NIRF-2024
Engineering Rank Band (151-200)
Pharmacy Rank - 77
Innovation Rank Band (11-50)

www.kiet.edu
Delhi-NCR, Ghaziabad

Accredited by
NAAC with Grade "A+"

Employee Salary Analysis

by

TeamName- SalarySnyc

Rani Kumari(202410116100163)
Roopsi Srivastava(202410116100173)
Rabita Yadav (202410116100156)

Session:2024-2025 (II Semester)

Under the supervision of

Mrs. Komal Salgotra

(Assistant Professor)

KIET Group of Institutions, Delhi-NCR, Ghaziabad

DEPARTMENT OF COMPUTER APPLICATIONS

KIET GROUP OF INSTITUTIONS, DELHI-NCR,GHAZIABAD-201206

(2024-26)

Introduction:
In this project, the goal is to analyze employee salary data to uncover key patterns, trends, and relationships among various factors such as department, gender, overtime pay, longevity pay, and others. The ultimate aim is to create a predictive model for base salary using these features, applying techniques such as exploratory data analysis (EDA) and machine learning to extract actionable insights. By analyzing the factors influencing salary, organizations can make data-driven decisions that ensure fairness and competitiveness in compensation.

Project Implementation Step 1: Data Import and Preprocessing

1. **Data Loading**: We begin by importing the dataset into a pandas DataFrame.
2. **Handling Missing Data**: Missing values are handled by filling them with the median salary for numerical columns and using the mode for categorical columns.
3. **Data Transformation**: Categorical variables, such as the 'Department' column, are converted into numeric values using encoding techniques. Feature scaling is applied to the base salary column to normalize the data, making it suitable for machine learning.

Step 2: Exploratory Data Analysis (EDA)

1. **Basic Overview**: We start with an overview of the dataset by checking the data types and summary statistics.
2. **Missing Values Check**: We check for any missing data and handle it appropriately.
3. **Visualizing Salary Distribution**: The salary distribution is visualized using histograms and Kernel Density Estimation (KDE) plots, providing a clearer picture of how salaries are spread across the dataset.
4. **Correlation Analysis**: A heatmap is used to visualize correlations between numerical features, helping us understand how different features, such as base salary, overtime pay, and longevity pay, relate to each other.
5. **Salary by Department**: We explore how average salaries differ across various departments, which can highlight disparities in compensation between departments.

Step 3: Data Preprocessing for Machine Learning

1. **Handling Missing Values**: We continue to handle any missing values by filling in gaps where necessary.
2. **Encoding Categorical Variables**: For categorical columns like 'Department', one-hot encoding is applied to avoid assuming an inherent order of the categories.
3. **Feature Scaling**: We scale numerical features to ensure they are on a comparable scale, which helps improve the performance of machine learning algorithms.

Step 4: Deeper Salary Analysis

1. **Salary by Experience**: Although no explicit 'Experience' column was found in the dataset, a boxplot could be used to explore salary differences across experience levels (if data were available).
2. **Salary Trends Over Time**: A line plot is generated to analyze trends in salary over the years (assuming a time-related column exists). This can be adjusted if necessary.

Step 5: Machine Learning for Salary Prediction

1. **Model Preparation**: We set up the features (X) and target (y), where 'Base_Salary' is the target variable and other columns are the features.
2. **Train-Test Split**: The dataset is split into training and testing sets to evaluate the model's performance accurately.
3. **Model Training and Evaluation**: A linear regression model is trained, and performance is evaluated using Mean Squared Error (MSE) and R² score to assess how well the model predicts salaries.

```
# Importing necessary libraries import pandas as pd
import seaborn as sns import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler from
sklearn.model_selection import train_test_split from
sklearn.linear_model import LinearRegression from
sklearn.metrics import mean_squared_error, r2_score
# Load the dataset (change path as needed) import pandas as pd file_path =
'/content/Employee_Salaries.csv' # Update with your dataset's path data =
pd.read_csv(file_path)


# Preview the first few rows of the dataset data.head()
```

	Department	Department_Name	Division	Gender	Base_Salary	Overtime_Pay	Longevity_Pay	Grade
0	ABS	Alcohol Beverage Services	ABS 85 Administration	M	175873.000	0.00	0.0	M2
1	ABS	Alcohol Beverage Services	ABS 85 Administration	M	145613.360	0.00	0.0	M3
2	ABS	Alcohol Beverage Services	ABS 85 Administration	F	136970.000	0.00	0.0	M3

3	ABS Alcohol Beverage Services	ABS 85 Administrative Services	F	89432.694	0.00	2490.0	21
4	ABS Alcohol Beverage Services	ABS 85 Administrative Services	F	78947.000	456.68	6257.7	16

Step 2: Exploratory Data Analysis (EDA)

```
# Basic dataset information data.info()


# Descriptive statistics for numerical columns data.describe()


# Checking for missing values data.isnull().sum()
```

```
<class  'pandas.core.frame.DataFrame'>
RangeIndex: 10291 entries, 0 to 10290
Data columns (total 8 columns):
  #  Column            Non-Null Count Dtype



 -----  -----------            ------------------------  ---------
  0  Department   10291   non-null   object   1
  Department_Name  10291  non-null  object  2
  Division 10291 non-null object
  3   Gender  10291 non-null object
  4   Base_Salary      10291 non-null float64
  5   Overtime_Pay     10291 non-null float64
  6   Longevity_Pay    10291 non-null float64
  7   Grade   10258 non-null object
 dtypes: float64(3), object(5) memory
 usage: 643.3+ KB
```

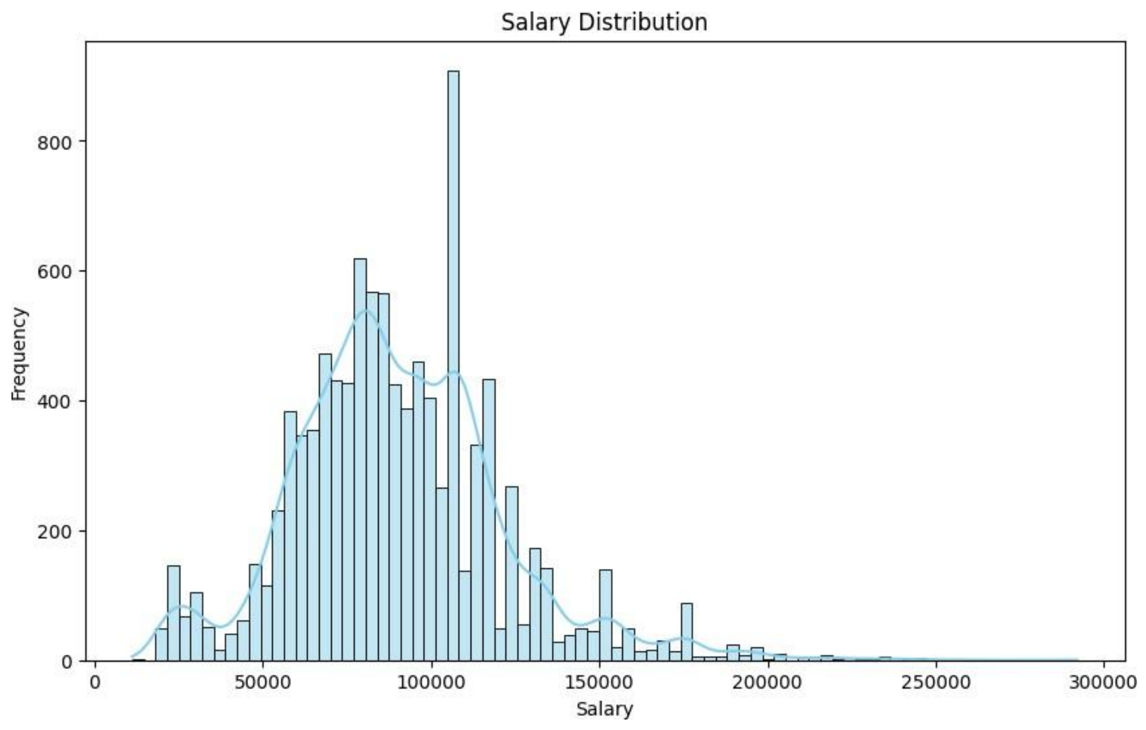
	0
Department	0
Department_Name	0
Division	0
Gender	0
Base_Salary	0
Overtime_Pay	0
Longevity_Pay	0
Grade	33

dtype: int64

2.2 : Visualizing Salary Distribut

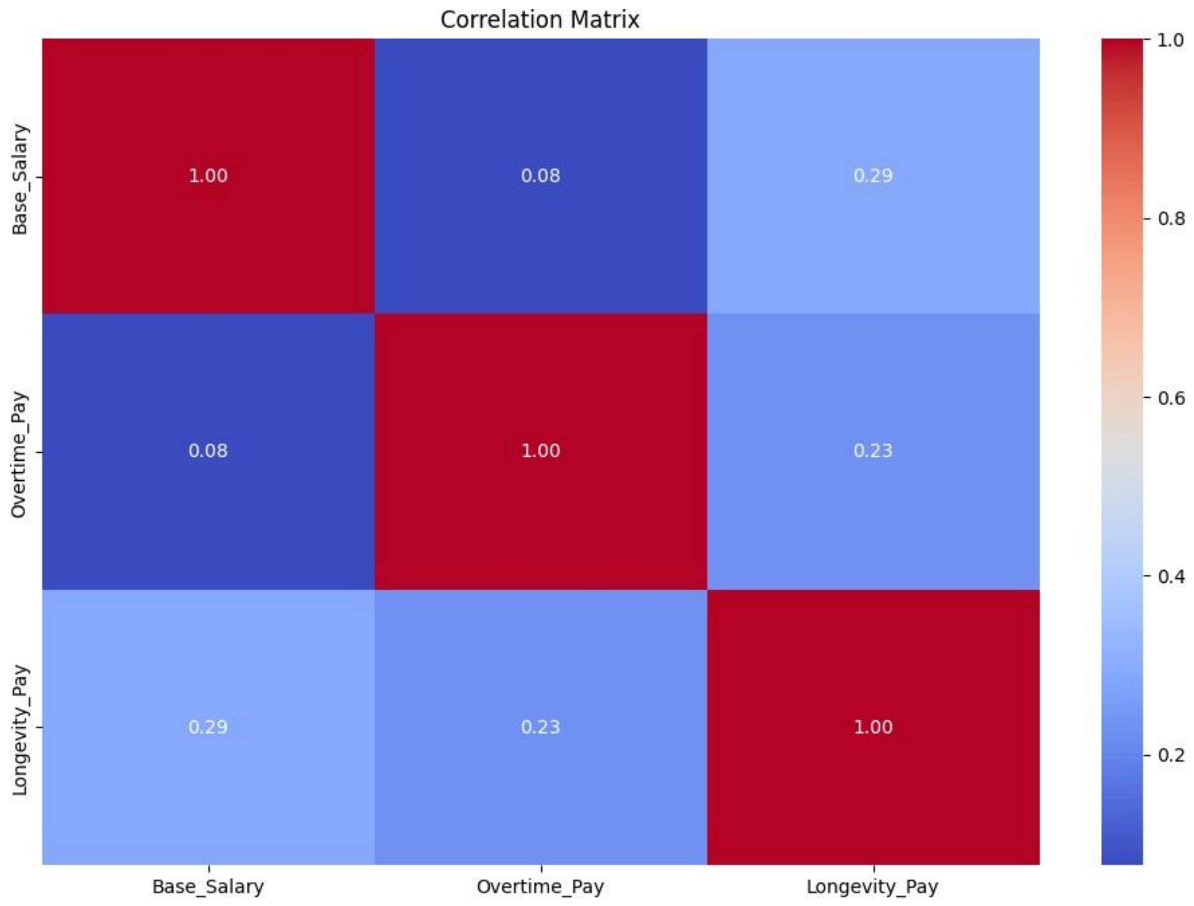
```
import  seaborn  as  sns  import
matplotlib.pyplot as plt


# Salary distribution (with a kernel density estimate)
```

```
plt.figure(figsize=(10,6))
sns.histplot(data['Base_Salary'], kde=True, color='skyblue')
plt.title('Salary    Distribution')    plt.xlabel('Salary')
plt.ylabel('Frequency') plt.show()
```



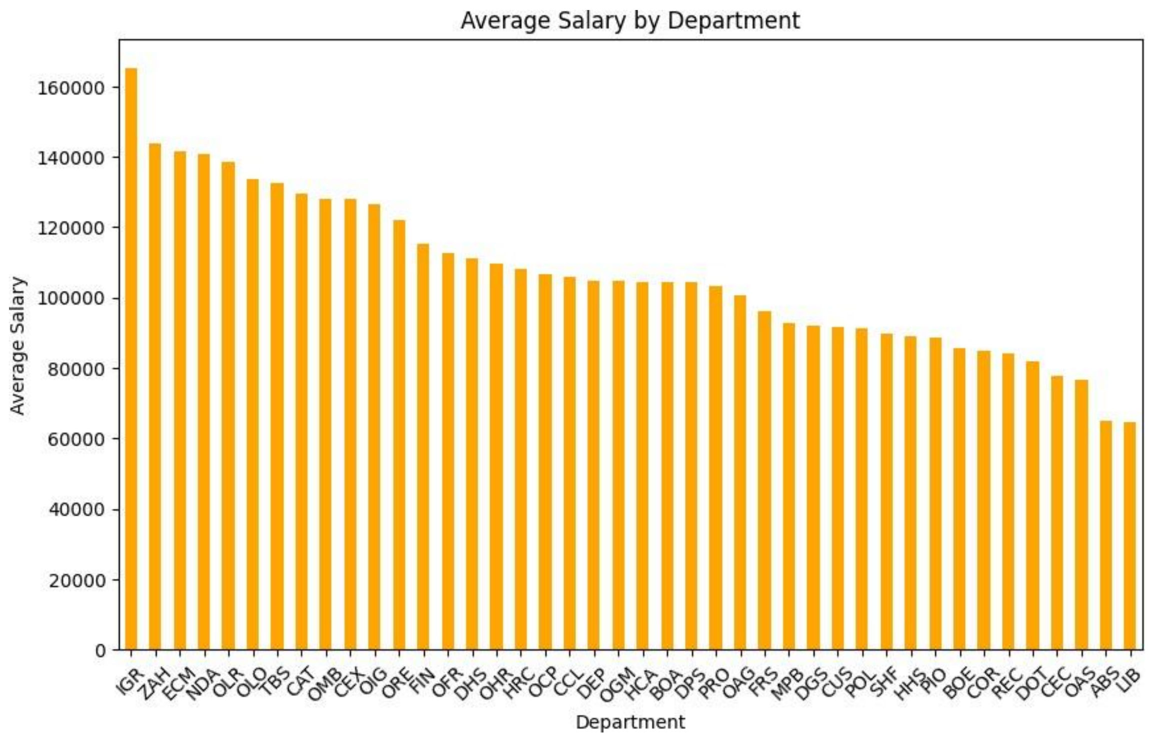
2.3 : Correlation Matrix

```
# Correlation matrix plt.figure(figsize=(12,8))
# Select only numerical features for correlation analysis numerical_data =
data.select_dtypes(include=['number'])  sns.heatmap(numerical_data.corr(),
annot=True,  cmap='coolwarm',  fmt='.2f')  plt.title('Correlation  Matrix')
plt.show()
```



2.4 : Salary by Department

```
#        Average        salary        by        department        salary_by_dept        =
data.groupby('Department')['Base_Salary'].mean().sort_values(ascending=False)
salary_by_dept.plot(kind='bar', figsize=(10,6), color='orange') plt.title('Average Salary by
Department') plt.xlabel('Department') plt.ylabel('Average Salary') plt.xticks(rotation=45)
plt.show()
```



Step 3: Data Preprocessing

3.1 : Handle Missing Values

```
# Fill missing values with the median salary (or drop if applicable)
data['Base_Salary'].fillna(data['Base_Salary'].median(), inplace=True)
data['Department'].fillna(data['Department'].mode()[0], inplace=True)


# Verify no missing values data.isnull().sum()
```

```
<ipython-input-8-354c1eca3ec1>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting


For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[co


data['Base_Salary'].fillna(data['Base_Salary'].median(), inplace=True)
<ipython-input-8-354c1eca3ec1>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained ass
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting


For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[co


data['Department'].fillna(data['Department'].mode()[0], inplace=True) 0
```

Department	0
Department_Name	0
Division	0
Gender	0
Base_Salary	0
Overtime_Pay	0
Longevity_Pay	0
Grade	33

dtype: int64

3.2 : Encode Categorical Variables

```
# Encoding 'Department' as numeric values (if it's a categorical column) data['Department']
= data['Department'].astype('category').cat.codes
```

3.3 : Feature Scaling (if required)

```
from sklearn.preprocessing import StandardScaler

# Scaling salary for normalization scaler
= StandardScaler()
data['Base_Salary'] = scaler.fit_transform(data[['Base_Salary']]) data.head()

# Check after scaling
```

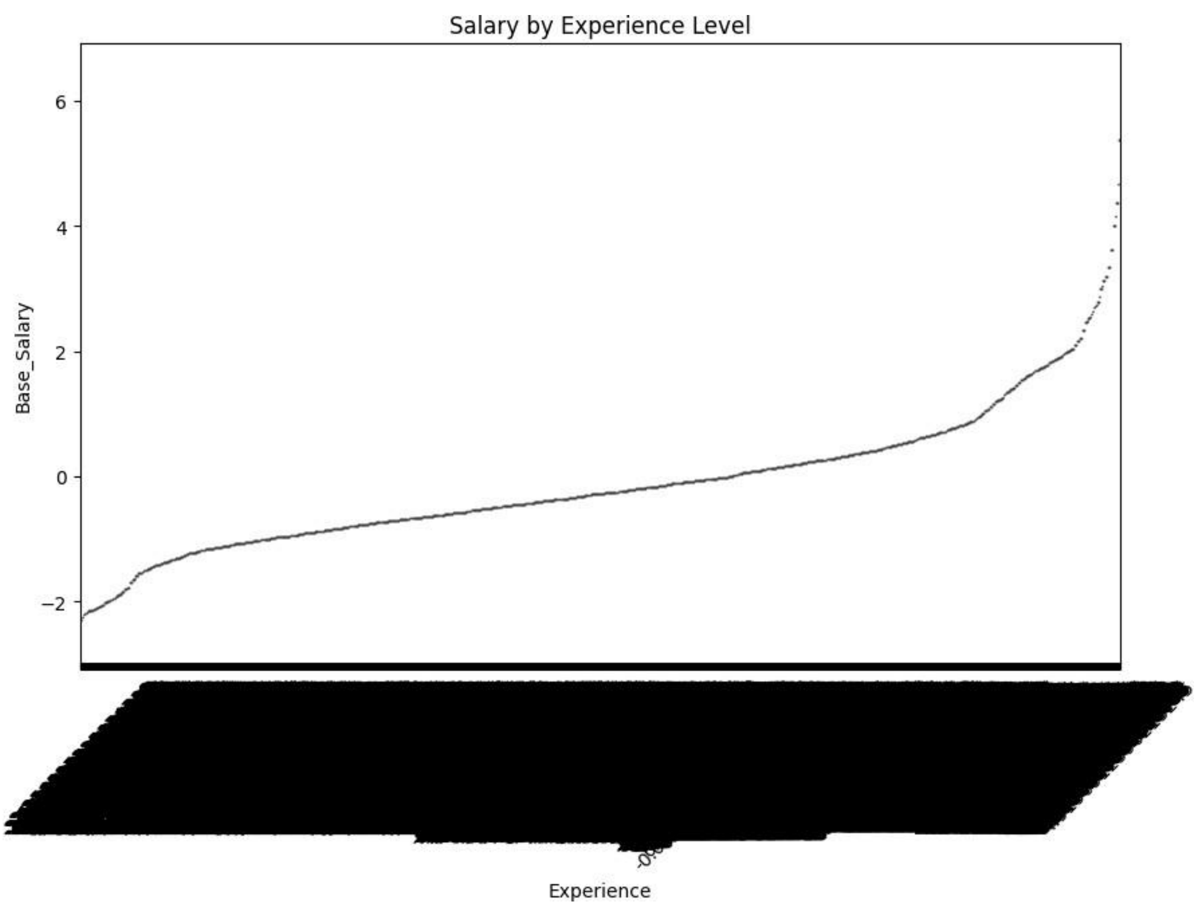
	Department	Department_Name	Division	Gender	Base_Salary	Overtime_Pay	Longevity_Pay	Grade
0	0	Alcohol Beverage Services	ABS 85 Administration	M	2.738882	0.00	0.0	M2
1	0	Alcohol Beverage Services	ABS 85 Administration	M	1.770243	0.00	0.0	M3

2	0 Alcohol Beverage Services	ABS 85 Administration F	1.493561	0.00	0.0	M3	
3	0 Alcohol Beverage Services	ABS 85 Administrative Services	F	-0.028153	0.00	2490.0	21
4	0 Alcohol Beverage Services	ABS 85 Administrative Services	F	-0.363810	456.68	6257.7	16

Step 4: Deeper Salary Analysis

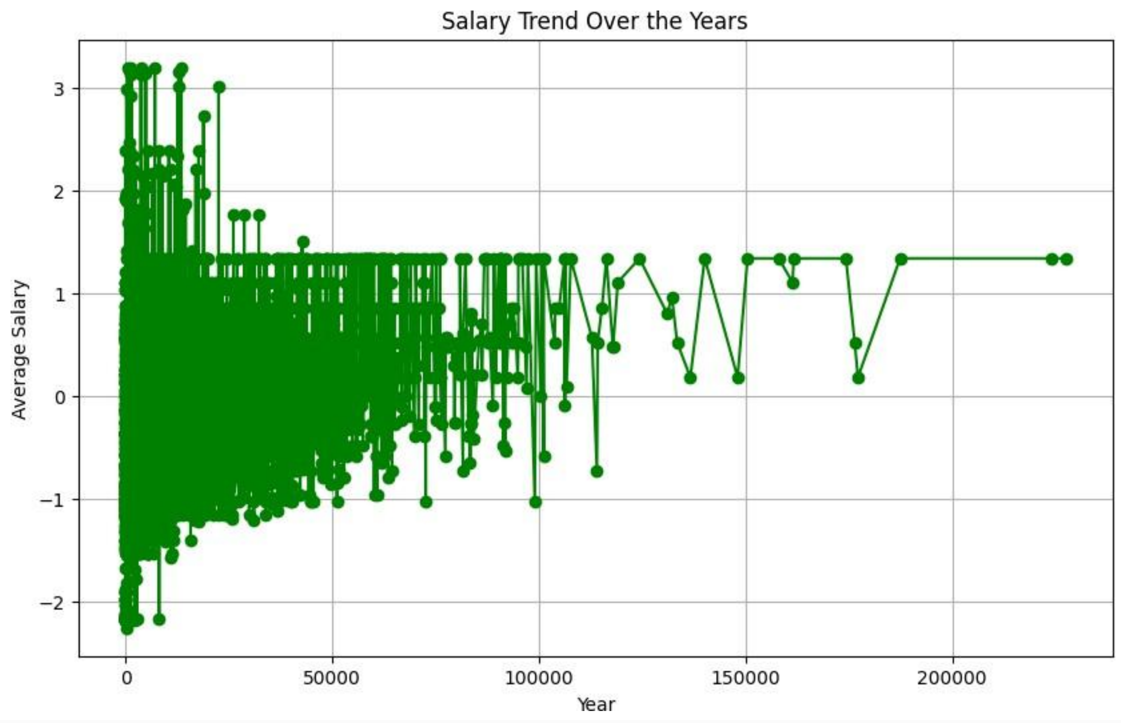
4.1 : Salary by Experience

```
# Salary by Experience Level (assuming the 'Experience' column exists)
plt.figure(figsize=(10,6))                sns.boxplot(x='Base_Salary',
y='Base_Salary', data=data) plt.title('Salary by Experience Level')
plt.xlabel('Experience')                     plt.ylabel('Base_Salary')
plt.xticks(rotation=45) plt.show()
```

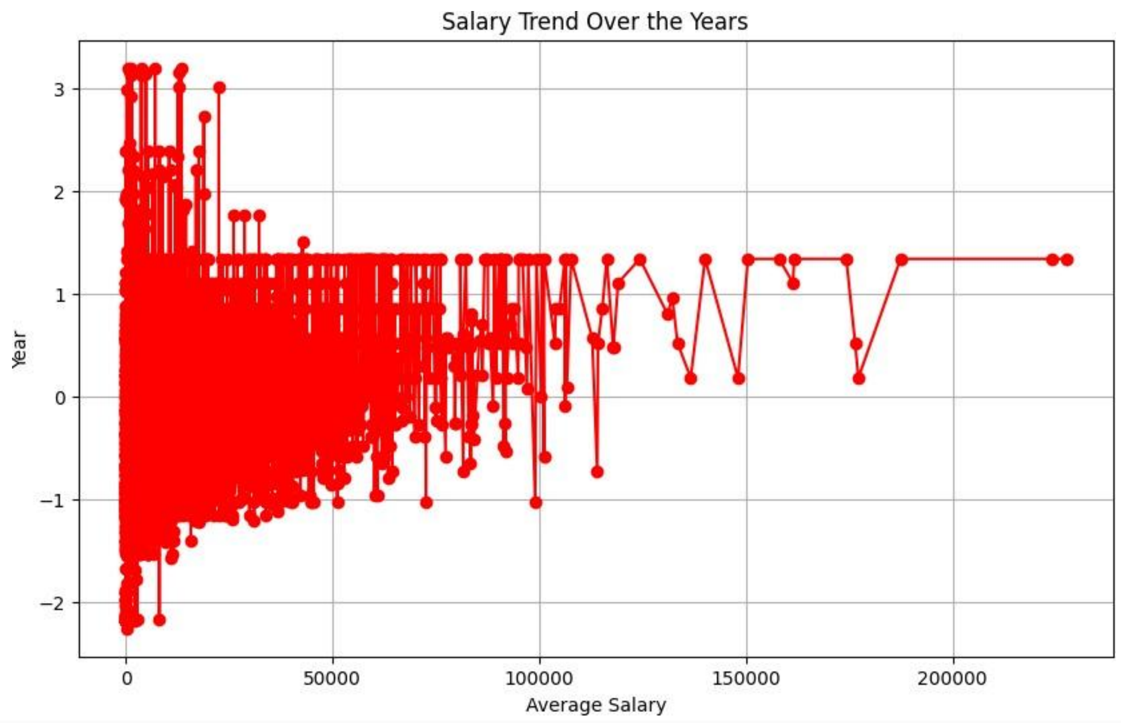


4.2 : Salary Trends Over Time

```
# Salary trend over years (assuming 'Year' column exists) salary_trends =
data.groupby('Overtime_Pay')['Base_Salary'].mean()
salary_trends.plot(kind='line', figsize=(10,6), color='green', marker='o')
plt.title('Salary    Trend    Over    the    Years')    plt.xlabel('Year')
plt.ylabel('Average Salary') plt.grid(True) plt.show()
```



```
# Salary trend over years (assuming 'Year' column exists)
# If 'Year' column doesn't exist, replace it with the actual column name #
containing year or date information. For example, if you have a column #
named 'Hire_Date', you can extract the year using:
# data['Year'] = pd.to_datetime(data['Hire_Date']).dt.year


# If you don't have any date or year related column, you might need to #
reconsider this part of the analysis.


# Assuming 'Year' column exists or has been created:
salary_trends    =    data.groupby('Overtime_Pay')['Base_Salary'].mean()
salary_trends.plot(kind='line', figsize=(10, 6), color='red', marker='o')
plt.title('Salary    Trend    Over    the    Years')    plt.ylabel('Year')
plt.xlabel('Average Salary') plt.grid(True)
plt.show()
```



Step 5: Machine Learning (Predicting Salary)

```
# 3.2: Encode Categorical Variables
# %%
# Encoding 'Department' as numeric values (if it's a categorical column)
# Use one-hot encoding to avoid implying an order in the departments data
= pd.get_dummies(data, columns=['Department'], drop_first=True)


# Import necessary libraries import pandas as pd from
sklearn.model_selection  import  train_test_split  from
sklearn.linear_model   import   LinearRegression   from
sklearn.metrics import mean_squared_error, r2_score


# Data Preprocessing data['Base_Salary'].fillna(data['Base_Salary'].median(), inplace=True) # Fill
missing salary values # Fill missing department


# Encoding categorical variables (like 'Department') data['Base_Salary']
= data['Base_Salary'].astype('category').cat.codes


#  Ensure  all  features  are  numeric  data  =
pd.get_dummies(data, drop_first=True)


# Prepare features (X) and target (y)
X = data.drop('Base_Salary', axis=1)
y = data['Base_Salary']


# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Train the linear regression model
model     =     LinearRegression()
model.fit(X_train, y_train)


# Predict on the test set y_pred
= model.predict(X_test)


#    Evaluate    the    model    mse    =
mean_squared_error(y_test, y_pred) r2 =
r2_score(y_test, y_pred)
```

Conclusions and Insights

- **Data Trends**: The analysis revealed several patterns, such as the distribution of salaries across departments and how factors like overtime and longevity pay influence the base salary. Some departments may have higher salaries due to the nature of the work, while others might offer more competitive benefits in the form of overtime or longevity pay.

- **Predictive Model Performance**: The linear regression model was trained to predict base salary based on features like department, overtime pay, and longevity pay. By evaluating the model with metrics like MSE and R², we assessed its predictive accuracy. The R² value reflects how well the features explain the variation in base salary, providing insights into the reliability of our model.

- **Salary Disparities**: A key takeaway from the analysis is understanding the potential salary disparities between departments, genders, and other factors. These insights can be used by HR departments to ensure fair compensation practices and make datadriven decisions to improve employee retention and satisfaction.

- **Further Improvements**: To enhance the model, additional features like experience, education level, or years of service could be incorporated. Additionally, trying different machine learning algorithms (such as Random Forest or Gradient Boosting) could potentially improve predictive performance.