IMPORTING LIBRARIES

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.metrics import mean_squared_error, r2_score, accuracy_score, confusion_matrix, classification_report
```

DATA LOAD & OVERVIEW

```
df = pd.read_csv("/content/03_Data Science Salaries 2023 Analysis.csv")

print(df.head())
print(df.info())
print(df.describe())
```

```
   work_year experience_level employment_type                job_title  \
0       2023               SE              FT  Principal Data Scientist
1       2023               MI              CT               ML Engineer
2       2023               MI              CT               ML Engineer
3       2023               SE              FT           Data Scientist
4       2023               SE              FT           Data Scientist

   salary salary_currency  salary_in_usd employee_residence  remote_ratio  \
0   80000             EUR          85847                 ES           100
1   30000             USD          30000                 US           100
2   25500             USD          25500                 US           100
3  175000             USD         175000                 CA           100
4  120000             USD         120000                 CA           100

  company_location company_size
0               ES            L
1               US            S
2               US            S
3               CA            M
4               CA            M
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3755 entries, 0 to 3754
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   work_year           3755 non-null   int64
 1   experience_level    3755 non-null   object
 2   employment_type     3755 non-null   object
 3   job_title           3755 non-null   object
 4   salary              3755 non-null   int64
 5   salary_currency     3755 non-null   object
 6   salary_in_usd       3755 non-null   int64
 7   employee_residence  3755 non-null   object
 8   remote_ratio        3755 non-null   int64
 9   company_location    3755 non-null   object
 10  company_size        3755 non-null   object
dtypes: int64(4), object(7)
memory usage: 322.8+ KB
None
         work_year        salary  salary_in_usd  remote_ratio
count  3755.000000  3.755000e+03    3755.000000   3755.000000
mean   2022.373635  1.906956e+05  137570.389880     46.271638
std       0.691448  6.716765e+05   63055.625278     48.589050
min    2020.000000  6.000000e+03    5132.000000      0.000000
25%    2022.000000  1.000000e+05   95000.000000      0.000000
50%    2022.000000  1.380000e+05  135000.000000      0.000000
75%    2023.000000  1.800000e+05  175000.000000    100.000000
max    2023.000000  3.040000e+07  450000.000000    100.000000
```

DATA CLEANING AND PREPROCESSING

```
df.dropna(inplace=True)
```

RETAIN ORIGINAL COMPANY_LOCATION FOR PLOTTING

```
company_location_original = df['company_location'].copy()
```

CONVERTING CATEGORICAL DATA TO NUMERICAL USING ONEHOTENCODER

```
categorical_columns = ['experience_level', 'employment_type', 'job_title', 'salary_currency', 'employee_residence', 'company_location', 'com
df = pd.get_dummies(df, columns=categorical_columns, drop_first=True)
```

DESCRIPTIVE STATISTICS

```
print(df.describe())
```

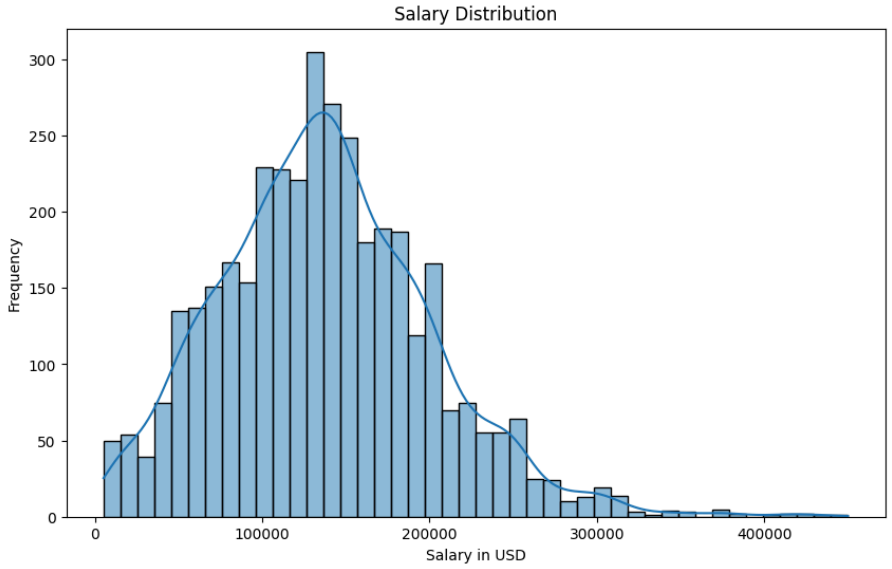
```
          work_year        salary  salary_in_usd  remote_ratio
count  3755.000000  3.755000e+03    3755.000000   3755.000000
mean   2022.373635  1.906956e+05  137570.389880     46.271638
std       0.691448  6.716765e+05   63055.625278     48.589050
min    2020.000000  6.000000e+03    5132.000000      0.000000
25%    2022.000000  1.000000e+05   95000.000000      0.000000
50%    2022.000000  1.380000e+05  135000.000000      0.000000
75%    2023.000000  1.800000e+05  175000.000000    100.000000
max    2023.000000  3.040000e+07  450000.000000    100.000000
```

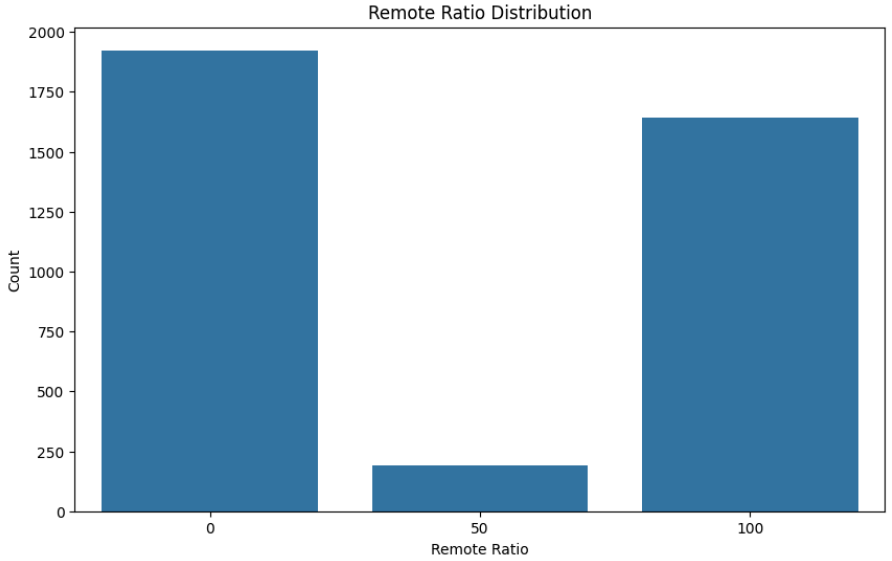
DATA VISUALIZATION

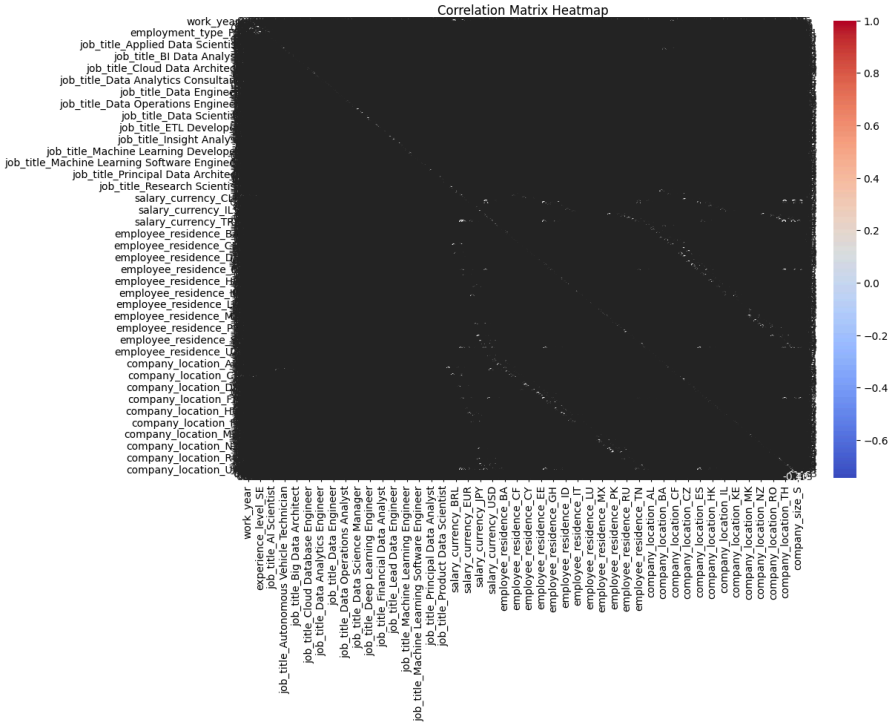
```
# Salary Distribution
plt.figure(figsize=(10, 6))
sns.histplot(df['salary_in_usd'], kde=True)
plt.title('Salary Distribution')
plt.xlabel('Salary in USD')
plt.ylabel('Frequency')
plt.show()

# Remote Ratio Distribution
plt.figure(figsize=(10, 6))
sns.countplot(x='remote_ratio', data=df)
plt.title('Remote Ratio Distribution')
plt.xlabel('Remote Ratio')
plt.ylabel('Count')
plt.show()

# Heatmap for Correlation Matrix
plt.figure(figsize=(12, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('Correlation Matrix Heatmap')
plt.show()
```


Salary Distribution
Frequency
Salary in USD
0
50
100
150
200
250
300
0
100000
200000
300000
400000


Remote Ratio Distribution
Count
Remote Ratio
0
250
500
750
1000
1250
1500
1750
2000
0
50
100


Correlation Matrix Heatmap
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6

MACHINE LEARNING IMPLEMENTATION

PREDICTING SALARY (LINEAR REGRESSION)

```
X = df.drop(columns=['salary_in_usd'])
y = df['salary_in_usd']

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

# Feature Scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Training Linear Regression Model
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)

# Predicting and Evaluating Linear Regression Model
y_pred = lin_reg.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Linear Regression Mean Squared Error: {mse}')
print(f'Linear Regression R-squared: {r2}')
```

```
Linear Regression Mean Squared Error: 1.7323922885863813e+37
Linear Regression R-squared: -4.388258648001202e+27
```

PREDICTING EMPLOYMENT TYPE (LOGISTIC REGRESSION)

```
median_salary = df['salary_in_usd'].median()
df['above_median_salary'] = (df['salary_in_usd'] > median_salary).astype(int)

X_log = df.drop(columns=['salary_in_usd', 'above_median_salary'])
y_log = df['above_median_salary']

# Splitting the data into training and testing sets
X_log_train, X_log_test, y_log_train, y_log_test = train_test_split(X_log, y_log, test_size=0.2, random_state=42)

# Feature Scaling
X_log_train = scaler.fit_transform(X_log_train)
X_log_test = scaler.transform(X_log_test)

# Training Logistic Regression Model
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_log_train, y_log_train)

# Predicting and Evaluating Logistic Regression Model
y_log_pred = log_reg.predict(X_log_test)
accuracy = accuracy_score(y_log_test, y_log_pred)
print(f'Logistic Regression Accuracy: {accuracy}')
print('Confusion Matrix:')
print(confusion_matrix(y_log_test, y_log_pred))
print('Classification Report:')
print(classification_report(y_log_test, y_log_pred))
```

```
Logistic Regression Accuracy: 0.8242343541944075
Confusion Matrix:
[[331  84]
 [ 48 288]]
Classification Report:
              precision    recall  f1-score   support
```

ADVANCED ANALYSIS (GEOGRAPHICAL INSIGHTS)

```
# Analyzing the distribution of salaries across different company locations
plt.figure(figsize=(14, 7))
sns.boxplot(x=company_location_original, y='salary_in_usd', data=df)
plt.title('Salary Distribution by Company Location')
plt.xlabel('Company Location')
plt.ylabel('Salary in USD')
plt.xticks(rotation=90)
plt.show()
```