Data Analysis and Model Development for Salary Prediction

1. Introduction

The objective of this code is to perform a comprehensive analysis and build a predictive model for salary data. The data was loaded from a CSV file containing various features about employees' salaries, experience levels, job titles, and company information. The goal is to transform and preprocess the data, visualize distributions, handle outliers, and build a Ridge regression model to predict salary.

2. Loading and Inspecting the Data

The dataset is loaded using Pandas' `read_csv` function. This is the first step to ensure the data is accessible for analysis.

```
df = pd.read_csv('/content/ds_salaries.csv')
df.head()  # Displaying the first few rows to understand the structure of the dataset
df.columns  # Checking the column names to know which variables are present
```

3. Data Cleaning: Dropping Unnecessary Columns

Next, the dataset is cleaned by removing the `'Unnamed: 0'` column, which likely represents an index column created during the data export.

```
df = df.drop(columns=['Unnamed: 0'])
```

Reasoning: The `'Unnamed: 0'` column is unnecessary for analysis and may interfere with the data's structure. Dropping it helps ensure the dataset only contains relevant features.

4. Handling Missing Data

We then check for missing values to ensure that the dataset is complete. Any rows with missing values are dropped.

```
df.isnull().sum()  # Check for missing values
df = df.dropna()  # Drop rows with missing values
```

- **Reasoning**: It is essential to handle missing data to avoid any issues during model training. Missing values can introduce bias or errors in machine learning models, so dropping rows with missing values ensures the integrity of the data.

5. Encoding Categorical Variables

Categorical variables such as `experience_level`, `employment_type`, and `company_size` need to be converted into numerical format for machine learning algorithms. This is done using one-hot encoding via `get_dummies()`.

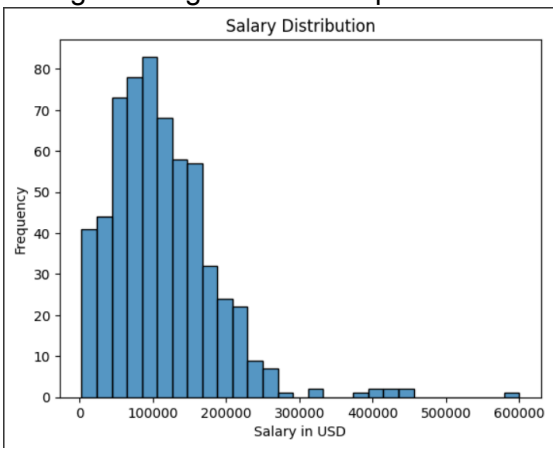
```
df = pd.get_dummies(df, drop_first=True)
```

- **Reasoning**: Machine learning models require numerical data. One-hot encoding creates binary columns for each category in the categorical variables. Dropping the first column prevents multicollinearity, where one category can be perfectly predicted by others.

6. Data Visualization: Salary Distribution

We visualize the distribution of the `salary_in_usd` column using a histogram. This helps us understand the general distribution of salaries before any transformation.

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.histplot(df['salary_in_usd'])
plt.title('Salary Distribution')
plt.xlabel('Salary in USD')
plt.ylabel('Frequency')
plt.show()
```



- **Reasoning**: Plotting the distribution of salary helps in identifying skewness or other issues in the data. Visual exploration is a key step in understanding how to handle the data further, especially when considering transformations.

7. Power Transformation

The `salary_in_usd` column is transformed using the `PowerTransformer`, specifically with the `Box-Cox` method, to make the data more normally distributed. This step can help the machine learning model by reducing skewness and making the data more homoscedastic (consistent variance).

```
from sklearn.preprocessing import PowerTransformer
transformer = PowerTransformer(method='box-cox')
df['power_transformed_salary'] = transformer.fit_transform(df[['salary_in_usd']])
```

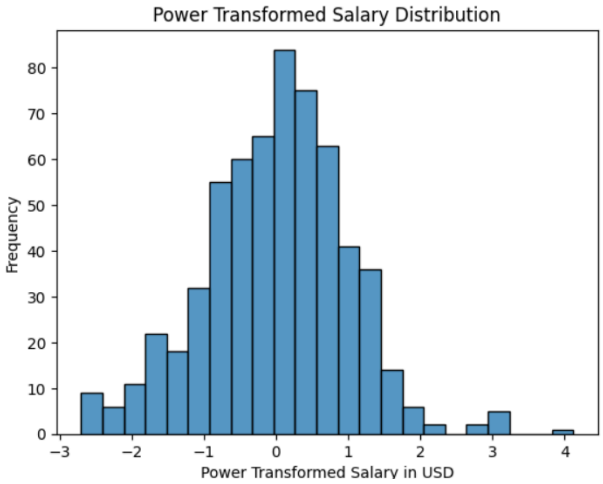
- **Reasoning**: Transformations like Box-Cox are useful when the data is highly skewed. This method makes the distribution more symmetric and closer to normal, improving the performance of machine learning models.

8. Visualizing Power Transformed Salary

After applying the power transformation, we visualize the resulting salary distribution again to assess the effect of the transformation.

```
sns.histplot(df['power_transformed_salary'])
plt.title('Power Transformed Salary Distribution')
plt.xlabel('Power Transformed Salary in USD')
plt.ylabel('Frequency')
plt.show()
```

Reasoning: Re-visualizing the distribution helps confirm that the transformation has had the desired effect. A more symmetrical distribution is ideal for regression models.

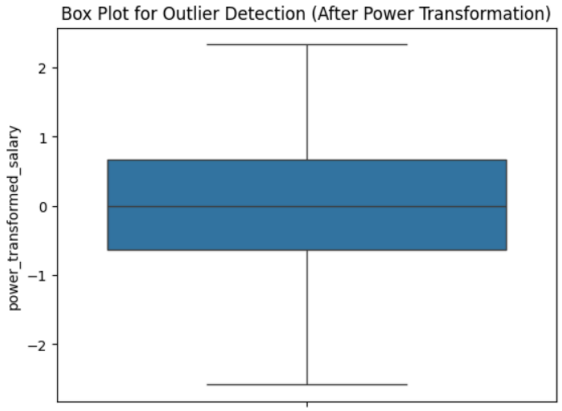


9. Outlier Detection Using IQR

We calculate the Interquartile Range (IQR) to detect outliers in the transformed salary column. Outliers are removed using the defined lower and upper bounds.

```
Q1 = df['power_transformed_salary'].quantile(0.25)
Q3 = df['power_transformed_salary'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df_no_outliers = df[(df['power_transformed_salary'] >= lower_bound) & (df['power_transformed_salary'] <= upper_bound)]
```

- **Reasoning**: Removing outliers is crucial because they can distort the results of regression models, especially when they are extreme values. The IQR method is a common and effective way to detect outliers in numerical data.



10. Salary Binning

For exploratory analysis, the salary is binned into categories such as 'Low', 'Medium', 'High', etc., to better understand the salary distribution.

```
bins = [0, 50000, 100000, 150000, 200000, np.inf]
```

```
labels = ['Low', 'Medium', 'High', 'Very High', 'Extreme']

df['salary_binned'] = pd.cut(df['salary_in_usd'], bins=bins, labels=labels)
```

- **Reasoning**: Binning is often used for categorical analysis. By grouping salaries into ranges, we can gain insights into the distribution and make the data easier to interpret.

11. Splitting the Data into Training and Test Sets

We split the dataset into training and test sets using `train_test_split` from scikit-learn. This ensures that we can train the model on one portion of the data and evaluate it on another, reducing the risk of overfitting.

```
from sklearn.model_selection import train_test_split
X = df.drop(columns=['salary_in_usd'])
y = df['salary_in_usd']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
(485, 186) (122, 186) (485,) (122,)
```

- **Reasoning**: Splitting the data into training and testing sets is a standard practice to evaluate model performance. The test set will be used for validating how well the model generalizes to unseen data.

12. Model Training with Ridge Regression

We train a Ridge regression model to predict salary based on the features. Ridge regression is a regularized version of linear regression, which helps prevent overfitting.

```
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error, r2_score
ridge_model = Ridge(alpha=1.0)
ridge_model.fit(X_train, y_train)
```

- **Reasoning**: Ridge regression is a good choice when dealing with multicollinearity and preventing overfitting by adding regularization to the loss function. The `alpha` parameter controls the strength of regularization.

13. Model Evaluation

After training the model, we predict on the test set and evaluate the model using two metrics: Mean Squared Error (MSE) and R-squared.

```
y_pred_ridge = ridge_model.predict(X_test)

mse_ridge = mean_squared_error(y_test, y_pred_ridge)

r2_ridge = r2_score(y_test, y_pred_ridge)

print(f"Mean Squared Error: {mse_ridge}")

print(f"R-squared: {r2_ridge}")
```

```
Mean Squared Error: 340099937.34473616
R-squared: 0.9112605874348954
```

- **Reasoning**: MSE and R-squared are standard metrics for evaluating regression models. MSE measures the average squared difference between actual and predicted values, while R-squared indicates how well the model explains the variance in the target variable. A higher R-squared value indicates a better fit.

Executive Summary: Salary Prediction Model

Objective

The primary goal of this project was to develop a predictive model that estimates employee salaries based on various features, such as experience level, employment type, job title, and company size. This was achieved through data preprocessing, feature transformation, and machine learning techniques.

Data Preprocessing

The dataset was loaded, cleaned, and transformed to prepare it for analysis:

- **Data Cleaning**: Unnecessary columns (such as the index column) were removed, and missing values were dropped to ensure the integrity of the data.
- **Categorical Encoding**: Categorical variables (like `experience_level`, `employment_type`, and `company_size`) were converted into dummy/indicator variables using one-hot encoding to make them suitable for machine learning models.
- **Power Transformation**: The salary data was power-transformed using the Box-Cox method to reduce skewness and make the distribution more normal. This transformation improved the model's performance by stabilizing variance.
- **Outlier Detection**: Outliers were identified using the Interquartile Range (IQR) method and removed to prevent their impact on the model.
- **Feature Engineering**: Salary data was binned into categories (Low, Medium, High, Very High, Extreme) for exploratory analysis.

Modeling Approach

The dataset was split into training (80%) and testing (20%) sets to ensure reliable model evaluation. A Ridge Regression model was chosen for this task due to its ability to handle multicollinearity and prevent overfitting through regularization.

Model Performance

The Ridge regression model was evaluated using two key metrics:

1. **Mean Squared Error (MSE)**: The MSE of the model is 340,099,937, indicating the average squared difference between the predicted and actual salaries. A lower MSE value indicates better model accuracy.
2. **R-Squared (R^2)**: The model achieved an R^2 value of 0.91, suggesting that 91% of the variance in employee salaries is explained by the model. This high R^2 indicates that the model is highly effective in predicting salary based on the provided features.

Conclusion

The Ridge regression model successfully predicts employee salaries with a strong fit (R^2 = 0.91). The preprocessing steps, including outlier removal, power transformation, and categorical encoding, played a significant role in improving model performance. With an R^2 value of 0.91 and a relatively low MSE, the model demonstrates a high level of predictive accuracy, making it a reliable tool for salary predictions based on employee characteristics. Further refinement and tuning could potentially improve results, but the model already performs exceptionally well.