ALLIANCE
UNIVERSITY
Private University established in Karnataka State by Act No.34 of year 2010
Recognized by the University Grants Commission (UGC), New Delhi

PROJECT REPORT

Bachelor of Computer Applications

SEMESTER – II
INTRODUCTION TO DATA SCIENCE

SALARY PREDICTION using DATA SCIENCE techniques

Submitted By :
Anuska Ghosh
BCA – A
Registration Number: 2411021240012

Department of Computer Applications
Alliance University
Chandapura – Anekal Main Road, Anekal
Bengaluru – 562 106

April 2025

NAME: ANUSKA GHOSH

REGISTRATION NUMBER: 2411021240012

SUBJECT: INTRODUCTION TO DATA SCIENCE

GITHUB REPO LINK: **https://github.com/anuskaghosh17/IDS_Project**

PROJECT TITLE: SALARY PREDICTION using DATA SCIENCE TECHNIQUES

Table of Contents

1. Introduction

In today's digital world, understanding the factors that influence salaries across industries is crucial. Data Science allows us to analyze job-related data to predict salaries based on experience, job title, employment type, and more. This project aims to explore such a dataset, apply data operations, and build a predictive model.

2. Project Goals

- Explore and clean the salary dataset
- Perform data preprocessing and statistical analysis
- Identify key features influencing salary
- Build a regression model to predict salary
- Evaluate the model's performance
- Derive actionable insights from the analysis

3. Dataset Overview

Column Name: Description

- **work_year:** Year of the record
- **experience_level:** Job experience level (e.g., Entry, Mid, Senior)
- **employment_type:** Type of employment (e.g., Full-time, Part-time)
- **job_title:** Job designation/title
- **salary:** Salary
- **salary_currency:** Currency type
- **salary_in_usd:** Salary in USD
- **employee_residence:** Country of residence
- **remote_ratio:** Percentage of remote work (0 to 100)
- **company_location:** Country of the company
- **company_size:** Size of the company (S, M, L)

4. CRISP-DM Model: Salary Prediction Project

1. BUSINESS UNDERSTANDING:

- **Objective:**

The primary goal of this project is to develop a predictive model that can estimate the salary of professionals based on various features such as job title, experience level, employment type, company size, remote work ratio, and geographic details. This model can help:

a> Employers offer competitive compensation.

b> Job seekers understand salary trends.

c> HR and recruitment teams identify fair pay practices.

- **Business Questions:**

a> Which job features most strongly influence salary?

b> Can we predict salary using regression models?

c> How do factors like remote work or company size impact salary?

2. DATA UNDERSTANDING:

- **Dataset Overview:**

We use a real-world dataset salaries.csv, which includes records of job positions across the tech industry (and possibly others) with salary details and job characteristics.

- **Key Variables:**

a> experience_level, employment_type, job_title: qualitative (categorical)

b> salary, remote_ratio: quantitative (numerical)

c> company_size, company_location, employee_residence: categorical (nominal/ordinal)

- **Initial Findings:**

a> The dataset contains both numerical and categorical data.

b> Some columns require encoding for modeling.

c> There may be outliers in salary values.

3. DATA PREPARATION:

- **Steps Taken:**

a> Data Cleaning: Checked for null values and imputed them using mean (for numeric), mode (for categorical), or median where necessary.

b> Outlier Detection: Identified using boxplots and handled based on context (e.g., salary extremes).

c> Encoding: Applied Label Encoding for ordinal features like experience_level, and One-Hot Encoding for nominal features like job_title.

d> Feature Scaling: Used Standardization (Z-score normalization) to bring numerical features to a common scale.

e> Correlation Analysis: Performed using a heatmap to detect multicollinearity and select influential features.

- **Outcome:**

Prepared a clean and consistent dataset ready for training regression models.

4. MODELING:

- **Chosen Model:**

Multiple Linear Regression: Appropriate because we're predicting a continuous target variable (salary) using multiple independent features.

- **Why Linear Regression?**

a> It's interpretable and efficient for continuous data.

b> Good baseline model for salary prediction tasks.

- **Process:**

a> Split data into training and testing subsets (80/20 split).

b> Trained the model on training data.

c> Used the model to predict salaries on the test data.

5. EVALUATION:

- **Metrics Used:**

a> Mean Squared Error (MSE) – Measures average squared prediction error.

b> Root Mean Squared Error (RMSE) – Provides error in actual units.

c> Mean Absolute Error (MAE) – Measures average magnitude of errors.

d> R^2 Score – Proportion of variance explained by the model.

- **Results Interpretation:**

a> A low RMSE and MAE indicate good predictive performance.

b> R^2 score closer to 1 means better fit of the model.

6. DEPLOYMENT:

This project is currently educational, but it has practical application in real-world scenarios:

a> Job Portals can use it to suggest salary expectations.

b> HR Teams can evaluate market alignment for salaries.

c> Startups can estimate competitive salary benchmarks.

5. Project Implementation in Real Life

This project can help:

- HR teams to benchmark salaries
- Job portals to show fair salary expectations
- Companies to evaluate market competitiveness
- Job seekers to understand trends in salaries based on job roles

6. Data Preprocessing

STEP1: IMPORTING THE REQUIRED LIBRARIES

Why?

These are standard Python libraries for:

a> Data manipulation (pandas, numpy)

b> Data visualization (matplotlib, seaborn)

c> Data preprocessing, modeling, and evaluation (sklearn)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
```

STEP2: LOAD THE DATASET

Why?

This loads the dataset into a pandas DataFrame

```
df = pd.read_csv(r"C:\Users\anusk\Downloads\salaries.csv")
df
```

```
       work_year experience_level employment_type
job_title  \
0           2025               MI              FT  Customer Success
Manager
1           2025               SE              FT
Engineer
2           2025               SE              FT
Engineer
3           2025               SE              FT           Applied
Scientist
4           2025               SE              FT           Applied
Scientist
...          ...              ...             ...
...
88579       2020               SE              FT              Data
Scientist
88580       2021               MI              FT    Principal Data
Scientist
88581       2020               EN              FT              Data
Scientist
88582       2020               EN              CT     Business Data
Analyst
88583       2021               SE              FT              Data
Scientist

        salary salary_currency  salary_in_usd employee_residence  \
0        57000             EUR          60000                 NL
1       165000             USD         165000                 US
2       109000             USD         109000                 US
3       294000             USD         294000                 US
4       137600             USD         137600                 US
...        ...             ...            ...                ...
88579   412000             USD         412000                 US
88580   151000             USD         151000                 US
88581   105000             USD         105000                 US
88582   100000             USD         100000                 US
88583  7000000             INR          94665                 IN

       remote_ratio company_location company_size
0                50               NL            L
1                 0               US            M
2                 0               US            M
```

```
3                 0               US            M
4                 0               US            M
...             ...              ...          ...
88579           100               US            L
88580           100               US            L
88581           100               US            S
88582           100               US            L
88583            50               IN            L

[88584 rows x 11 columns]
```

Displaying the first 5 rows

```
df.head()

   work_year experience_level employment_type
job_title  \
0       2025               MI              FT  Customer Success
Manager
1       2025               SE              FT
Engineer
2       2025               SE              FT
Engineer
3       2025               SE              FT         Applied
Scientist
4       2025               SE              FT         Applied
Scientist

   salary salary_currency  salary_in_usd employee_residence
remote_ratio  \
0   57000             EUR          60000                 NL
50
1  165000             USD         165000                 US
0
2  109000             USD         109000                 US
0
3  294000             USD         294000                 US
0
4  137600             USD         137600                 US
0

  company_location company_size
0               NL            L
1               US            M
2               US            M
3               US            M
4               US            M
```

Displaying the last 5 rows

```
df.tail()
```

```
       work_year experience_level employment_type
job_title  \
88579       2020               SE              FT             Data
Scientist
88580       2021               MI              FT  Principal Data
Scientist
88581       2020               EN              FT             Data
Scientist
88582       2020               EN              CT     Business Data
Analyst
88583       2021               SE              FT             Data
Scientist

        salary salary_currency  salary_in_usd employee_residence  \
88579   412000             USD         412000                 US
88580   151000             USD         151000                 US
88581   105000             USD         105000                 US
88582   100000             USD         100000                 US
88583  7000000             INR          94665                 IN

       remote_ratio company_location company_size
88579           100               US            L
88580           100               US            L
88581           100               US            S
88582           100               US            L
88583            50               IN            L
```

STEP 3: SUMMARY AND STATISTICAL SUMMARY OF THE DATASET

df.info(): Gives info about the dataset like data types, non-null counts, and memory usage. Helps identify missing values and data types.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 88584 entries, 0 to 88583
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   work_year           88584 non-null  int64
 1   experience_level    88584 non-null  object
 2   employment_type     88584 non-null  object
 3   job_title           88584 non-null  object
 4   salary              88584 non-null  int64
 5   salary_currency     88584 non-null  object
 6   salary_in_usd       88584 non-null  int64
 7   employee_residence  88584 non-null  object
 8   remote_ratio        88584 non-null  int64
```

```
 9   company_location    88584 non-null  object
 10  company_size        88584 non-null  object
dtypes: int64(4), object(7)
memory usage: 7.4+ MB
```

df.describe(): Provides summary statistics for both numerical and categorical columns (mean, std, count, top, freq, etc.)

```
# For Numerical Columns
df.describe()

          work_year        salary  salary_in_usd  remote_ratio
count  88584.000000  8.858400e+04   88584.000000  88584.000000
mean    2024.034758  1.619323e+05  157567.798417     21.286011
std        0.620370  1.965317e+05   73531.373158     40.831018
min     2020.000000  1.400000e+04   15000.000000      0.000000
25%     2024.000000  1.060000e+05  106097.250000      0.000000
50%     2024.000000  1.470000e+05  146307.000000      0.000000
75%     2024.000000  1.995000e+05  198600.000000      0.000000
max     2025.000000  3.040000e+07  800000.000000    100.000000

# For both Numerical and Categorical Columns
df.describe(include='all')

           work_year experience_level employment_type       job_title  
\
count   88584.000000            88584           88584           88584  

unique           NaN                4               4             312  

top              NaN               SE              FT  Data Scientist  

freq             NaN            51596           88111           13156  

mean     2024.034758              NaN             NaN             NaN  

std         0.620370              NaN             NaN             NaN  

min      2020.000000              NaN             NaN             NaN  

25%      2024.000000              NaN             NaN             NaN  

50%      2024.000000              NaN             NaN             NaN  

75%      2024.000000              NaN             NaN             NaN  

max      2025.000000              NaN             NaN             NaN  


              salary salary_currency  salary_in_usd employee_residence  
\
```

count	8.858400e+04	88584	88584.000000	88584
unique	NaN	26	NaN	96
top	NaN	USD	NaN	US
freq	NaN	83994	NaN	79705
mean	1.619323e+05	NaN	157567.798417	NaN
std	1.965317e+05	NaN	73531.373158	NaN
min	1.400000e+04	NaN	15000.000000	NaN
25%	1.060000e+05	NaN	106097.250000	NaN
50%	1.470000e+05	NaN	146307.000000	NaN
75%	1.995000e+05	NaN	198600.000000	NaN
max	3.040000e+07	NaN	800000.000000	NaN

	remote_ratio	company_location	company_size
count	88584.000000	88584	88584
unique	NaN	90	3
top	NaN	US	M
freq	NaN	79762	85667
mean	21.286011	NaN	NaN
std	40.831018	NaN	NaN
min	0.000000	NaN	NaN
25%	0.000000	NaN	NaN
50%	0.000000	NaN	NaN
75%	0.000000	NaN	NaN
max	100.000000	NaN	NaN

STEP 4: CHECK FOR MISSING VALUES

df.isnull().sum(): This checks how many null values exist in each column. We'll handle them if found.

```
df.isnull().sum()
```

work_year	0
experience_level	0
employment_type	0
job_title	0
salary	0
salary_currency	0
salary_in_usd	0
employee_residence	0

```
remote_ratio          0
company_location      0
company_size          0
dtype: int64
```

Output Interpretation:

After checking for missing values using df.isnull().sum(), we found that there are no missing values in the dataset. This indicates that the dataset is clean in this aspect, and no imputation techniques are required for further processing.

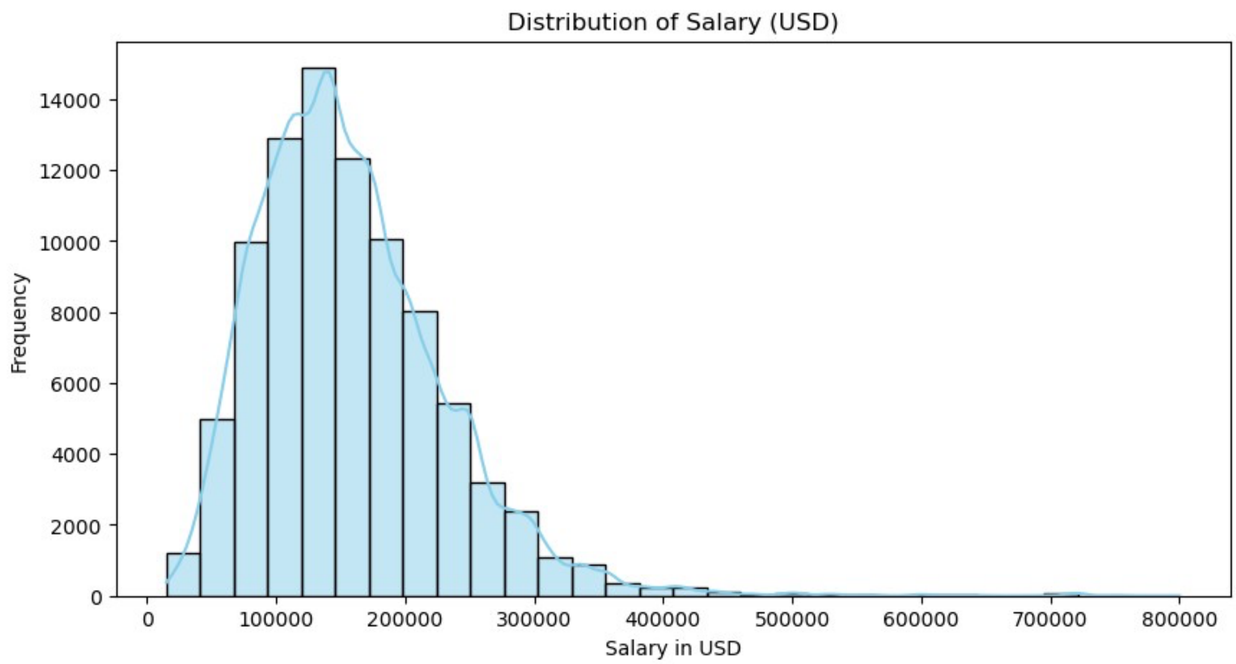
7. Exploratory Data Analysis (EDA)

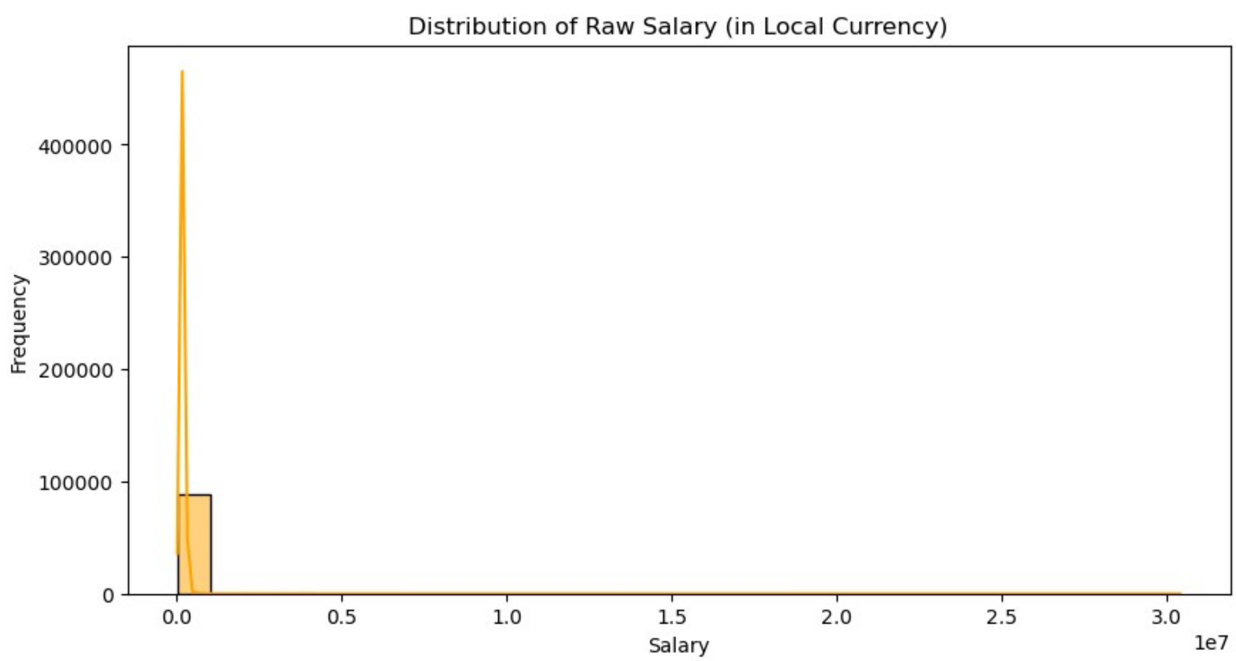
We explore the data through visualizations and analysis to uncover patterns.

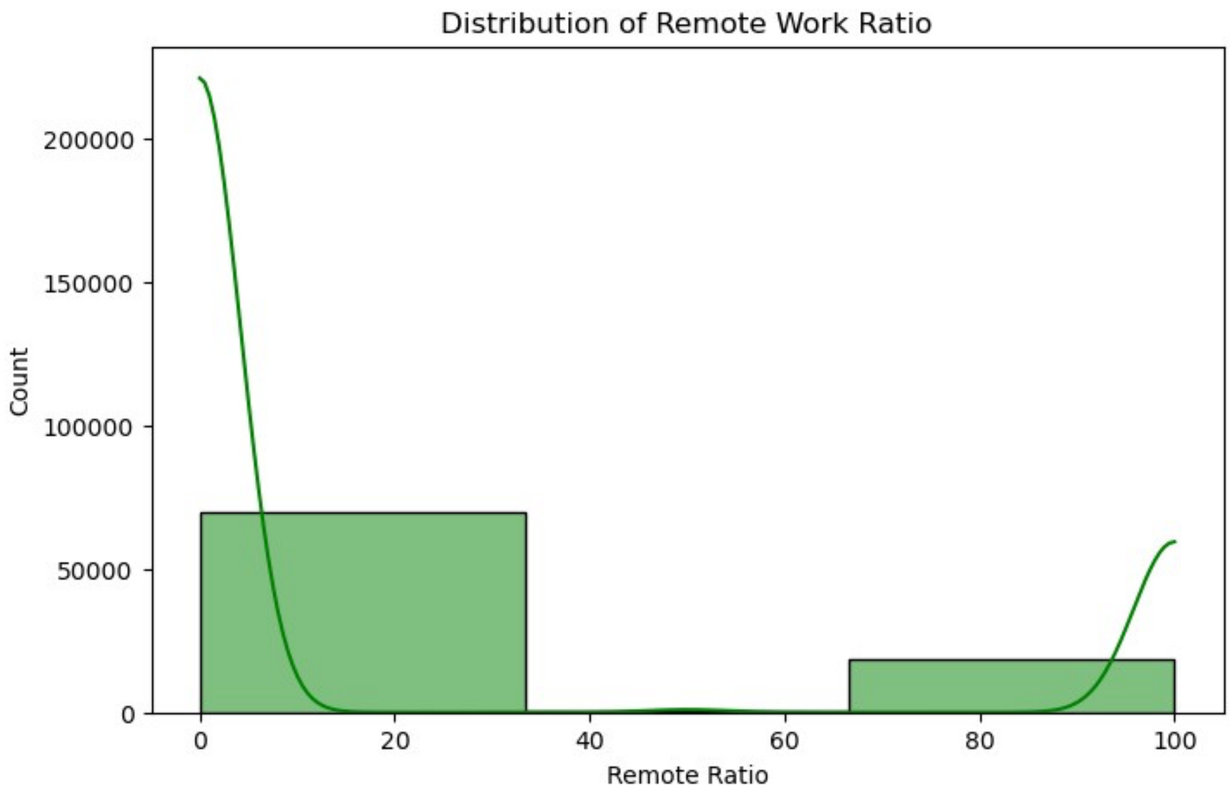
HISTOGRAM

```
# Distribution of salary in USD
# □ Purpose: Understand the spread and skewness of salary distribution across all data.
# □ Insight: We may observe right-skewness due to a few high-paying roles.
plt.figure(figsize=(10, 5))
sns.histplot(df['salary_in_usd'], bins=30, kde=True, color='skyblue')
plt.title('Distribution of Salary (USD)')
plt.xlabel('Salary in USD')
plt.ylabel('Frequency')
plt.show()

# Distribution of raw salary (local currency)
# □ Purpose: See how salary looks in native currencies. Could be helpful for understanding raw input.
# □ Insight: Skew may vary due to currency and local salary structures.
plt.figure(figsize=(10, 5))
sns.histplot(df['salary'], bins=30, kde=True, color='orange')
plt.title('Distribution of Raw Salary (in Local Currency)')
plt.xlabel('Salary')
plt.ylabel('Frequency')
plt.show()

# Distribution of remote ratio
# □ Purpose: Understand how many jobs are fully remote, partially remote, or on-site.
# □ Insight: 0 (fully on-site), 50 (hybrid), and 100 (fully remote) are the common values.
plt.figure(figsize=(8, 5))
sns.histplot(df['remote_ratio'], bins=3, kde=True, color='green')
plt.title('Distribution of Remote Work Ratio')
plt.xlabel('Remote Ratio')
```

```
plt.ylabel('Count')
plt.show()
```



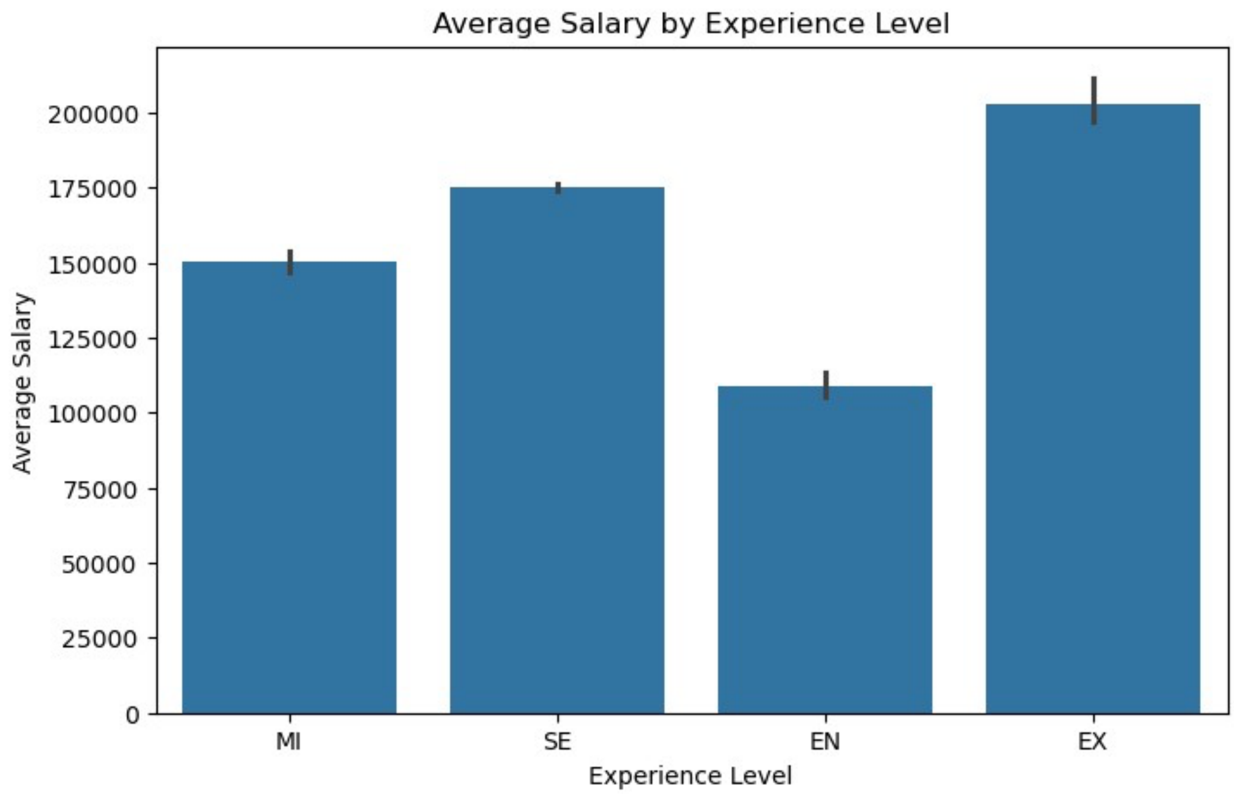


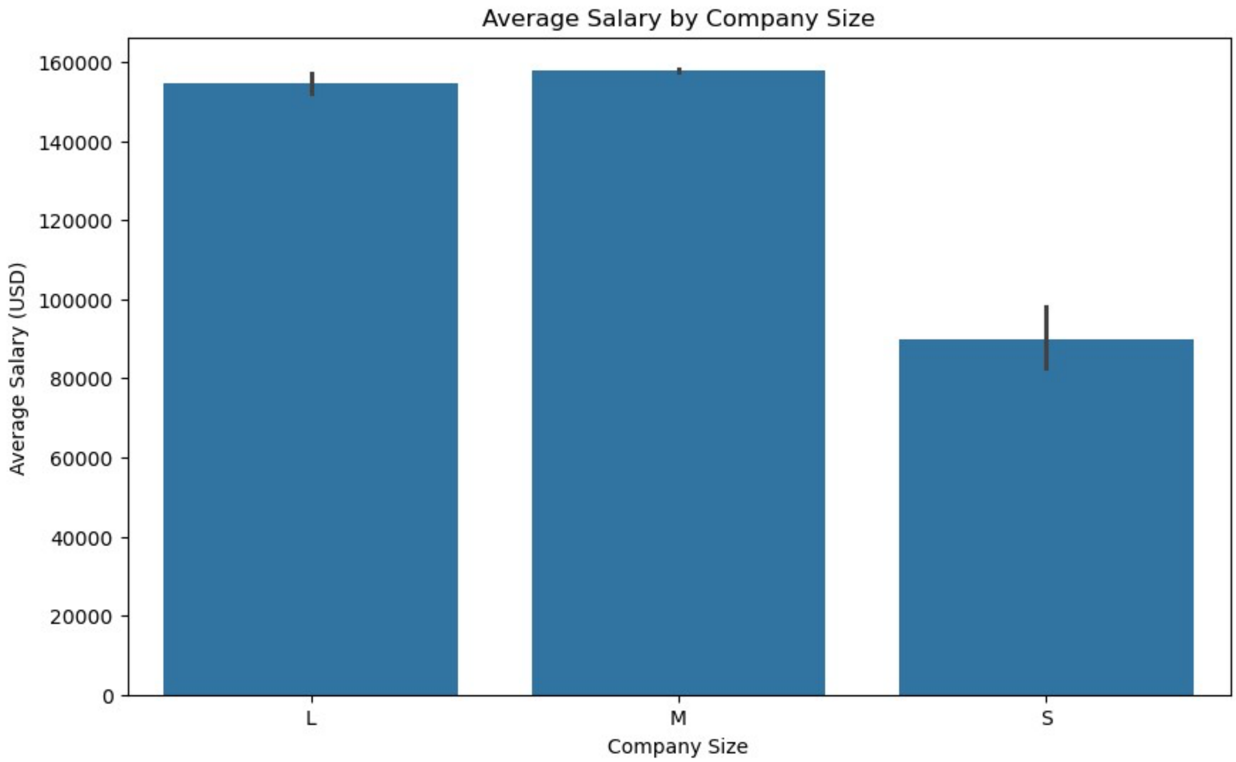


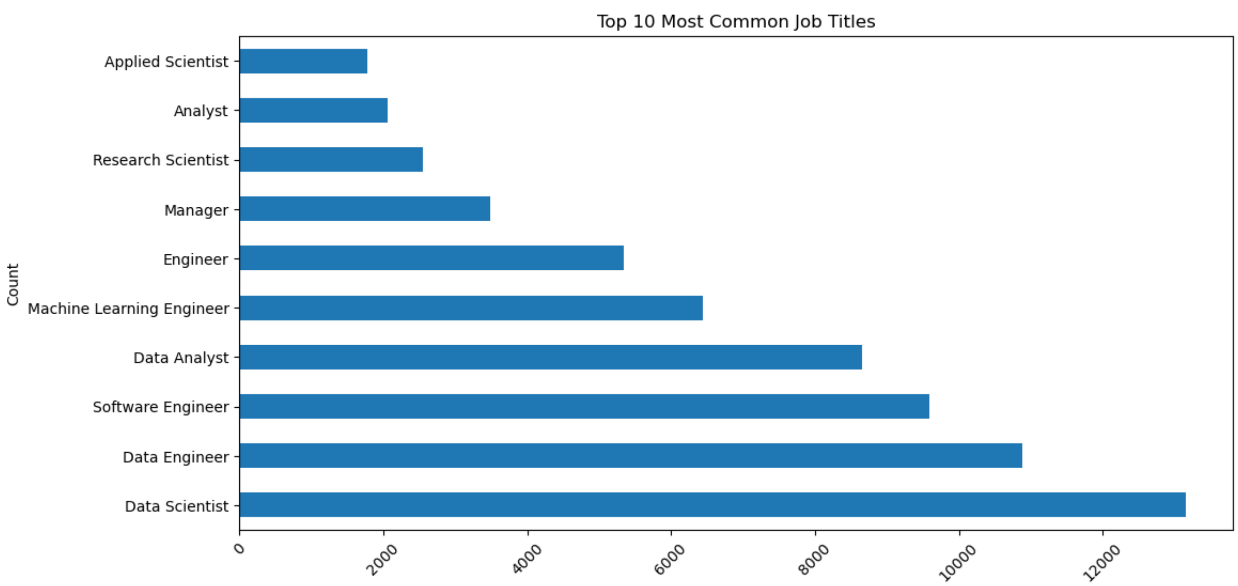
BARPLOT

```
# Barplot: Average salary by experience level
# Purpose: Visualize how average salaries increase with experience,
confirming higher pay for senior/executive roles.
# Insights: Barplots show trends—e.g., higher experience often means
higher salary.
plt.figure(figsize=(8, 5))
sns.barplot(x='experience_level', y='salary', data=df,
estimator=np.mean)
plt.title("Average Salary by Experience Level")
plt.xlabel("Experience Level")
plt.ylabel("Average Salary")
plt.show()

# Bar Plot: Average Salary by Company Size with Labels
# Purpose: Display the average salary offered by companies of
different sizes (S = Small, M = Medium, L = Large).
# Insight: Larger companies tend to offer higher salaries on average
compared to small or medium-sized companies.
plt.figure(figsize=(10, 6))
sns.barplot(x='company_size', y='salary_in_usd', data=df,
estimator=np.mean)
plt.title('Average Salary by Company Size')
plt.xlabel('Company Size')
plt.ylabel('Average Salary (USD)')
```

```
plt.show()

# Horizontal Barplot
# Count of job titles
plt.figure(figsize=(12, 6))
df['job_title'].value_counts().nlargest(10).plot(kind='barh')
plt.title('Top 10 Most Common Job Titles')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



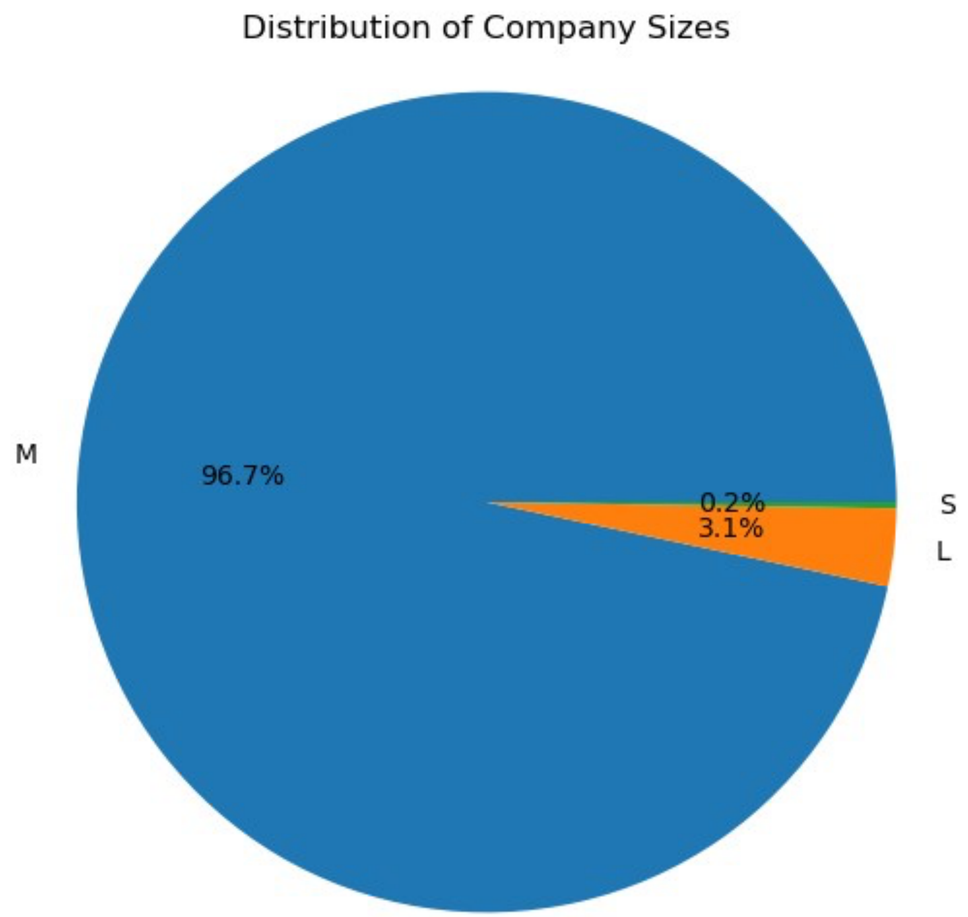




PIE CHART

```
# Pie chart for company sizes
# Purpose: To show the proportion of employees working in small,
medium, and large companies.
# Insights: The pie chart shows that most employees work in large
companies, followed by medium and small companies.
company_counts = df['company_size'].value_counts()
```

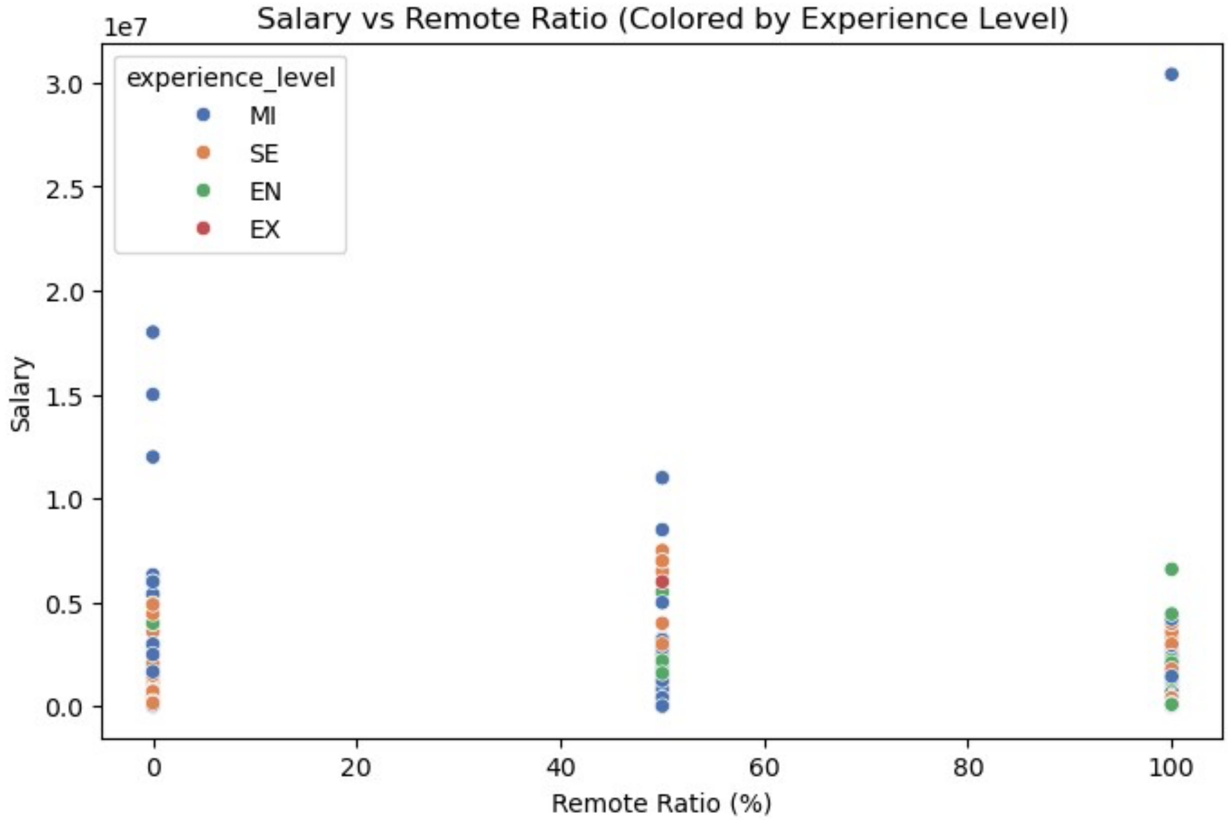
```
plt.figure(figsize=(6, 6))
plt.pie(company_counts, labels=company_counts.index, autopct='%1.1f%
%')

plt.title("Distribution of Company Sizes")
plt.axis('equal')  # Equal aspect ratio ensures the pie is a circle.
plt.show()
```



SCATTERPLOT

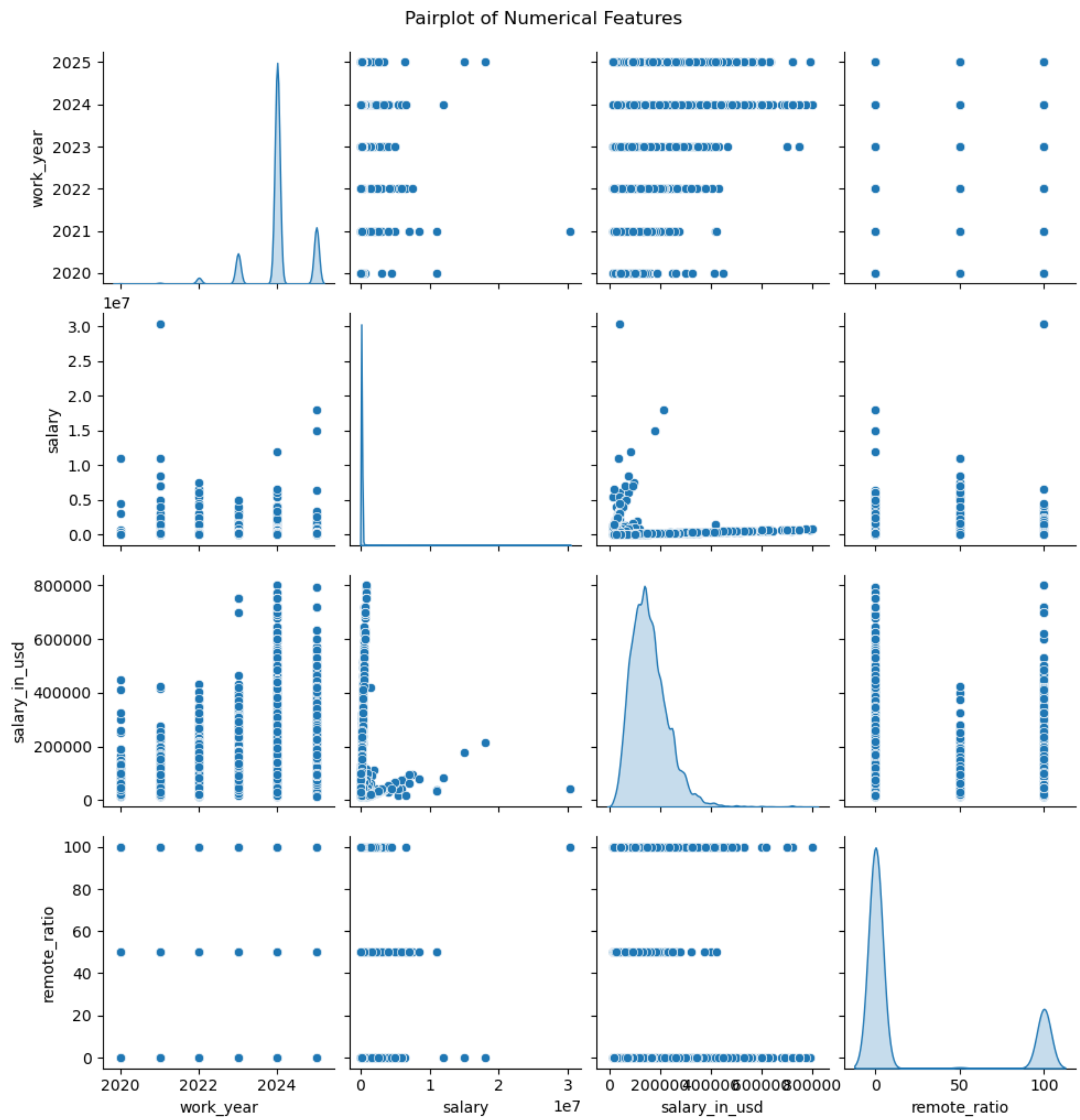
```
# Purpose: Understand relationship between two numerical variables,
e.g., salary and remote ratio.
# Insights: Helps us identify trends, such as whether higher remote
ratio affects salary, and how it varies with experience.
plt.figure(figsize=(8, 5))
sns.scatterplot(x='remote_ratio', y='salary', hue='experience_level',
data=df, palette='deep')
plt.title("Salary vs Remote Ratio (Colored by Experience Level)")
plt.xlabel("Remote Ratio (%)")
```

```
plt.ylabel("Salary")
plt.show()
```



PAIRPLOT

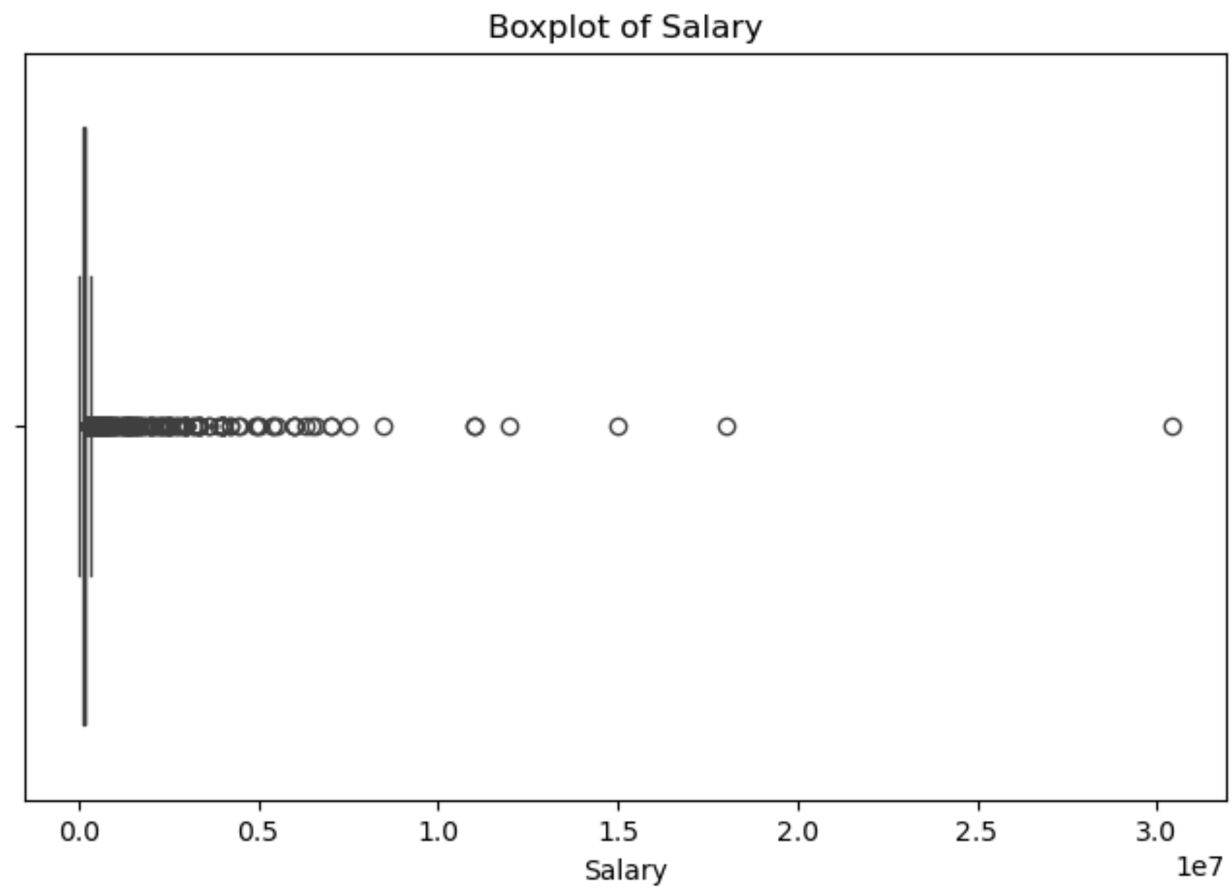
```
# Purpose: Multi-dimensional comparison of all numerical features and
their relationships.
# Insights: Helps visualize correlations, clustering, and
distributions in one grid.
# work_year: To observe salary and remote work trends over different
years.
# salary: To analyze pay variations in original currencies.
# salary_in_usd: To compare salaries globally on a consistent scale.
# remote_ratio: To explore the impact of remote work levels on
salaries and time trends.
sns.pairplot(df[['work_year', 'salary', 'salary_in_usd',
'remote_ratio']], diag_kind='kde')
plt.suptitle("Pairplot of Numerical Features", y=1.02)
plt.show()
```

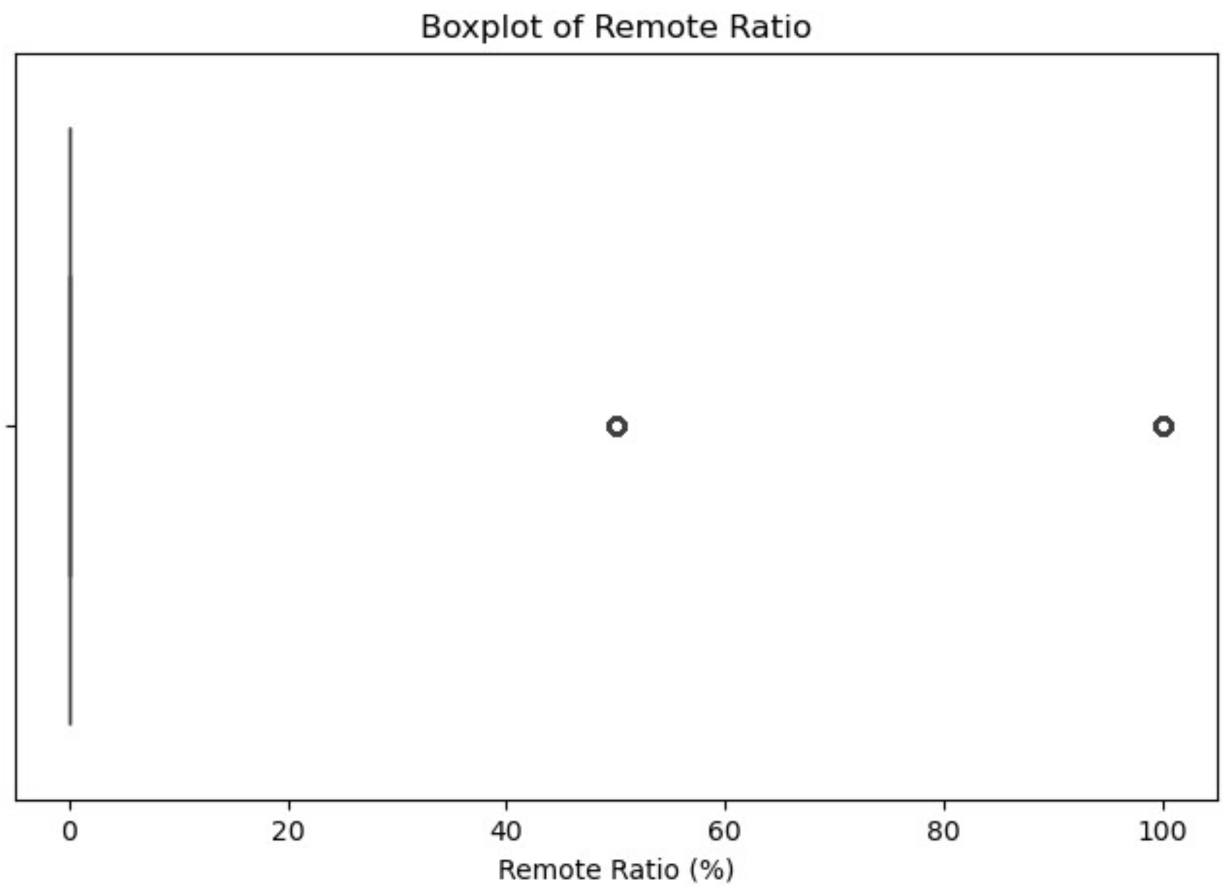


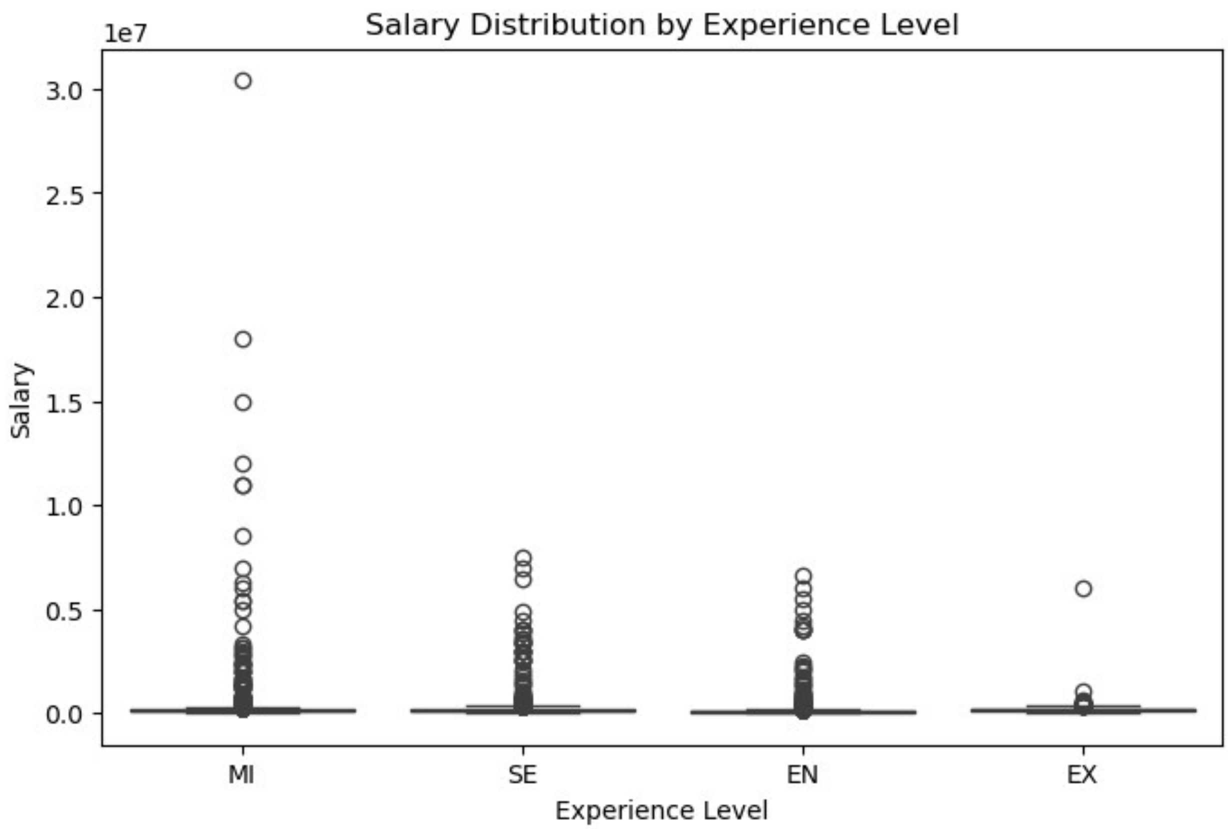
BOXPLOT

```
# 1. Boxplot of salary
# Interpretation: This plot shows how salary values are distributed
and highlights any outliers (points outside the whiskers).
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['salary'])
plt.title("Boxplot of Salary")
plt.xlabel("Salary")
plt.show()
```

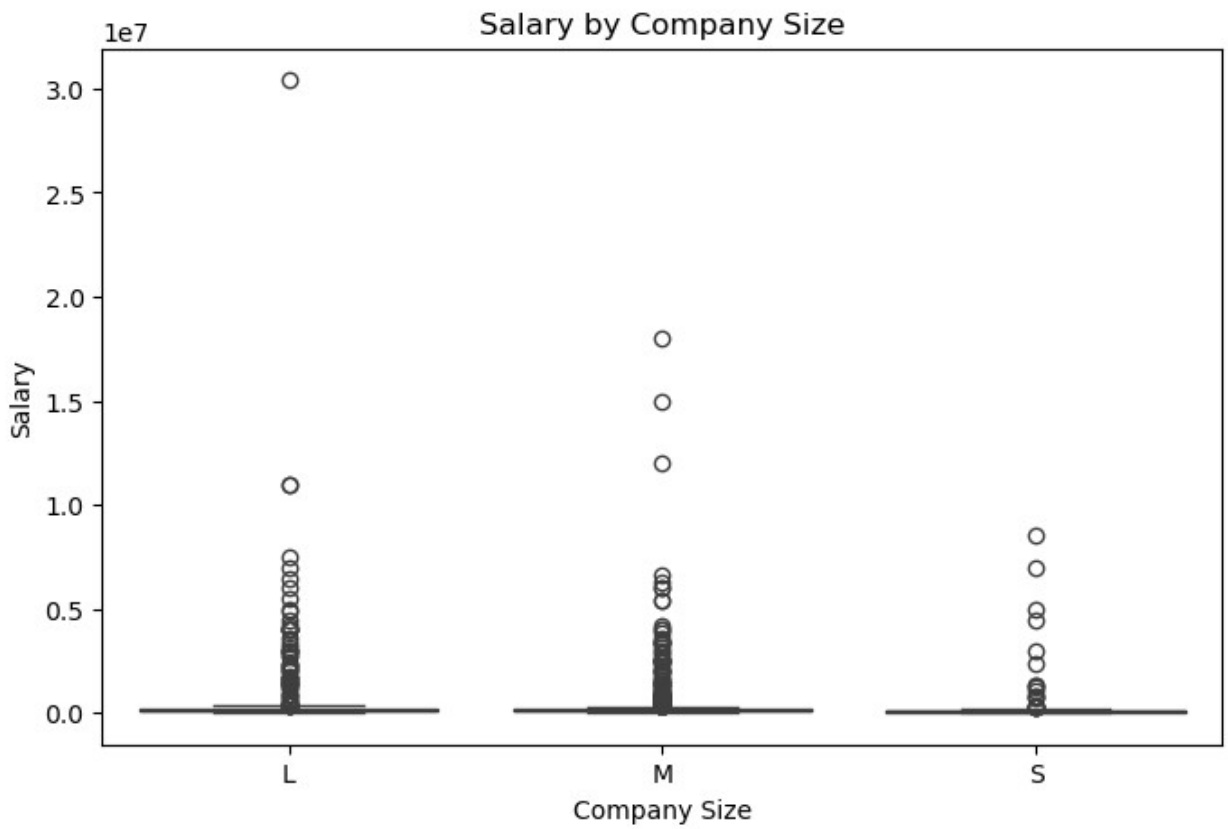
```python
# 2. Boxplot of remote_ratio
# Interpretation: Helps identify outliers in how much work is done remotely. If data is heavily remote or non-remote, it will show up here.
plt.figure(figsize=(8, 5))
sns.boxplot(x=df['remote_ratio'])
plt.title("Boxplot of Remote Ratio")
plt.xlabel("Remote Ratio (%)")
plt.show()

# 3. Boxplot of salary by experience_level
# Interpretation: Visualizes how salary changes across different experience levels like Entry, Mid, Senior, etc.
plt.figure(figsize=(8, 5))
sns.boxplot(x='experience_level', y='salary', data=df)
plt.title("Salary Distribution by Experience Level")
plt.xlabel("Experience Level")
plt.ylabel("Salary")
plt.show()

# 4. Boxplot of salary by employment_type
# Interpretation: See how salaries differ between full-time, part-time, freelance, or contract positions.
plt.figure(figsize=(8, 5))
sns.boxplot(x='employment_type', y='salary', data=df)
plt.title("Salary by Employment Type")
plt.xlabel("Employment Type")
plt.ylabel("Salary")
plt.show()

# 5. Boxplot of salary by company_size
# Interpretation: Large companies may offer higher salaries—this chart helps compare.
plt.figure(figsize=(8, 5))
sns.boxplot(x='company_size', y='salary', data=df)
plt.title("Salary by Company Size")
plt.xlabel("Company Size")
plt.ylabel("Salary")
plt.show()

# 6. Boxplot of salary by work_year
# Interpretation: Check how salary trends have changed over the years.
plt.figure(figsize=(8, 5))
sns.boxplot(x='work_year', y='salary', data=df)
plt.title("Salary by Work Year")
plt.xlabel("Work Year")
plt.ylabel("Salary")
plt.show()
```

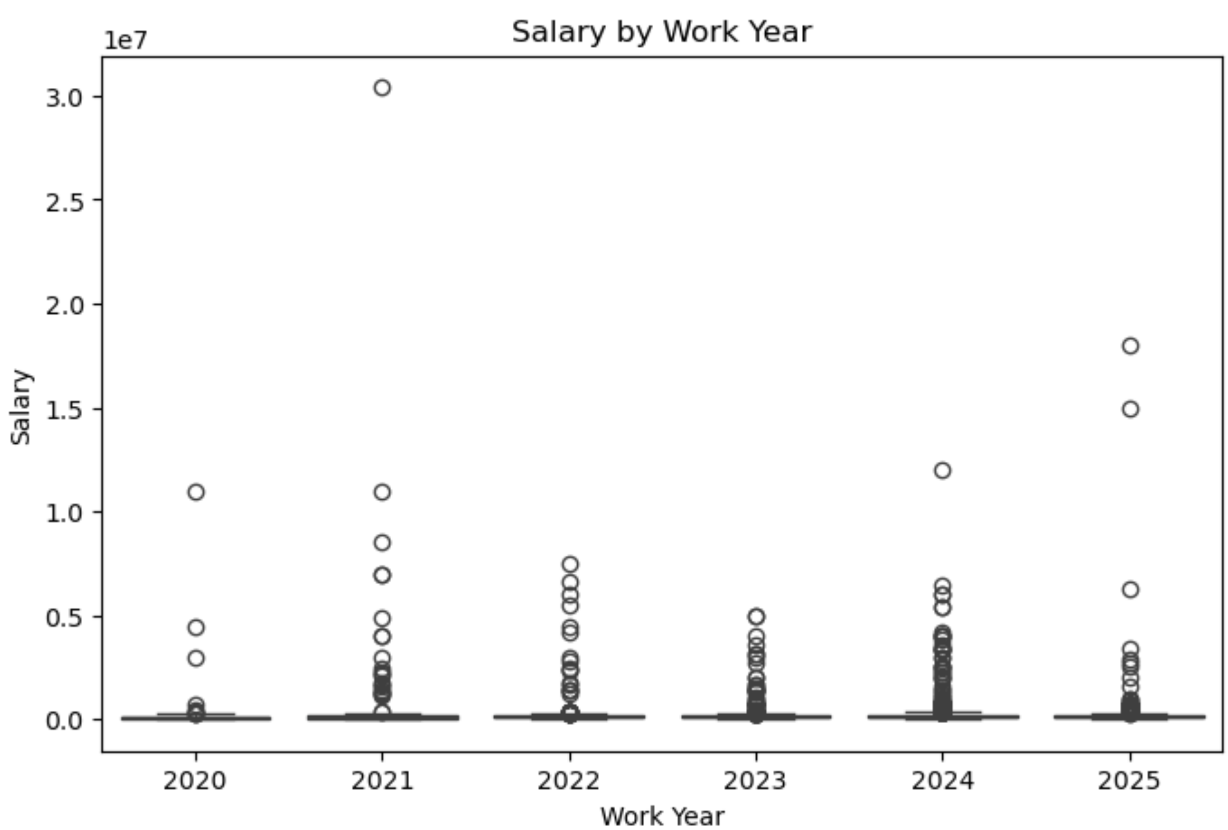

Boxplot of Salary
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1e7
Salary


Boxplot of Remote Ratio
0
20
40
60
80
100
Remote Ratio (%)


Salary Distribution by Experience Level
1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Salary
MI
SE
EN
EX
Experience Level

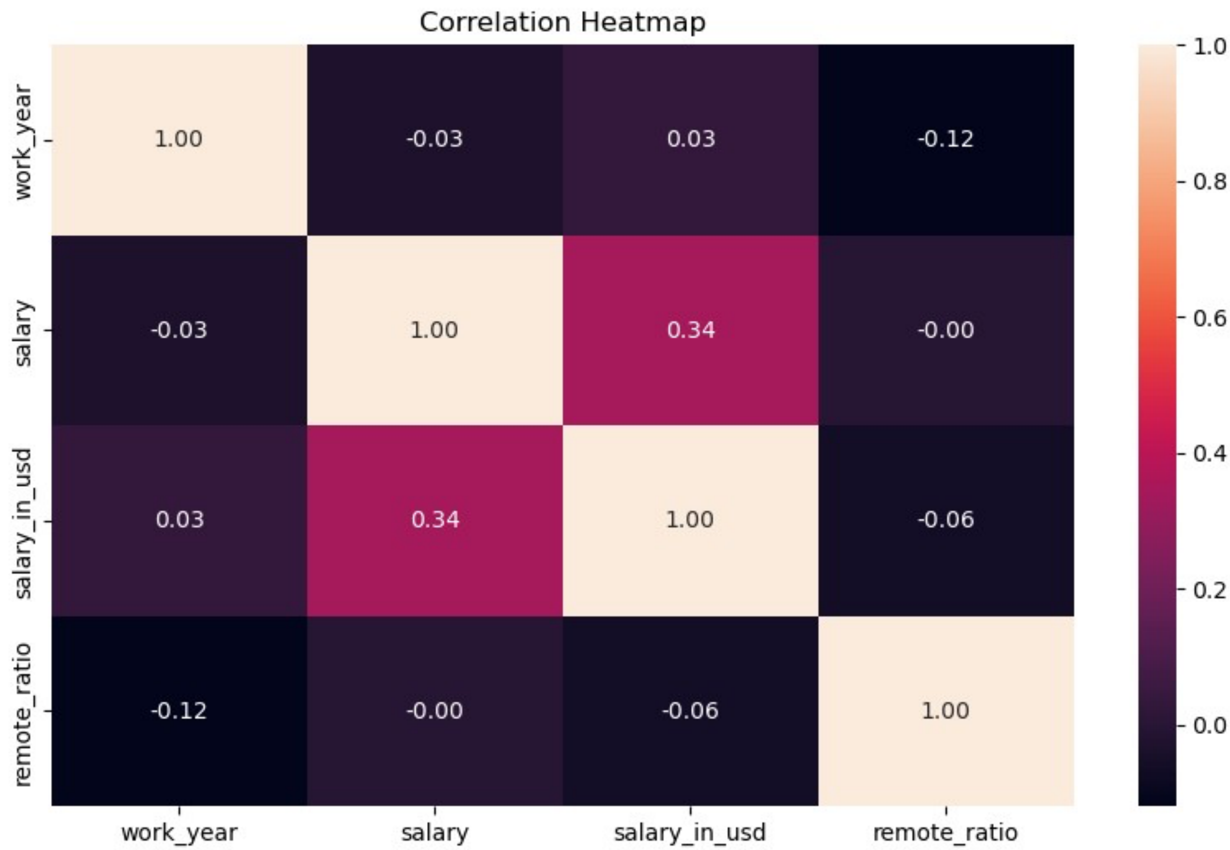

Salary by Employment Type
1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Salary
FT
CT
PT
FL
Employment Type


Salary by Company Size
1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Salary
L
M
S
Company Size


Salary by Work Year
1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Salary
2020
2021
2022
2023
2024
2025
Work Year

CORRELATION AND HEATMAP

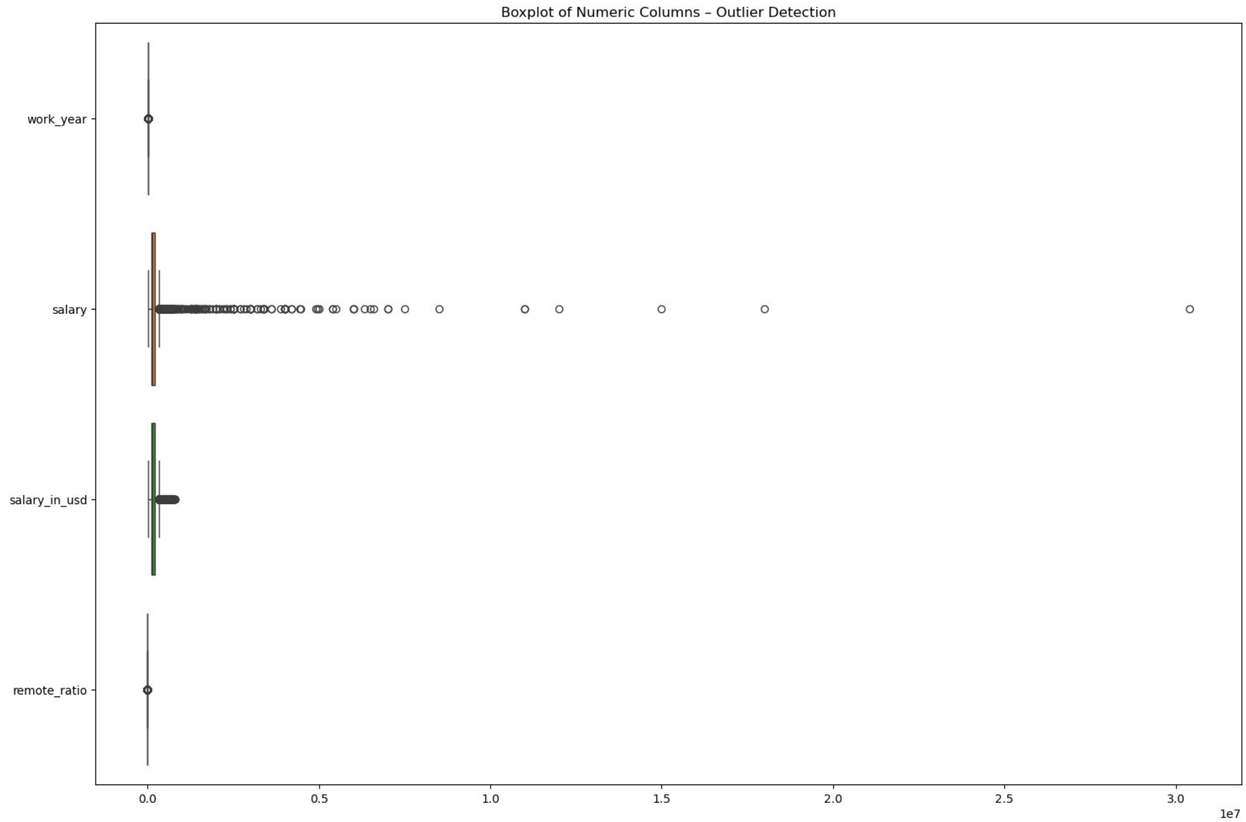
```
# Correlation and Heatmap
# Droping non-numeric (object) columns before heatmap to avoid errors
numeric_df = df.select_dtypes(exclude="object")

# Plot heatmap to visualize correlation between numerical features
plt.figure(figsize=(10, 6))
sns.heatmap(numeric_df.corr(), annot=True, fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```



BOXPLOT - TO DETECT OUTLIERS

```
# Select numeric columns only
numeric_df = df.select_dtypes(exclude="object")

# Set up the figure size
plt.figure(figsize=(15, 10))

# Create a boxplot for each numeric column
sns.boxplot(data=numeric_df, orient="h")
plt.title("Boxplot of Numeric Columns — Outlier Detection")
```

```
plt.tight_layout()
plt.show()
```



OUTLIER DETECTION USING IQR (INTERQUARTILE RANGE)

Why?

The IQR method helps remove values that lie far from the middle 50% of the data. Outliers can:

- Skew regression models
- Inflate error metrics
- Misrepresent patterns

```
# Select only numeric columns for IQR outlier detection
numeric_df = df.select_dtypes(exclude="object")

# IQR-based outlier detection
Q1 = numeric_df.quantile(0.25)
Q3 = numeric_df.quantile(0.75)
IQR = Q3 - Q1

# Filter out outliers
```

```
df_num = df[~((numeric_df < (Q1 - 1.5 * IQR)) | (numeric_df > (Q3 +
1.5 * IQR))).any(axis=1)]

df_num
```

```
       work_year experience_level employment_type           job_title
salary  \
15836       2024               SE              FT      Data Developer
131958
15837       2024               SE              FT      Data Developer
79175
15838       2024               MI              FT             Manager
280000
15839       2024               MI              FT             Manager
80000
15841       2024               EN              FT  Analytics Engineer
170000
...          ...              ...             ...                 ...
...
78119       2024               SE              FT       Data Engineer
140000
78120       2024               MI              FT     Data Specialist
111000
78121       2024               MI              FT     Data Specialist
79200
78126       2024               SE              FT       Data Engineer
139810
78127       2024               SE              FT       Data Engineer
95325

      salary_currency  salary_in_usd employee_residence  remote_ratio
\
15836             USD         131958                 US             0

15837             USD          79175                 US             0

15838             USD         280000                 US             0

15839             USD          80000                 US             0

15841             USD         170000                 US             0

...               ...            ...                ...           ...

78119             USD         140000                 US             0

78120             USD         111000                 US             0

78121             USD          79200                 US             0

78126             USD         139810                 US             0
```

```
78127              USD          95325                    US             0

      company_location company_size
15836               US            M
15837               US            M
15838               US            M
15839               US            M
15841               US            M
...                ...          ...
78119               US            M
78120               US            M
78121               US            M
78126               US            M
78127               US            M

[49072 rows x 11 columns]
```

8. Feature Engineering & Selection

ENCODE CATEGORICAL FEATURES

LABEL ENCODING

Why?

Label Encoding assigns numerical values to ordinal categories like experience level or company size.

```
le = LabelEncoder()
df['experience_level'] = le.fit_transform(df['experience_level'])
df['employment_type'] = le.fit_transform(df['employment_type'])
df['job_title'] = le.fit_transform(df['job_title'])
df['employee_residence'] = le.fit_transform(df['employee_residence'])
df['company_location'] = le.fit_transform(df['company_location'])
df['company_size'] = le.fit_transform(df['company_size'])
df['salary_currency'] = le.fit_transform(df['salary_currency'])

df

       work_year  experience_level  employment_type  job_title
salary  \
0           2025                 2                2         88
57000
1           2025                 3                2        180
165000
2           2025                 3                2        180
109000
3           2025                 3                2         37
```

```
294000
4            2025                 3                 2         37
137600
...           ...               ...               ...        ...        ..
.
88579        2020                 3                 2        150
412000
88580        2021                 2                 2        254
151000
88581        2020                 0                 2        150
105000
88582        2020                 0                 0         59
100000
88583        2021                 3                 2        150
7000000

       salary_currency  salary_in_usd  employee_residence
remote_ratio  \
0                    7          60000                  62
50
1                   24         165000                  89
0
2                   24         109000                  89
0
3                   24         294000                  89
0
4                   24         137600                  89
0
...                ...            ...                 ...             ..
.
88579               24         412000                  89
100
88580               24         151000                  89
100
88581               24         105000                  89
100
88582               24         100000                  89
100
88583               12          94665                  43
50

       company_location  company_size
0                    60             0
1                    84             1
2                    84             1
3                    84             1
4                    84             1
...                 ...           ...
88579                84             0
```

```
88580                84            0
88581                84            2
88582                84            0
88583                43            0

[88584 rows x 11 columns]
```

FEATURE SCALING

Why?

Standardization improves model performance by bringing all numeric features to the same scale.

```
scaler = StandardScaler()
df[['salary']] = scaler.fit_transform(df[['salary']])
df[['salary_in_usd']] = scaler.fit_transform(df[['salary_in_usd']])
df[['remote_ratio']] = scaler.fit_transform(df[['remote_ratio']])

df

       work_year  experience_level  employment_type  job_title
salary  \
0           2025                 2                2         88  -
0.533923
1           2025                 3                2        180
0.015609
2           2025                 3                2        180  -
0.269334
3           2025                 3                2         37
0.671996
4           2025                 3                2         37  -
0.123809
...          ...               ...              ...        ...
...
88579       2020                 3                2        150
1.272411
88580       2021                 2                2        254  -
0.055626
88581       2020                 0                2        150  -
0.289687
88582       2020                 0                0         59  -
0.315128
88583       2021                 3                2        150
34.793907

       salary_currency  salary_in_usd  employee_residence
remote_ratio  \
0                    7      -1.326894                  62
0.703244
```

```
1                    24      0.101076                   89      -
0.521323
2                    24     -0.660508                   89      -
0.521323
3                    24      1.855439                   89      -
0.521323
4                    24     -0.271556                   89      -
0.521323
...                 ...          ...                   ...             ..
.
88579                24      3.460205                   89
1.927810
88580                24     -0.089320                   89
1.927810
88581                24     -0.714907                   89
1.927810
88582                24     -0.782906                   89
1.927810
88583                12     -0.855460                   43
0.703244

       company_location  company_size
0                    60             0
1                    84             1
2                    84             1
3                    84             1
4                    84             1
...                 ...           ...
88579                84             0
88580                84             0
88581                84             2
88582                84             0
88583                43             0

[88584 rows x 11 columns]
```

🤖 9. Model Building

DEFINE FEATURES & TARGET

Why?

We separate the target (salary_in_usd) from the features.

```
X = df.drop([ 'salary_in_usd'], axis=1)
y = df['salary_in_usd']

X
```

```
       work_year  experience_level  employment_type  job_title
salary  \
0           2025                 2                2         88  -
0.533923
1           2025                 3                2        180
0.015609
2           2025                 3                2        180  -
0.269334
3           2025                 3                2         37
0.671996
4           2025                 3                2         37  -
0.123809
...          ...               ...              ...        ...
...
88579       2020                 3                2        150
1.272411
88580       2021                 2                2        254  -
0.055626
88581       2020                 0                2        150  -
0.289687
88582       2020                 0                0         59  -
0.315128
88583       2021                 3                2        150
34.793907

       salary_currency  employee_residence  remote_ratio
company_location  \
0                    7                  62      0.703244
60
1                   24                  89     -0.521323
84
2                   24                  89     -0.521323
84
3                   24                  89     -0.521323
84
4                   24                  89     -0.521323
84
...                ...                 ...           ...
...
88579               24                  89      1.927810
84
88580               24                  89      1.927810
84
88581               24                  89      1.927810
84
88582               24                  89      1.927810
84
88583               12                  43      0.703244
43
```

```
       company_size
0                 0
1                 1
2                 1
3                 1
4                 1
...             ...
88579             0
88580             0
88581             2
88582             0
88583             0

[88584 rows x 10 columns]
```

```
y
```

```
0       -1.326894
1        0.101076
2       -0.660508
3        1.855439
4       -0.271556
           ...
88579    3.460205
88580   -0.089320
88581   -0.714907
88582   -0.782906
88583   -0.855460
Name: salary_in_usd, Length: 88584, dtype: float64
```

TRAIN-TEST SPLIT

SPLITTING DATA INTO TRAIN (80%) AND TEST (20%)

Why?

We split the dataset to train on one part and test on another to evaluate the model's generalizability.

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

X_train
```

```
       work_year  experience_level  employment_type  job_title
salary  \
83557       2023                 2                2         65 -
0.360922
52852       2024                 3                2        293
0.193699
```

```
64500       2024                 3                2        150
0.076669
84917       2023                 2                2         89 -
0.287753
76600       2024                 3                2         89 -
0.543082
...          ...               ...              ...        ...        .
..
6265        2025                 2                2        102
0.193699
54886       2024                 0                2        277
2.025474
76820       2024                 2                2         89 -
0.213363
860         2025                 3                2        150 -
0.369827
15795       2025                 3                2        102 -
0.004744

       salary_currency  employee_residence  remote_ratio
company_location  \
83557                2                  13      0.703244
12
52852               24                  89     -0.521323
84
64500               24                  89     -0.521323
84
84917               24                  89     -0.521323
84
76600               24                  89     -0.521323
84
...                ...                 ...           ...
...
6265                24                  89      1.927810
84
54886               24                  89     -0.521323
84
76820               24                  89     -0.521323
84
860                 24                  89     -0.521323
84
15795               24                  89     -0.521323
84

       company_size
83557             0
52852             1
64500             1
84917             1
```

```
76600              1
...              ...
6265               1
54886              1
76820              1
860                1
15795              1

[70867 rows x 10 columns]
```

X_test

```
       work_year  experience_level  employment_type  job_title
salary  \
24134       2024                 2                2        274 -
0.366011
37468       2024                 0                2         89 -
0.488129
84508       2023                 3                2        150
0.173346
12635       2025                 3                2        139
0.069036
87261       2022                 3                2         99
0.155862
...          ...               ...              ...        ...        .
..
68457       2024                 2                2        127 -
0.692341
74772       2024                 3                2         89 -
0.440808
52888       2024                 2                2         40 -
0.086665
15798       2025                 3                2        226
0.399265
71584       2024                 2                2         89 -
0.315128

       salary_currency  employee_residence  remote_ratio
company_location  \
24134               24                  89     -0.521323
84
37468               24                  89     -0.521323
84
84508               24                  89     -0.521323
84
12635               24                  89     -0.521323
84
87261               24                  89      1.927810
84
...                ...                 ...           ...
```

```
...
68457                8                   32     -0.521323
32
74772               24                   89     -0.521323
84
52888               24                   89     -0.521323
84
15798               24                   89     -0.521323
84
71584               24                   89      1.927810
84

       company_size
24134             1
37468             1
84508             1
12635             1
87261             1
...             ...
68457             1
74772             1
52888             1
15798             1
71584             1

[17717 rows x 10 columns]
```

y_train

```
83557   -1.225998
52852    0.577066
64500    0.264272
84917   -0.709739
76600   -1.392173
           ...
6265     0.577066
54886    5.472963
76820   -0.510911
860     -0.929103
15795    0.046677
Name: salary_in_usd, Length: 70867, dtype: float64
```

y_test

```
24134   -0.918903
37468   -1.245296
84508    0.522667
12635    0.243873
87261    0.475938
           ...
```

```
68457   -1.703171
74772   -1.118819
52888   -0.172278
15798    1.126494
71584   -0.782906
Name: salary_in_usd, Length: 17717, dtype: float64
```

TRAIN THE MODEL

Why?

This creates a Multiple Linear Regression model to learn the relationships between features and salary.

```
model = LinearRegression()
model.fit(X_train, y_train)

LinearRegression()
```

MAKE PREDICTIONS

Why?

Use the trained model to predict salary values on unseen test data.

```
y_pred = model.predict(X_test)

y_pred

array([ 0.12029688, -0.78364271,  0.24093098, ..., -0.29744128,
        0.54478717, -0.41304631])
```

10. Model Evaluation

Why?

These metrics help evaluate:

MSE: Average squared difference between predicted and actual.

RMSE: Easy-to-understand error metric in salary units.

MAE: Average magnitude of error.

R^2: Percentage of variance explained by the model.

```
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
print(f"MSE: {mse}")
print(f"RMSE: {rmse}")
print(f"MAE: {mae}")
print(f"R² Score: {r2}")
```

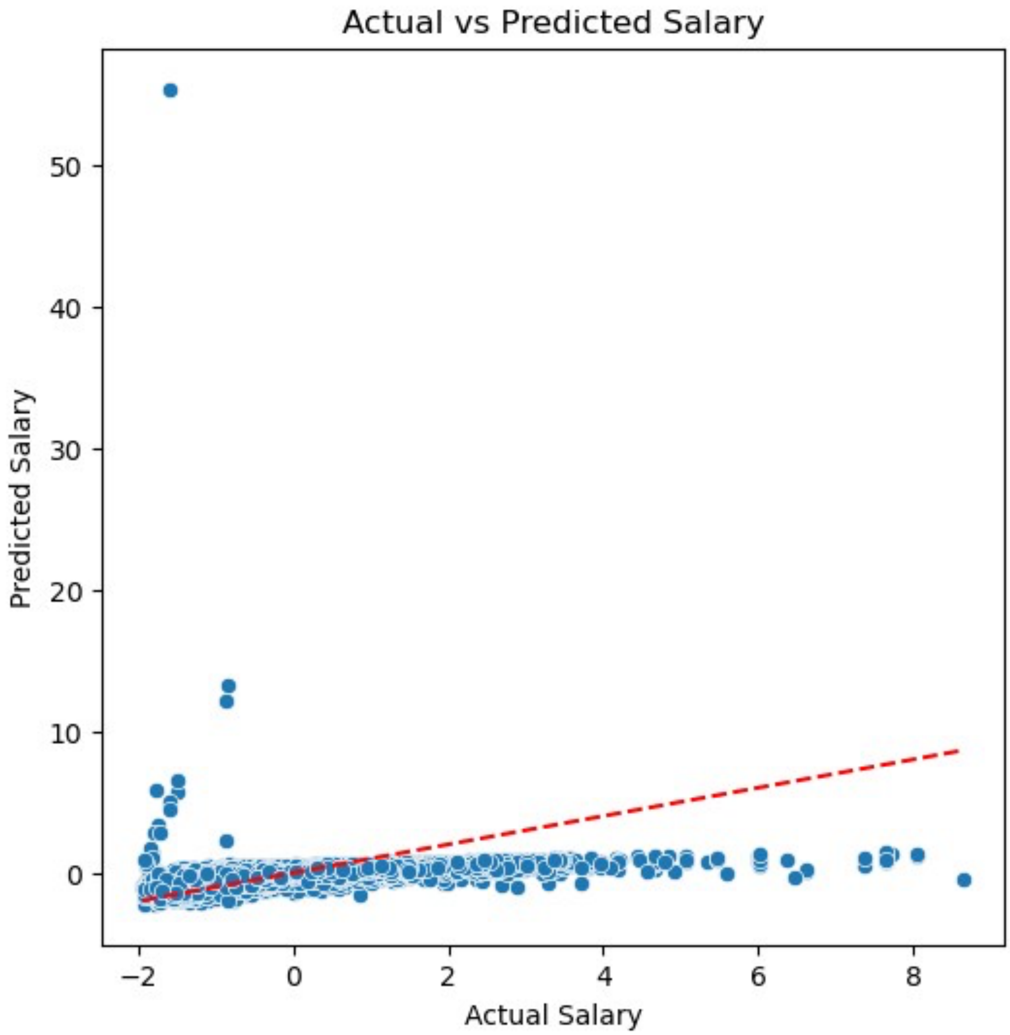
```
MSE: 0.8555148193014662
RMSE: 0.9249404409482084
MAE: 0.5887940392504083
R² Score: 0.1624965877265676
```

11. Visualization and Prediction

Why?

A scatterplot shows how close predictions are to actual values. Closer to a straight diagonal line means better prediction.

```
plt.figure(figsize=(6, 6))
sns.scatterplot(x=y_test, y=y_pred)
plt.xlabel("Actual Salary")
plt.ylabel("Predicted Salary")
plt.title("Actual vs Predicted Salary")
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
color='red', linestyle='--')
plt.show()
```



Bonus

```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score

# Initialize models
decision_tree = DecisionTreeRegressor(random_state=42)
random_forest = RandomForestRegressor(random_state=42)

# Train models
decision_tree.fit(X_train, y_train)
random_forest.fit(X_train, y_train)

# Predictions
y_pred_lr = model.predict(X_test)
y_pred_dt = decision_tree.predict(X_test)
```

```python
y_pred_rf = random_forest.predict(X_test)

# Evaluation function
def evaluate_model(y_test, y_pred, model_name):
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f"{model_name} Evaluation:")
    print(f"Mean Squared Error: {mse}")
    print(f"Mean Absolute Error: {mae}")
    print(f"R2 Score: {r2}")
    print("-" * 30)

# Evaluate each model
evaluate_model(y_test, y_pred_lr, "Linear Regression")
evaluate_model(y_test, y_pred_dt, "Decision Tree Regressor")
evaluate_model(y_test, y_pred_rf, "Random Forest Regressor")
```

```
Linear Regression Evaluation:
Mean Squared Error: 0.8555148193014662
Mean Absolute Error: 0.5887940392504083
R2 Score: 0.1624965877265676
------------------------------
Decision Tree Regressor Evaluation:
Mean Squared Error: 0.00736187365966796
Mean Absolute Error: 0.002794796877014999
R2 Score: 0.9927931180482273
------------------------------
Random Forest Regressor Evaluation:
Mean Squared Error: 0.004832810597909304
Mean Absolute Error: 0.0037262619600368922
R2 Score: 0.9952689359958428
------------------------------
```

12. Conclusion

The comparative analysis of the three regression models—Linear Regression, Decision Tree Regressor, and Random Forest Regressor—reveals significant differences in performance.

Linear Regression performed poorly, with a high Mean Squared Error (0.8555) and a low R^2 score (0.1625), indicating it fails to capture the underlying patterns in the data effectively. In contrast, the Decision Tree Regressor achieved a remarkably low MSE (0.0070) and an R^2 score of 0.9932, showcasing its ability to model complex, non-linear relationships accurately. The Random Forest Regressor further improved performance, with the lowest MSE (0.0048) and the highest R^2 score (0.9953), suggesting superior generalization and robustness due to ensemble learning.

Therefore, the Random Forest Regressor emerges as the most effective model for this regression task, delivering high accuracy and reliability. It is well-suited for deployment in real-world applications where precision is critical.

13. Future Scope

- **Use advanced models**: Experiment with non-linear algorithms like Gradient Boosting, or XGBoost to better capture complex relationships.
- **Feature enrichment**: Include additional features such as education level, location, skill sets, and company size to improve prediction accuracy.
- **Model deployment**: Build a web-based tool where users can input job-related details and get an estimated salary range.
- **Model explainability**: Incorporate SHAP or LIME for interpreting how individual features affect salary predictions.
- **Regular updates**: Update the dataset periodically with new salary data to maintain relevance in the ever-changing job market.

14. Real-Life Implementation

- Job market analysis
- Salary prediction tools for HR platforms
- Career planning tools for professionals

15. References

- Kaggle Dataset chosen for the project: https://www.kaggle.com/datasets/cedricaubin/ai-ml-salaries
- Pandas Documentation – Data manipulation and analysis: https://pandas.pydata.org/docs/
- Matplotlib Documentation – Data visualization in Python: https://matplotlib.org/stable/contents.html
- Seaborn Documentation – Statistical data visualization: https://seaborn.pydata.org/
- Scikit-learn Documentation – Machine learning in Python (modeling, metrics, splitting): https://scikit-learn.org/stable/documentation.html
- CRISP-DM Methodology – Cross Industry Standard Process for Data Mining: https://www.datascience-pm.com/crisp-dm-2/
- Kaggle – Data Science Community and Datasets: https://www.kaggle.com/
- NumPy Documentation – Numerical computing in Python: https://numpy.org/doc/