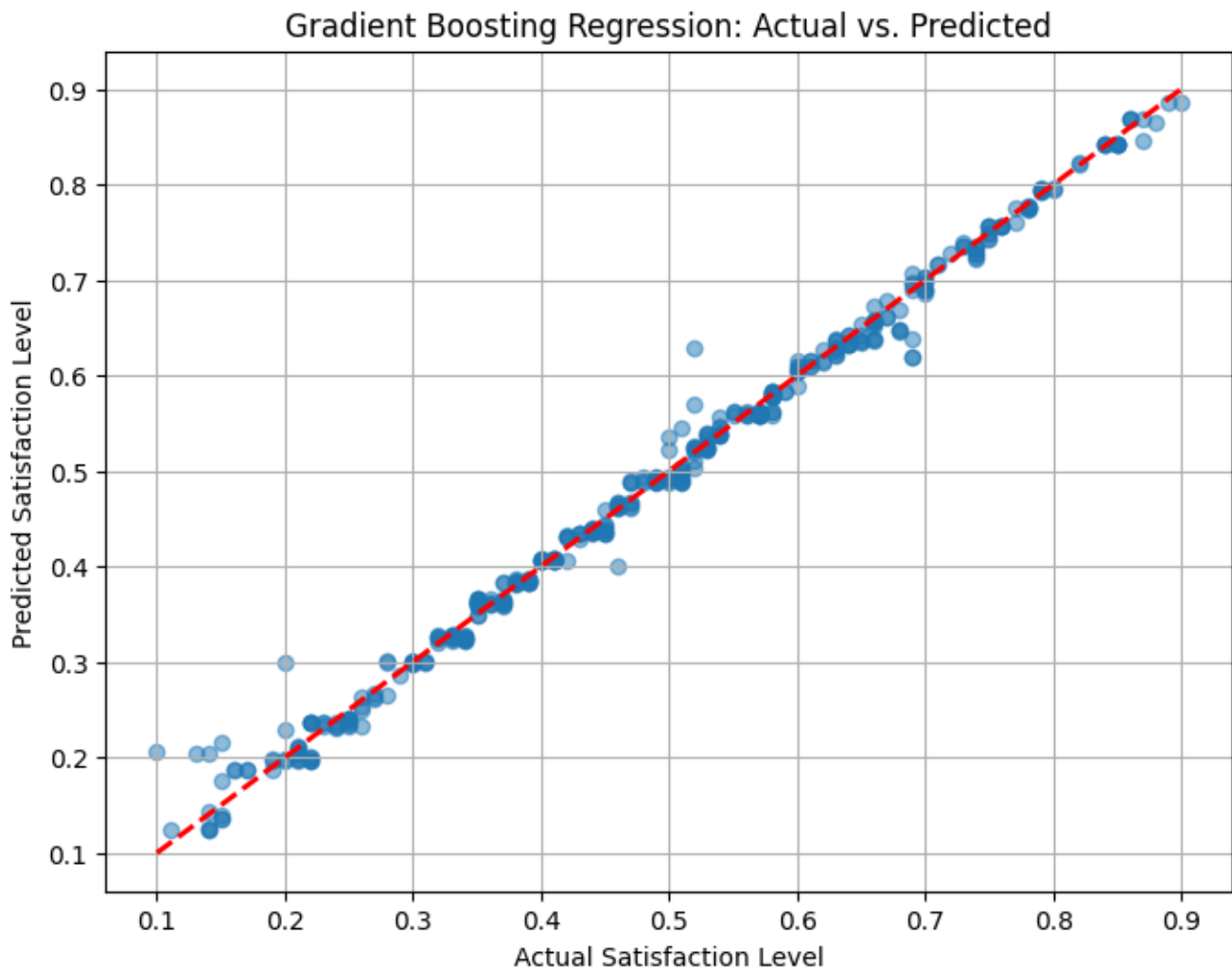


Gradient Boosting Regression

```
X_train, X_test, y_train, y_test = train_test_split(X_reg, y_reg, test_size=0.2,
random_state=42)
gb_reg = GradientBoostingRegressor(n_estimators=100, random_state=42)
gb_reg.fit(X_train, y_train)
# Make predictions on the test set
gb_reg_predictions = gb_reg.predict(X_test)
# Evaluate the model
mse = mean_squared_error(y_test, gb_reg_predictions)
r2 = r2_score(y_test, gb_reg_predictions)
# Print the evaluation metrics
print(f"Gradient Boosting Regression Mean Squared Error: {mse:.4f}")
print(f"Gradient Boosting Regression R-squared: {r2:.4f}")
# Create a scatter plot of actual vs. predicted values
plt.figure(figsize=(8, 6))
plt.scatter(y_test, gb_reg_predictions, alpha=0.5)
```

```
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red',
linestyle='--', lw=2)
plt.xlabel('Actual Satisfaction Level')
plt.ylabel('Predicted Satisfaction Level')
plt.title('Gradient Boosting Regression: Actual vs. Predicted')
plt.grid(True)
# Show the plot
plt.show()
Gradient Boosting Regression Mean Squared Error: 0.0003
Gradient Boosting Regression R-squared: 0.9924
```

Gradient Boosting Regression model excelled with an impressively low Mean Squared Error (MSE) of 0.0003, illustrating precise predictions. Moreover, the high R-squared (R^2) value of 0.9924 indicates that around 99.24% of the variance in the target variable is accounted for by the model, underscoring its exceptional predictive accuracy and ability to capture intricate data patterns.



Conclusion: Among the three regression models, Gradient Boosting Regression is the most suitable choice due to its superior performance. It achieved the lowest Mean Squared Error (MSE) of 0.0003 and the highest R-squared (R^2) value of 0.9924, signifying exceptional

predictive accuracy and the ability to capture complex data relationships. This makes Gradient Boosting Regression the optimal model for accurately predicting employee satisfaction levels in the given dataset.

Categorical Value

Dependent variable as 'left' column

Models used Logistic Regression, Random Forest Classifier and Gradient Boosting Classifier

```
# Convert the 'left' column to a categorical variable
hrm['left'] = hrm['left'].astype('category')
# Split the data into features (X) and the target variable (y)
X = hrm.drop(columns=['left', 'sales', 'promotion_last_5years', 'Training Hours'])
y = hrm['left']
# Split the data into training and testing sets (e.g., 80-20 split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
# 1. Logistic Regression
logistic_reg = LogisticRegression()
logistic_reg.fit(X_train, y_train)
logistic_reg_predictions = logistic_reg.predict(X_test)
logistic_reg_accuracy = accuracy_score(y_test, logistic_reg_predictions)
print("Logistic Regression Accuracy:", logistic_reg_accuracy)
# 2. Random Forest Classifier
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
rf_classifier.fit(X_train, y_train)
rf_predictions = rf_classifier.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)
print("Random Forest Classifier Accuracy:", rf_accuracy)
# 3. Gradient Boosting Classifier
gb_classifier = GradientBoostingClassifier(n_estimators=100, random_state=42)
gb_classifier.fit(X_train, y_train)
gb_predictions = gb_classifier.predict(X_test)
gb_accuracy = accuracy_score(y_test, gb_predictions)
print("Gradient Boosting Classifier Accuracy:", gb_accuracy)
```

```
Logistic Regression Accuracy: 0.8056872037914692
Random Forest Classifier Accuracy: 0.7890995260663507
Gradient Boosting Classifier Accuracy: 0.8127962085308057
```

Among the three classification models, the Gradient Boosting Classifier exhibited the highest accuracy of 81.28%, followed closely by Logistic Regression with an accuracy of 80.57%. Random Forest Classifier had a slightly lower accuracy of 78.91%

Confusion Matrix

```
class_labels = ['Not Left', 'Left']
# Calculate the confusion matrix for Logistic Regression
```

```

logistic_reg_confusion_matrix = confusion_matrix(y_test,
logistic_reg_predictions)
confusion_matrix_formatted = pd.DataFrame(logistic_reg_confusion_matrix,
index=class_labels, columns=class_labels)
print("Logistic Regression Confusion Matrix:")
print(confusion_matrix_formatted)
# Calculate the confusion matrix for Random Forest Classifier
rf_confusion_matrix = confusion_matrix(y_test, rf_predictions)
print("Random Forest Classifier Confusion Matrix:")
rf_matrix_formatted = pd.DataFrame(rf_confusion_matrix, index=class_labels,
columns=class_labels)
print(rf_matrix_formatted)
# Calculate the confusion matrix for Gradient Boosting Classifier
gb_confusion_matrix = confusion_matrix(y_test, gb_predictions)
gb_matrix_formatted = pd.DataFrame(gb_confusion_matrix, index=class_labels,
columns=class_labels)
print("Gradient Boosting Classifier Confusion Matrix:")
print(gb_matrix_formatted)

```

```

Logistic Regression Confusion Matrix:
      Not Left  Left
Not Left    179    54
Left         28   161
Random Forest Classifier Confusion Matrix:
      Not Left  Left
Not Left    176    57
Left         32   157
Gradient Boosting Classifier Confusion Matrix:
      Not Left  Left
Not Left    167    66
Left         13   176

```

Gradient Boosting Classifier outperformed both Logistic Regression and Random Forest Classifier in terms of its confusion matrix. It had the highest number of correct predictions for both "Left" and "Not Left" classes, with only 13 instances of misclassification in the "Left" category. Logistic Regression had slightly lower performance, while Random Forest Classifier exhibited the most misclassifications in the "Left" category.

Conclusion: Gradient Boosting Classifier achieved the highest accuracy score (0.8128), indicating that it correctly classified a higher percentage of employees compared to the other models. Looking at the confusion matrix for the Gradient Boosting Classifier, it has a relatively low number of false negatives (13), which means it better identifies employees who left (Left) compared to the Logistic Regression and Random Forest models. Additionally, it has a higher true negative count (167) compared to Logistic Regression, indicating better accuracy in predicting employees who did not leave (Not Left). While Logistic Regression also performs well, the Gradient Boosting Classifier strikes a balance between precision and recall, making it suitable for scenarios where both correct identification of employees who left and those who did not is important.

Overall Conclusion:

In conclusion, the analysis conducted in this project sheds light on crucial aspects of employee satisfaction and attrition using HR analytics techniques. Through thorough exploration of a comprehensive dataset comprising various employee attributes and work-related factors, valuable insights have been unearthed.

The findings reveal significant correlations between factors such as satisfaction level, number of projects, salary, and employee turnover. It is evident that employee satisfaction plays a pivotal role in influencing attrition rates, with variables like workload, salary, and promotions impacting overall job satisfaction.

Moreover, the application of machine learning models has proven to be instrumental in predicting employee satisfaction levels and identifying potential churn. Among these models, Gradient Boosting Regression emerges as the most effective in predicting satisfaction levels, while Gradient Boosting Classifier exhibits superior performance in predicting employee attrition.

These predictive models, coupled with insightful visualizations and exploratory analyses, offer actionable insights for HR departments and organizational management. By leveraging these insights, companies can implement proactive measures to enhance employee retention, improve organizational performance, and mitigate the costs associated with turnover.

In essence, this project underscores the pivotal role of data-driven decision-making in human resource management. By harnessing the power of HR analytics, organizations can cultivate a more engaged workforce, foster a positive work environment, and ultimately drive sustainable growth and success.

This project serves as a testament to the transformative potential of analytics in shaping the future of work, where informed decisions pave the way for a more resilient, productive, and satisfied workforce.