Importing some libraries as it is important to perform the opreation

```
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.preprocessing import StandardScaler, LabelEncoder
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.metrics import accuracy_score, mean_squared_error
```

We can take any kind of file like CSV, XLSX, json,...

Read the file and load the into data frame.

```
# Step 1: Load the dataset and get an overview
df = pd.read_csv('ds_salaries.csv')
```

```
print(df.head())
```

```
   Unnamed: 0  work_year experience_level employment_type  \
0           0       2020               MI              FT
1           1       2020               SE              FT
2           2       2020               SE              FT
3           3       2020               MI              FT
4           4       2020               SE              FT

                    job_title  salary salary_currency  salary_in_usd  \
0              Data Scientist   70000             EUR          79833
1  Machine Learning Scientist  260000             USD         260000
2           Big Data Engineer   85000             GBP         109024
3        Product Data Analyst   20000             USD          20000
4   Machine Learning Engineer  150000             USD         150000

  employee_residence  remote_ratio company_location company_size
0                 DE             0               DE            L
1                 JP             0               JP            S
2                 GB            50               GB            M
3                 HN             0               HN            S
4                 US            50               US            L
```

```
#Using .info print some basic information of the datasets.
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 607 entries, 0 to 606
Data columns (total 12 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Unnamed: 0          607 non-null    int64 
 1   work_year           607 non-null    int64 
 2   experience_level    607 non-null    object
 3   employment_type     607 non-null    object
 4   job_title           607 non-null    object
 5   salary              607 non-null    int64 
 6   salary_currency     607 non-null    object
 7   salary_in_usd       607 non-null    int64 
 8   employee_residence  607 non-null    object
 9   remote_ratio        607 non-null    int64 
 10  company_location    607 non-null    object
 11  company_size        607 non-null    object
dtypes: int64(5), object(7)
memory usage: 57.0+ KB
None
```

```
# Step 2: Check for missing values and outliers
print(df.isnull().sum())
```

```
Unnamed: 0            0
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
```

```
# Step 3: Check skewness of numerical features
skewness = df.select_dtypes(include=[np.number]).skew()
print(skewness)
```

```
Unnamed: 0        0.000000
work_year        -0.735817
salary           14.052915
salary_in_usd     1.667545
remote_ratio     -0.904224
dtype: float64
```

```
# Step 4: Check skewness of numerical features
skewness = df.select_dtypes(include=[np.number]).skew()
print(skewness)
```

```
Unnamed: 0        0.000000
work_year        -0.735817
salary           14.052915
salary_in_usd     1.667545
remote_ratio     -0.904224
dtype: float64
```

```
# Step 5: Data preprocessing and feature encoding
# Assume 'salary_currency' and 'employee_residence' are categorical columns
label_encoder = LabelEncoder()
df['salary_currency'] = label_encoder.fit_transform(df['salary_currency'])
df['employee_residence'] = label_encoder.fit_transform(df['employee_residence'])
```

```
# Step 6: Feature encoding and split
# Assuming 'salary' is the target variable
X = df.drop('salary', axis=1)
y = df['salary']
```

```
# Handle categorical columns using one-hot encoding
X_encoded = pd.get_dummies(X, drop_first=True)
```

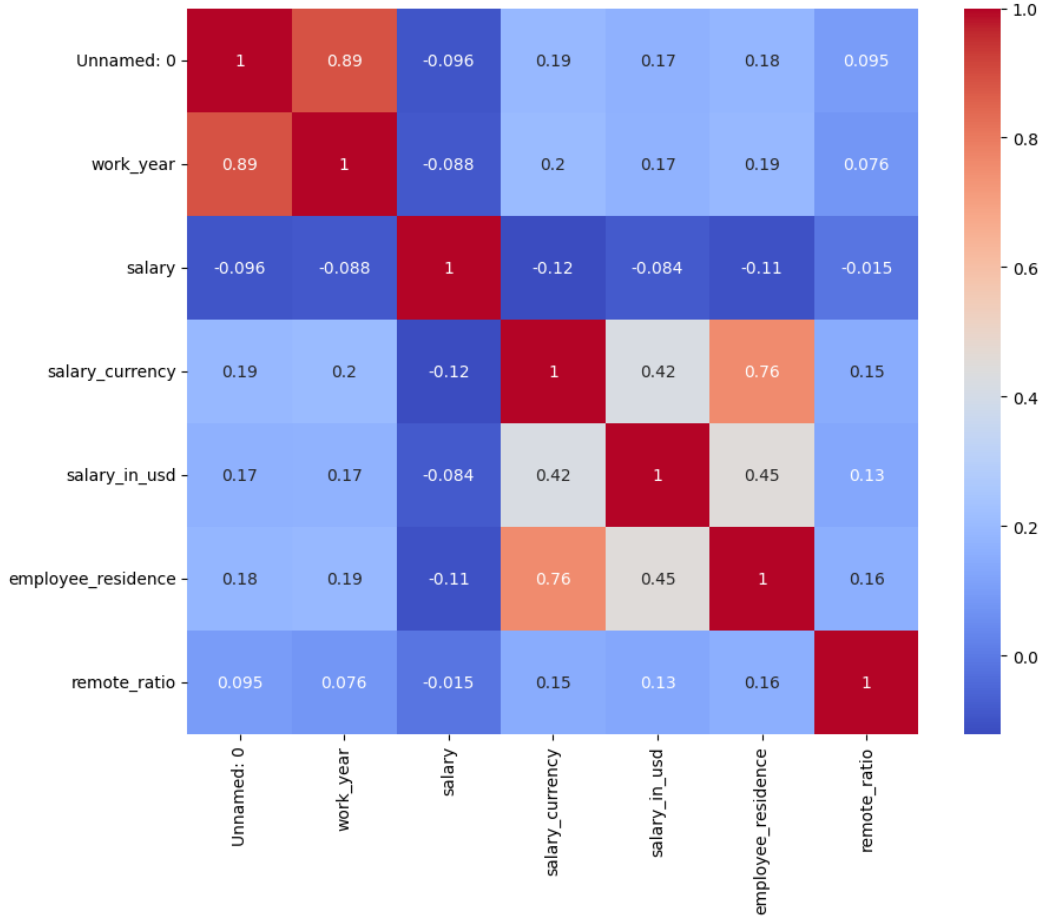
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_encoded, y, test_size=0.2, random_state=42)
```

```
# Feature scaling for numerical columns only
num_cols = X_encoded.select_dtypes(include=[np.number]).columns
scaler = StandardScaler()
X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()
X_train_scaled[num_cols] = scaler.fit_transform(X_train[num_cols])
X_test_scaled[num_cols] = scaler.transform(X_test[num_cols])
```

```
# Step 7: Visualize data
# Scatter plot
sns.scatterplot(x='work_year', y='salary', data=df)
plt.show()
```


```
# Heatmap to visualize correlation between numerical features
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.show()
```

```
<ipython-input-40-1967ccc99ac8>:3: FutureWarning: The default value of numeric_only in DataFrame.corr i
  sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```




```
# Step 8: Model Classification
# Initialize and train the classifier (Random Forest Classifier)
classifier = RandomForestClassifier(random_state=42)
classifier.fit(X_train_scaled, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

```
# Make predictions on the test set
y_pred = classifier.predict(X_test_scaled)
```

```
# Evaluate the classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.10655737704918032
```

```
# Step 9: Model Regression
# Initialize and train the regressor (Random Forest Regressor)
regressor = RandomForestRegressor(random_state=42)
regressor.fit(X_train_scaled, y_train)
```

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
# Make predictions on the test set
y_pred = regressor.predict(X_test_scaled)


# Evaluate the regressor
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 8021244042723.421
```

✓ 0s completed at 2:26 PM