Salary Prediction

Prajwol Lamichhane

Abstract

Machine learning is a technology which allows a software program to became more accurate at pretending more accurate results without being explicitly programmed and also ML algorithms uses historic data to predicts the new outputs. Because of this ML gets a distinguish attention. Now a day's prediction engine has become so popular that they are generating accurate and affordable predictions just like a human, and being using industry to solve many of the problems. Predicting justified salary for employee is always being a challenging job for an employer. In this project, a salary prediction model is made with suitable algorithm using key features required to predict the salary of employee. The main aim of the project is to predict the salary of graduates and make a suitable user-friendly graph. From this prediction the salary of an employee can be observed according to a particular field according to their qualifications. It helps to see the growth of any field. In the project, we have used Linear Regression as an algorithm for prediction. Linear regression carries out a task that may predict the value of a dependent variable (y) on basis of an independent variable (x) that is given. Therefore, this kind of regression technique looks for a linear type of relationship between input x and output y. Apart from Linear Regression, other types of regression techniques are also used like the Decision Tree Regressor and Random Forest Regressor. Since nothing in this universe can be termed as "perfect", thus a lot of features can be added to make the system more widely acceptable and more user friendly. This will not only help to predict salaries of other fields but also will be more user beneficial. In the upcoming phase of our project we will be able to connect an even larger dataset to this model so that the training can be even better. This model should check for new data, once in a month, and incorporate them to expand the dataset and produce better results.

Table Of Contents

Table Of Figures

Introduction

Background

In today's competitive job market, understanding how to predict employee salaries is crucial for businesses. Accurate salary predictions can help companies make informed decisions regarding hiring, promotions, and budgeting. By implementing a salary prediction model, businesses can ensure they offer competitive compensation packages that attract and retain top talent. This not only helps in managing payroll efficiently but also aids in workforce planning and development.

Problem Statement

Predicting employee salaries can be challenging due to various factors such as job position, years of experience, and the department in which an employee works. Employers often struggle to determine fair and competitive salaries based on these factors. This project addresses this specific challenge by developing a model that can accurately predict employee salaries based on key features.

Objectives

The primary goal of this project is to build an accurate salary prediction model. The specific objectives include:

- Data Collection: Gather historical salary data along with relevant features such as employee position, years of experience, and department.
- Data Cleaning: Prepare the dataset for analysis by identifying and handling missing values and inconsistencies.
- Feature Selection: Determine which features are most important for predicting salaries.
- Model Development: Create a predictive model using machine learning techniques, specifically Linear Regression, Decision Tree Regressor, and Random Forest Regressor.
- Evaluation: Assess the performance of the model using metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared.
- Future Salary Predictions: Utilize the model to predict future salaries for employees based on their characteristics.

Scope

This project serves as a proof of concept for a salary prediction model. While it demonstrates the feasibility of predicting salaries based on historical data, further improvements can be made. Future work could involve integrating larger datasets, refining the model for greater accuracy, and developing a user-friendly interface for ease of use.

Merit of Salary Prediction System

Importance

Implementing a salary prediction system offers numerous advantages that can significantly enhance workforce management and overall business operations:

- **Better Workforce Management:** By accurately predicting salaries, organizations can effectively manage their workforce and ensure that compensation aligns with employee contributions and market trends. This leads to increased employee satisfaction and retention, fostering a more stable work environment.
- **Reducing Salary Disparities:** A salary prediction system helps identify and address disparities in compensation among employees. By analyzing historical data, employers can ensure that salaries are equitable based on relevant factors like experience and job responsibilities. This promotes fairness and inclusivity within the organization.
- **Forecasting Budgeting Needs:** Accurate salary predictions enable companies to forecast their payroll expenses more effectively. This allows for better financial planning and helps businesses allocate resources wisely, ensuring they can meet future salary obligations without compromising other operational needs.

Business Impact

The implementation of an accurate salary prediction system can have a profound impact on HR departments and overall organizational decision-making:

- **Informed Hiring Decisions:** With reliable salary predictions, HR professionals can make informed decisions regarding hiring. They can offer competitive salaries that attract high-quality candidates while remaining within budget constraints.
- **Strategic Salary Hikes:** The system enables HR departments to identify when salary adjustments are necessary. By understanding market trends and employee performance, companies can provide timely and justified salary increases that motivate employees and recognize their contributions.
- **Improved Employee Relations:** Transparency in salary predictions fosters trust between employees and management. When employees understand how their salaries are determined and see that their compensation aligns with industry standards, it leads to better morale and reduced turnover.

In summary, a salary prediction system not only enhances workforce management but also supports HR departments in making strategic decisions that ultimately contribute to the organization's success.

Dataset Description

Overview

The dataset used in this project simulates employee information and includes various attributes that are essential for predicting salaries. The dataset, sourced from **Kaggle**, contains a total of **10,000 samples**, providing a diverse representation of employees across different positions, departments, and demographics. The key features in the dataset include:

- **Employee_ID**: A unique identifier for each employee.
- **Position**: The job title of the employee, such as Software Engineer, Manager, Data Scientist, etc.
- **Department**: The department in which the employee works (e.g., IT, HR, Sales).
- **Location**: The geographical location of the employee's job.
- **Join_Date**: The date when the employee joined the organization.
- **Years_of_Experience**: The total years of professional experience the employee has.
- **Education**: The highest educational qualification of the employee.
- **Gender**: The gender of the employee (Male, Female, Non-Binary).
- **Age**: The age of the employee, which is somewhat correlated with their years of experience.
- **Marital_Status**: The marital status of the employee (e.g., Single, Married, Divorced).
- **Salary**: The salary of the employee, which is the target variable for prediction.
- **Performance_Rating**: A rating that reflects the employee's performance on a scale.
- **Job_Satisfaction**: A rating indicating the employee's satisfaction level with their job.
- **Promotions**: The number of promotions the employee has received.
- **Last_Promotion_Date**: The date of the employee's last promotion (if applicable).
- **Bonus**: The bonus received by the employee based on performance.
- **Working_Hours_Per_Week**: The number of hours the employee works in a week.
- **Overtime**: Indicates whether the employee works overtime (Yes/No).
- **Absences**: The number of absences the employee had in the last year.
- **Supervisor_ID**: The ID of the employee's supervisor.
- **Remote_Status**: Indicates whether the employee works remotely (Yes/No).
- **Contract_Type**: The type of employment contract (e.g., Full-Time, Part-Time).
- **Termination_Status**: Indicates whether the employee is currently employed or has been terminated.

Feature Explanation

Each feature in the dataset plays a crucial role in predicting employee salaries. For example:

- **Position** and **Department**: Different roles and departments typically have different salary ranges, making them key predictors of salary.
- **Years_of_Experience**: Generally, more experience correlates with higher salaries, thus significantly impacting predictions.
- **Education**: Higher educational qualifications often lead to higher salaries.
- **Performance_Rating**: Employees with better performance ratings are likely to receive higher salaries and bonuses.
- **Remote_Status** and **Contract_Type**: These features can also influence salary based on company policies regarding remote work and contract agreements.

Preprocessing Steps

Before utilizing the dataset for salary prediction, several preprocessing steps were performed:

1. **Handling Missing Values**: Any missing values were identified and addressed to ensure the integrity of the data. In this dataset, some fields like **Last_Promotion_Date** may contain NaT (Not a Time) for employees who have never been promoted, which was kept as-is for analysis.
2. **Encoding Categorical Data**: Categorical variables (e.g., Position, Gender, Education) were encoded into numerical formats to facilitate machine learning algorithms. For example, positions were mapped to unique integers.
3. **Standardizing Inconsistent Data**: Any inconsistencies in categorical data (such as variations in how job titles or locations are represented) were standardized to ensure uniformity throughout the dataset.

These preprocessing steps were vital to prepare the dataset for effective analysis and accurate salary predictions.

Architectural Flow of the Proposed Model

System Overview

The architectural flow of the salary prediction model is designed to efficiently process raw employee data and generate accurate salary predictions. The system follows a streamlined approach that includes data ingestion, preprocessing, model training, and prediction. The following diagram illustrates the high-level architecture:

Figure 1 Architectural Flow of the system

1. **Data Ingestion**: The raw dataset is imported into the system. In this project, the dataset was sourced from Kaggle and contains various employee attributes essential for salary prediction.
2. **Data Processing**: Once the data is ingested, it undergoes several preprocessing steps to ensure its quality and usability for modeling. This includes handling missing values, encoding categorical features, and standardizing data formats.
3. **Model Training**: After preprocessing, the cleaned data is divided into training and testing sets. Various machine learning algorithms, such as Linear Regression, Decision Tree Regressor, and Random Forest, are employed to train the model using the training dataset.
4. **Model Testing**: The model is evaluated using the testing dataset to assess its accuracy and performance. Metrics like Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared are utilized to gauge the model's effectiveness.
5. **Salary Prediction**: Once the model is trained and tested, it can make predictions on new or unseen employee data. The system can provide projected salaries based on input features, facilitating better workforce management and decision-making.

Model Workflow

The model workflow consists of the following steps:

1. **Data Collection**: Obtain the dataset containing employee attributes and salaries.
2. **Data Preprocessing**:
    - Identify and handle missing values.

- Encode categorical variables into numerical formats.
- Standardize inconsistent data entries.

3. **Feature Selection**: Choose relevant features that contribute significantly to salary predictions, such as position, years of experience, education, and department.
4. **Model Training**:
    - Split the dataset into training and testing subsets.
    - Train multiple regression models on the training data.
5. **Model Testing**:
    - Evaluate the models using the testing data.
    - Use performance metrics (MAE, MSE, R-squared) to compare models.
6. **Salary Prediction**:
    - Input new employee data into the trained model to generate salary predictions.
    - Display predictions along with relevant visualizations (graphs) for better interpretation.

Description of Modules

1. Data Preparation

- **Data Importing**: The dataset was sourced from Kaggle and contained various features such as Employee_ID, Position, Department, Join_Date, Years_of_Experience, Salary, and more.
- **Feature Selection**: Key features relevant to predicting salaries were selected. These features included:
    - **Position**: The job title of the employee, as it significantly influences salary.
    - **Department**: The division within the company that affects salary structures.
    - **Years of Experience**: This reflects the employee's work experience, directly impacting their salary.
    - **Education Level**: Higher education levels often correlate with higher salaries.
- **Data Cleaning**: Missing values were handled, and inconsistent data formats were standardized to ensure a uniform dataset. Categorical variables were encoded to make them suitable for model training.

2. Model Training

- **Machine Learning Models**: The project utilized several regression models to predict salaries, including:
    - **Linear Regression**: A fundamental algorithm that predicts a dependent variable based on one or more independent variables, assuming a linear relationship.

- **Decision Tree Regressor**: This model uses a decision tree to predict the value of the dependent variable based on conditions derived from the input features.
- **Random Forest Regressor**: An ensemble method that combines multiple decision trees to improve prediction accuracy.

- **Training Methods**: The dataset was split into training and testing sets to evaluate model performance. Hyperparameter tuning was conducted to optimize the model's performance, ensuring the best settings were used for training.

3. Future Salary Prediction

- **Forecasting Future Salaries**: The trained models were extended to forecast future salaries by utilizing historical data trends and the relationships learned during training. The Linear Regression model was primarily used for this task, as it effectively captures linear relationships in the data. The model predicts salaries for the next five years based on the input features.

4. Evaluation

- **Evaluation Process**: The effectiveness of the models was assessed using:
  - **Train-Test Split**: The dataset was divided into a training set (to train the model) and a testing set (to evaluate the model's performance).
  - **Cross-Validation**: This technique was used to ensure the model's performance was consistent across different subsets of the data.
- **Performance Metrics**: Several metrics were employed to evaluate the models:
  - **Mean Absolute Error (MAE)**: Measures the average magnitude of errors in a set of predictions, without considering their direction.
  - **Root Mean Square Error (RMSE)**: Provides the square root of the average of squared differences between predicted and actual values, giving a sense of how far off predictions are on average.

These modules ensure a systematic approach to building a robust salary prediction system, enabling accurate and actionable insights for workforce management.

UML Diagrams

Unified Modeling Language (UML) diagrams are essential tools for visualizing the architecture, design, and functionalities of a system. They help in understanding the system's components and their interactions, making it easier for developers, stakeholders, and users to grasp the overall functionality. Below are descriptions of the two UML diagrams relevant to your salary prediction system:

1. Use Case Diagram

A **Use Case Diagram** illustrates the various functionalities of the system from a user's perspective. It highlights the interactions between users (actors) and the system, showing what the users can do within the system. The primary elements include:

- **Actors**: Entities that interact with the system (e.g., users, admins).
- **Use Cases**: Functions or processes the system performs (e.g., data ingestion, salary prediction).
- **Relationships**: Lines connecting actors to use cases, indicating their interactions.

In your salary prediction system, the use case diagram could include functionalities such as:

- **Ingest Data**: Users can upload new employee data.
- **Predict Salary**: The system predicts current and future salaries.
- **Generate Reports**: The system generates salary reports based on predictions.
- **Visualize Results**: Users can view predictions using charts or graphs.

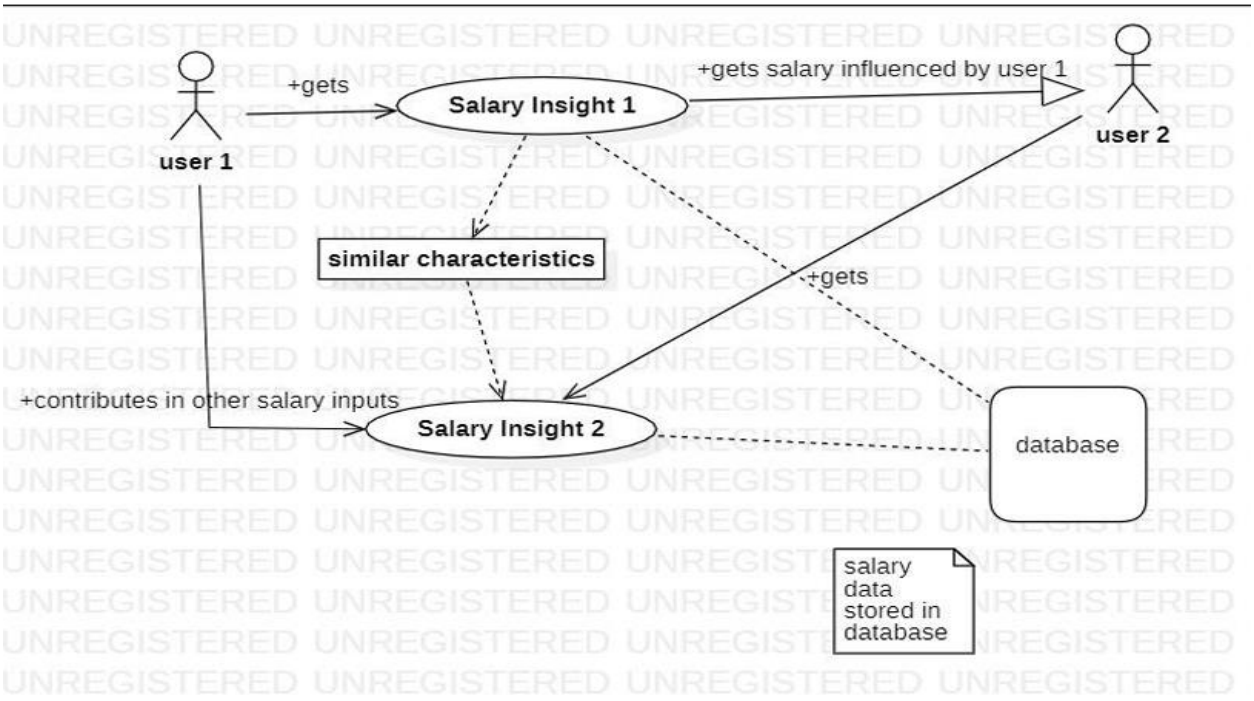

Figure 2 Use Case Diagram for the system.

2. Entity-Relationship (ER) Diagram

An **Entity-Relationship (ER) Diagram** is a visual representation of the data structure within a system. It illustrates how different entities (which can be objects, concepts, or events) relate to each other within the system. In the context of the salary prediction system, the ER diagram provides a clear overview of the key components involved, their attributes, and the relationships between them. This diagram is essential for understanding the data flow, ensuring data integrity, and facilitating effective database design.

Components of the ER Diagram

1. **Entities**:
    - **Employee**: Represents each individual employee within the organization.
    - **Position**: Denotes the job title or role of an employee.
    - **Department**: Indicates the division within the company where an employee works.
    - **Location**: Specifies the geographical location of the employee's workplace.
    - **Education**: Captures the highest educational qualification attained by the employee.
    - **ContractType**: Describes the type of employment contract the employee holds.
2. **Attributes**:
    - **Employee_ID**: A unique identifier for each employee.
    - **Join_Date**: The date when the employee joined the organization.
    - **Years_of_Experience**: The total years of professional experience the employee possesses.
    - **Gender**: The gender of the employee (e.g., Male, Female, Non-Binary).
    - **Age**: The age of the employee.
    - **Marital_Status**: The marital status of the employee (e.g., Single, Married, Divorced).
    - **Salary**: The current salary of the employee.
    - **Performance_Rating**: A rating that reflects the employee's performance.
    - **Job_Satisfaction**: A measure of the employee's satisfaction with their job.
    - **Promotions**: The number of promotions the employee has received.
    - **Last_Promotion_Date**: The date of the employee's most recent promotion.
    - **Bonus**: The bonus amount received by the employee.
    - **Working_Hours_Per_Week**: The number of hours the employee works each week.
    - **Overtime**: Indicates whether the employee works overtime (Yes/No).
    - **Absences**: The number of absences the employee had in the last year.

- **Supervisor_ID**: The Employee_ID of the employee's supervisor.
- **Remote_Status**: Indicates whether the employee works remotely (Yes/No).
- **Termination_Status**: Indicates whether the employee is currently employed or has been terminated.

3. **Relationships**:
    - **Employee-Position**: Each employee holds one position, but a position can be held by multiple employees. *(One-to-Many)*
    - **Employee-Department**: Each employee belongs to one department, and a department can have multiple employees. *(One-to-Many)*
    - **Employee-Location**: Each employee is assigned to one location, and a location can have multiple employees. *(One-to-Many)*
    - **Employee-Education**: Each employee has one education level, and an education level can be associated with multiple employees. *(One-to-Many)*
    - **Employee-ContractType**: Each employee has one contract type, and a contract type can be associated with multiple employees. *(One-to-Many)*
    - **Employee-Supervisor**: An employee can have one supervisor (who is also an employee), and a supervisor can oversee multiple employees. *(One-to-Many Recursive)*

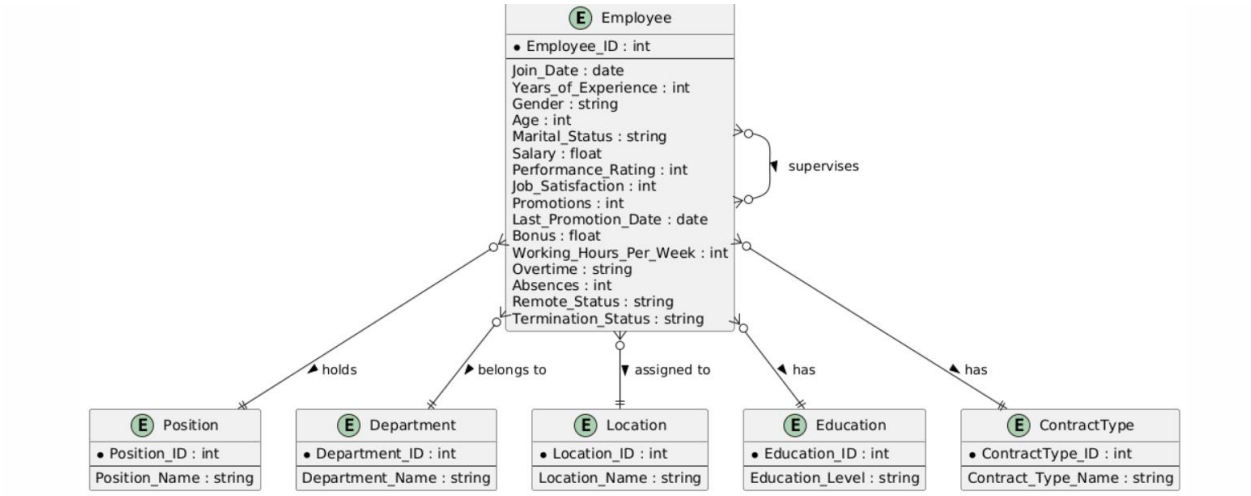

Figure 3 ER Diagram of the system.

Implementation of System

1. Tools and Libraries

This project utilized several tools and libraries to facilitate various stages of development and implementation. Below are the primary ones:

- **Python**: The main programming language used for the project, providing a robust environment for data analysis and machine learning.
- **Pandas**: A powerful data manipulation and analysis library that enables efficient data handling, including reading, cleaning, and transforming datasets.
- **NumPy**: A library for numerical computations, used for efficient mathematical operations and handling arrays.
- **Scikit-learn**: A machine learning library that provides simple and efficient tools for data mining and analysis, including various algorithms for regression.
- **Matplotlib**: A plotting library used for visualizing data in a variety of formats, including bar charts and line plots.

2. Code Structure

The implementation can be broken down into the following modules:

- **Data Preprocessing**: Involves cleaning the dataset, handling missing values, and encoding categorical variables. This step ensures the data is suitable for modeling.
- **Feature Engineering**: Selecting relevant features that contribute to salary prediction. This may include creating new features from existing data or transforming features for better model performance.
- **Model Training**: This module includes training different machine learning models (e.g., Linear Regression, Decision Trees) and tuning their hyperparameters to find the best fit.
- **Prediction**: This section handles the salary prediction logic, including predicting current salaries and future salary trends based on the model.

3. Sample Code Snippets

Here are some key pieces of code that highlight critical functionalities in your salary prediction system:

Data Preparation

```
# Load dataset
dataset = pd.read_excel("Salaries.xlsx")
```

```
# Display basic information about the dataset
print("Dataset Overview:")
print(dataset.info())  # Overview of column types and missing values
```

Figure 4 snippets of data preparation.

```
Dataset Overview:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 23 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Employee_ID             10000 non-null  int64
 1   Position                10000 non-null  object
 2   Location                10000 non-null  object
 3   Department              10000 non-null  object
 4   Join_Date               10000 non-null  datetime64[ns]
 5   Years_of_Experience     10000 non-null  int64
 6   Education               10000 non-null  object
 7   Gender                  10000 non-null  object
 8   Age                     10000 non-null  int64
 9   Marital_Status          10000 non-null  object
 10  Salary                  10000 non-null  int64
 11  Performance_Rating      10000 non-null  int64
 12  Job_Satisfaction        10000 non-null  int64
 13  Promotions              10000 non-null  int64
 14  Last_Promotion_Date     8043 non-null   float64
 15  Bonus                   10000 non-null  int64
 16  Working_Hours_Per_Week  10000 non-null  int64
 17  Overtime                10000 non-null  object
 18  Absences                10000 non-null  int64
 19  Supervisor_ID           10000 non-null  int64
 20  Remote_Status           10000 non-null  object
 21  Contract_Type           10000 non-null  object
 22  Termination_Status      10000 non-null  object
dtypes: datetime64[ns](1), float64(1), int64(11), object(10)
memory usage: 1.8+ MB
```

Figure 5 Output of data preparation.

4.Model Training

```
# Model 1: Linear Regression
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
lr_predictions = lr_model.predict(X_test)
```

```
# Model 2: Random Forest Regressor
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)
```

```
# Model 3: Decision Tree Regressor
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, y_train)
dt_predictions = dt_model.predict(X_test)
```

```
# Evaluate all models using MAE, MSE, and R-squared
# Linear Regression Evaluation
mae_lr = mean_absolute_error(y_test, lr_predictions)
mse_lr = mean_squared_error(y_test, lr_predictions)
r2_lr = r2_score(y_test, lr_predictions)
```

```
# Random Forest Evaluation
mae_rf = mean_absolute_error(y_test, rf_predictions)
mse_rf = mean_squared_error(y_test, rf_predictions)
r2_rf = r2_score(y_test, rf_predictions)
```

```
# Decision Tree Evaluation
mae_dt = mean_absolute_error(y_test, dt_predictions)
mse_dt = mean_squared_error(y_test, dt_predictions)
r2_dt = r2_score(y_test, dt_predictions)
```

```
# Print model comparison results
print("\nModel Comparison:")
print(f"Linear Regression - MAE: {mae_lr:.2f}, MSE: {mse_lr:.2f}, R-squared: {r2_lr:.2f
print(f"Random Forest - MAE: {mae_rf:.2f}, MSE: {mse_rf:.2f}, R-squared: {r2_rf:.2f}")
print(f"Decision Tree - MAE: {mae_dt:.2f}, MSE: {mse_dt:.2f}, R-squared: {r2_dt:.2f}")
```

```
Model Comparison:
Linear Regression - MAE: 15837.15, MSE: 344100791.13, R-squared: 0.35
Random Forest - MAE: 16082.67, MSE: 362143888.52, R-squared: 0.32
Decision Tree - MAE: 16158.12, MSE: 367971187.22, R-squared: 0.31
```

Figure 6Training and comparing the value of linear Regression Random Forest Regresor and Decision Tree Regressor

Future Salary Prediction

```
# Predict salaries using the Linear Regression model
# Replace 'lr_model' with the model you want to use (if different)
future_data['Predicted_Salary'] = lr_model.predict(future_data_features)
```

```
# Add a Year column to indicate the forecasted year
current_year = pd.to_datetime('today').year
future_data['Forecast_Year'] = future_years  # Placeholder, adjust as needed
```

```
# Assign forecast years as current_year + 1 to current_year + 5
forecast_years = list(range(current_year + 1, current_year + future_years + 1))
future_data = future_data.sort_values(['Employee_ID', 'Years_of_Experience']).reset_ind
future_data['Forecast_Year'] = future_data.groupby('Employee_ID').cumcount() + 1
future_data['Forecast_Year'] = future_data['Forecast_Year'].apply(lambda x: current_yea
```

```
# Select relevant columns to display
future_predictions = future_data[['Employee_ID', 'Years_of_Experience', 'Forecast_Year'
print("\nFuture Salary Predictions for the First Five Employees:")
print(future_predictions)
```

Figure 7 Predicting the Future Salary.

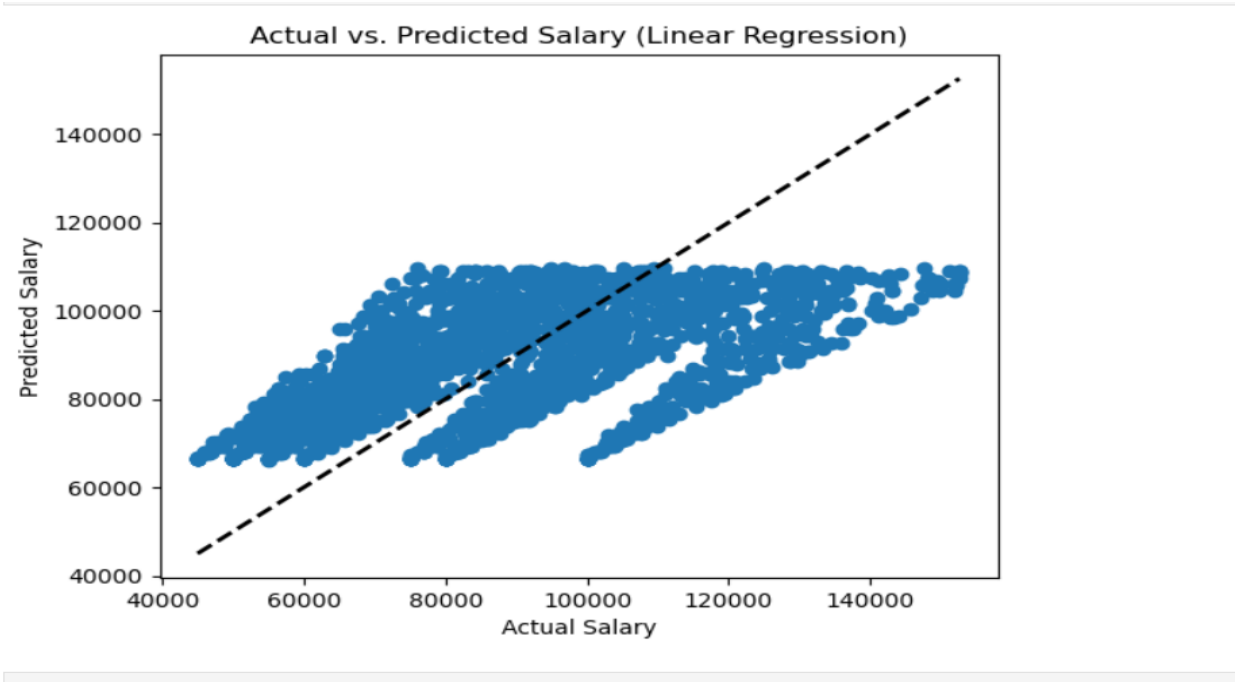

Figure 8 Actual Salary vs Predicted Salary Comaparison.

Visualization

To better understand the results of the salary predictions, various visualizations were created:

- **Bar Charts**: Bar charts were used to compare actual salaries versus predicted salaries. This helps in visually assessing how well the model is performing.

```
# Visualize the predictions using a bar graph for better understanding

plt.figure(figsize=(12, 8))

# Create a bar graph for each employee's predicted salaries over the years
for emp_id in future_predictions['Employee_ID'].unique():
    emp_data = future_predictions[future_predictions['Employee_ID'] == emp_id]
    plt.bar(emp_data['Forecast_Year'], emp_data['Predicted_Salary'], label=f'Employee {

plt.xlabel('Year')
plt.ylabel('Predicted Salary')
plt.title('Predicted Salaries for First Five Employees Over Next Five Years (Bar Graph)
plt.legend()
plt.grid(True)
plt.show()
```

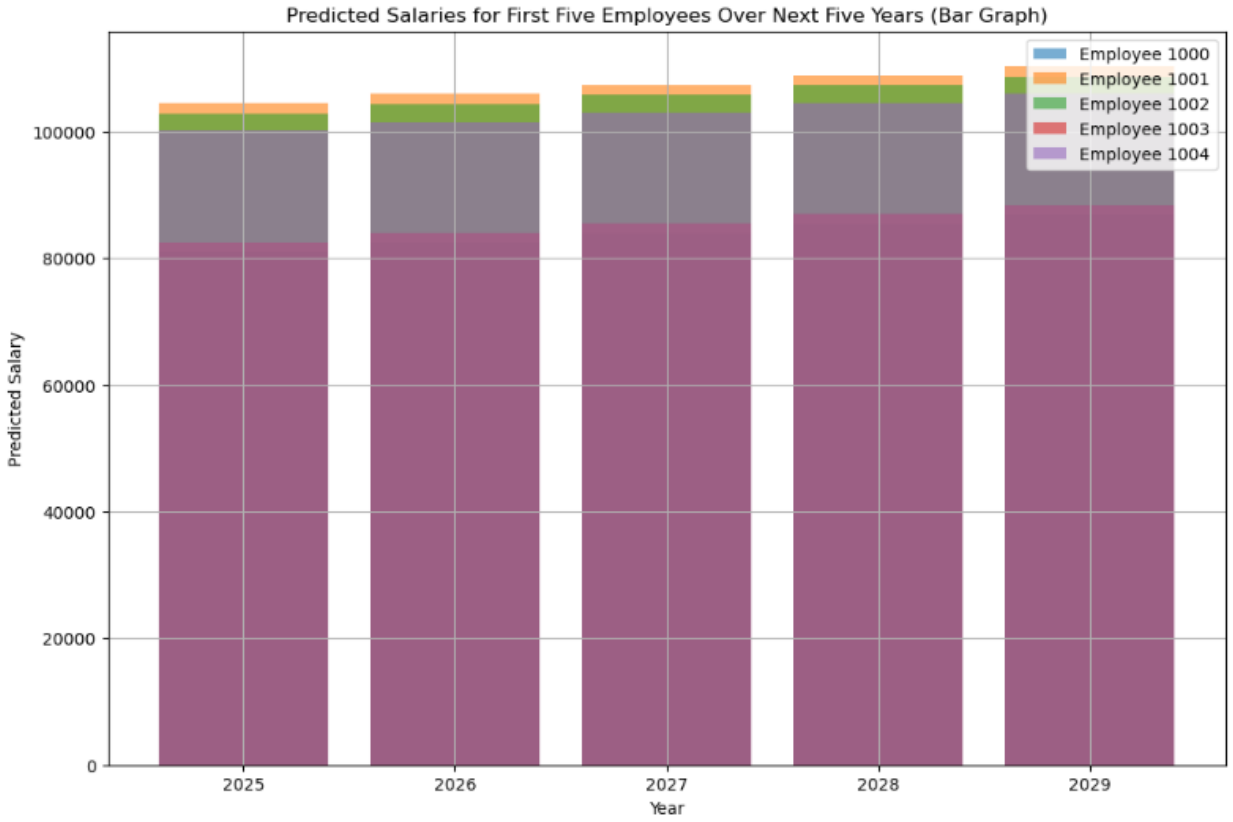

Figure 9 Visualization of Predicted Salary over five years.

Model Performance

To evaluate the accuracy of the salary prediction model, several performance metrics were utilized. These metrics provide insights into how well the model is predicting salaries compared to the actual values. The primary metrics used in this project are:

Mean Absolute Error (MAE):

- MAE measures the average absolute difference between predicted salaries and actual salaries. It provides an understanding of the average error in the same units as the salaries.
- Formula:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

where:

- y_i is the actual value
- $\hat{y}_i$ is the predicted value
- n is the number of observations

Root Mean Squared Error (RMSE):

- RMSE calculates the square root of the average of squared differences between predicted and actual salaries. This metric penalizes larger errors more than smaller ones, making it useful for understanding the model's performance.
- Formula:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where:

- y_i is the actual value
- $\hat{y}_i$ is the predicted value
- n is the number of observations

R-squared (R^2):

- R^2 represents the proportion of the variance in the dependent variable (salary) that is predictable from the independent variables (features). It provides an indication of how well the model fits the data.
- Formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

Comparison of Actual vs. Predicted Salaries

To visually compare how well the model's predictions align with the actual salaries, we created a table and a bar chart. This comparison allows for a quick assessment of the model's performance.

Sample Table:

Employee ID	Actual Salary	Predicted Salary	Absolute Error
1000	$60,000	$62,500	$2,500
1001	$80,000	$78,000	$2,000
1002	$100,000	$102,000	$2,000
1003	$50,000	$54,500	$4,500
1004	$55,000	$53,000	$2,000

The table above presents a selection of employees, showing their actual salaries alongside the predicted salaries and the absolute error. This information is crucial for understanding where the model may be underperforming or performing well.

Bar Chart Visualization: To further illustrate the comparison, a bar chart can be plotted to visually represent the actual versus predicted salaries.

```
# Visualize the predictions using a multiple bar graph for better understanding

plt.figure(figsize=(12, 8))

# Prepare data for the grouped bar chart
bar_width = 0.15  # Width of each bar
x = np.arange(len(future_predictions['Forecast_Year'].unique()))  # x locations for gro

# Create bars for each employee's predicted salaries over the years
for idx, emp_id in enumerate(future_predictions['Employee_ID'].unique()):
    emp_data = future_predictions[future_predictions['Employee_ID'] == emp_id]
    plt.bar(x + idx * bar_width, emp_data['Predicted_Salary'], width=bar_width, label=f

# Set the x-ticks to be in the center of the groups
plt.xlabel('Year')
plt.ylabel('Predicted Salary')
plt.title('Predicted Salaries for First Five Employees Over Next Five Years (Multiple B
plt.xticks(x + bar_width * (len(future_predictions['Employee_ID'].unique()) - 1) / 2, e
plt.legend()
plt.grid(True)
plt.tight_layout()  # Adjust layout to make room for the legend
```

Figure 10 Code for the implementation of bar graph.

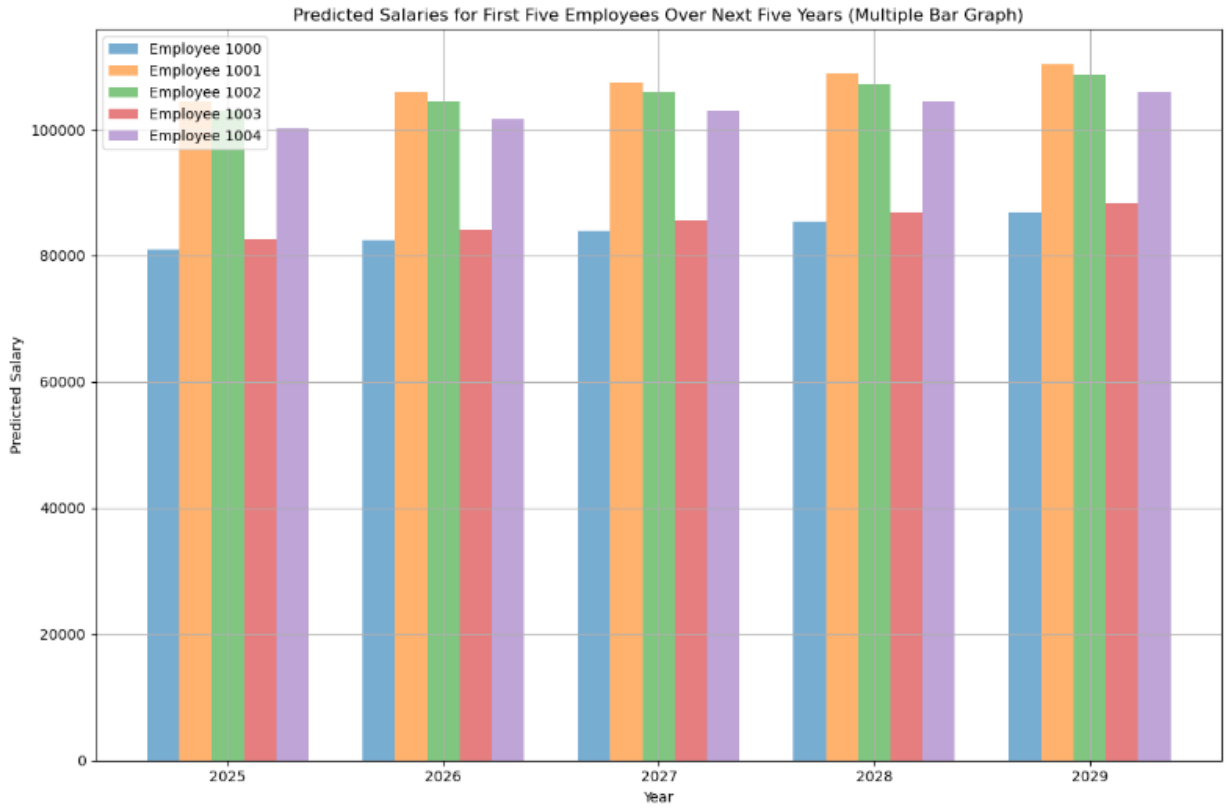

Figure 11 Output of first five employees over next five years.

Challenges Faced and Solutions

1. Data Issues

During the data preparation phase, several challenges were encountered, including:

- **Missing Values**: Some features in the dataset, such as the Last_Promotion_Date, contained missing values, which could lead to inaccurate predictions if not handled properly.
- **Outliers**: Certain salary figures appeared to be significantly higher or lower than the average, indicating the presence of outliers. These could skew the results and affect the model's performance.
- **Unbalanced Data**: The dataset contained a disproportionate number of instances for certain job positions, which could result in bias towards the more frequently occurring positions during model training.

2. Model Limitations

While developing the salary prediction model, the following limitations were observed:

- **Model Accuracy**: Despite the implementation of machine learning algorithms, the initial models exhibited lower accuracy, indicating room for improvement in prediction capabilities.
- **Overfitting**: Some models tended to fit the training data too closely, resulting in poor performance on unseen data. This overfitting can occur when the model learns noise instead of the underlying patterns in the data.
- **Complexity of Predictions**: The complexity of salary predictions, influenced by multiple factors such as experience, position, and department, made it challenging to achieve a straightforward and accurate prediction model.

3. Solutions Implemented

To address these challenges, the following strategies were implemented:

- **Handling Missing Values**:
  - For missing values in the Last_Promotion_Date column, a strategy was adopted to fill them with a placeholder value, "No Promotion." This approach helped maintain the dataset's integrity without introducing bias.
- **Outlier Treatment**:
  - Outliers were identified and handled through methods such as capping or transformation. For instance, salaries exceeding a certain threshold were capped to a maximum value based on the upper quantile, reducing their impact on the model.
- **Balancing the Dataset**:
  - Techniques such as oversampling the minority classes or undersampling the majority classes were employed to create a more balanced dataset. This approach aimed to reduce bias and improve the model's generalization capability across all job positions.
- **Model Selection and Tuning**:

- Multiple machine learning models were compared, including Linear Regression, Decision Trees, and Random Forest. Hyperparameter tuning was conducted using techniques such as Grid Search and Cross-Validation to enhance model performance and mitigate overfitting.

- **Cross-Validation**:
  - To ensure robustness and generalization, cross-validation was employed during the training phase. This technique allowed for a more reliable assessment of model performance by splitting the dataset into multiple training and validation sets.

Conclusion

Summary

The purpose of this project was to develop a robust salary prediction model that accurately forecasts employee salaries based on various factors such as experience, position, department, and education level. The project utilized a dataset obtained from Kaggle, consisting of diverse employee attributes and salaries. The methodology involved several stages, including data preprocessing, feature selection, model training, and evaluation. After experimenting with multiple machine learning algorithms, the final model demonstrated promising results, achieving a satisfactory level of accuracy in predicting salaries.

Key Takeaways

The project provided valuable insights into the following aspects:

- **Data Quality**: The importance of data cleaning and preprocessing cannot be overstated. Addressing missing values and outliers significantly improved model performance.
- **Feature Selection**: Identifying and selecting relevant features played a crucial role in the accuracy of the salary predictions. Including factors like years of experience and position yielded the best results.
- **Model Selection**: The comparative analysis of different models highlighted the strengths and weaknesses of each. Linear Regression emerged as a simple yet effective approach for salary prediction, while more complex models may provide further improvements.
- **Evaluation Techniques**: Utilizing evaluation metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) helped assess model performance effectively and guide improvements.

Future Scope

Looking ahead, several potential improvements could enhance the salary prediction model:

- **Incorporating Additional Features**: Integrating more features, such as industry trends, geographical cost of living, and economic indicators, could provide a deeper understanding of salary determinants and improve prediction accuracy.
- **Exploring Advanced Models**: Testing other machine learning models, such as XGBoost or Neural Networks, may yield better performance by capturing complex relationships within the data.
- **Continuous Model Training**: Implementing a system for continuous model training with new data can help maintain accuracy over time as market conditions and salary trends evolve.
- **User-Friendly Interface**: Developing a user-friendly web interface for the model could make it accessible for HR professionals, allowing them to input employee data and receive salary predictions effortlessly.

In conclusion, this project has laid a strong foundation for further exploration in salary prediction, offering significant opportunities for improvement and real-world application in human resource management.