import pandas as pd

data=pd.read\_csv("ds\_salaries.csv")

data.head()

data.shape

data.info()

data.isnull().sum()

data.job\_title.unique()

data.describe()

from sklearn.preprocessing import LabelEncoder

column = "job\_title"

data2=data.copy()

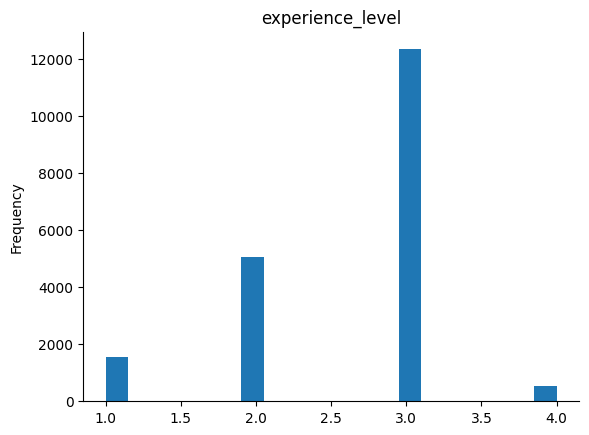
data[column] = LabelEncoder().fit\_transform(data[column])

from matplotlib import pyplot as plt

import seaborn as sns

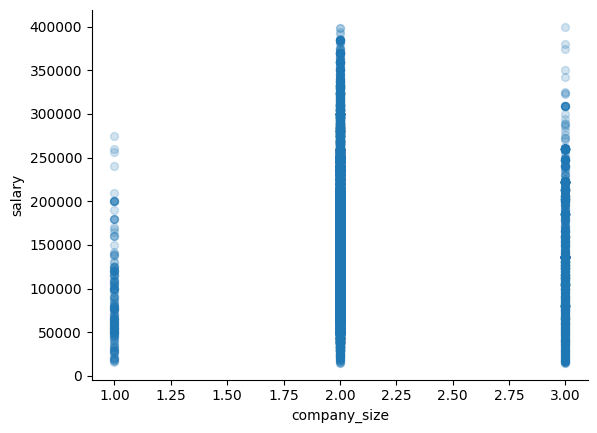
data['experience\_level'].plot(kind='hist', bins=20, title='experience\_level')

plt.gca().spines[['top', 'right',]].set\_visible(False)



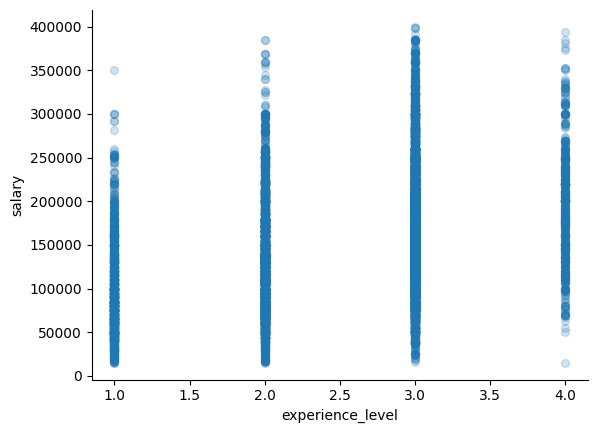
data.plot(kind='scatter', x='company\_size', y='salary', s=32, alpha=.2)

plt.gca().spines[['top', 'right',]].set\_visible(False)



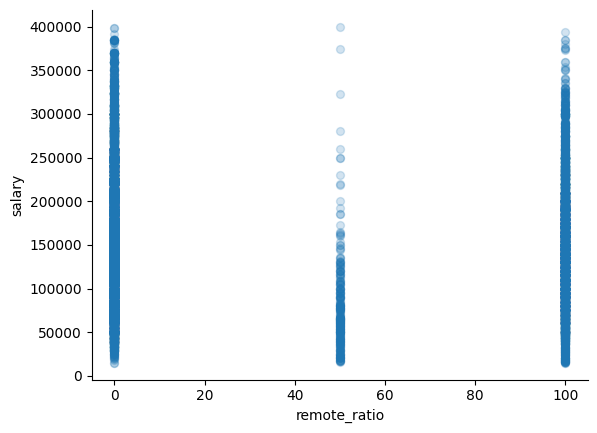
data.plot(kind='scatter', x='experience\_level', y='salary', s=32, alpha=.2)

plt.gca().spines[['top', 'right',]].set\_visible(False)



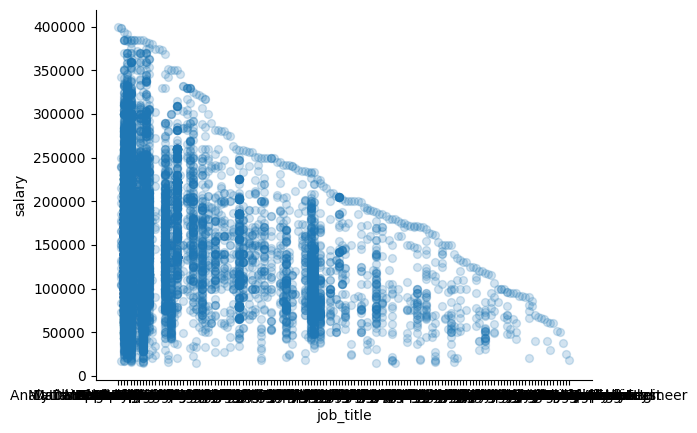
data.plot(kind='scatter', x='remote\_ratio', y='salary', s=32, alpha=.2)

plt.gca().spines[['top', 'right',]].set\_visible(False)



data2.plot(kind='scatter', x='job\_title', y='salary', s=32, alpha=.2)

plt.gca().spines[['top', 'right',]].set\_visible(False)



plt.figure(figsize=(12, 6))

ax = plt.gca()

ax.set\_facecolor('#f0f0f0')

sns.lineplot(x='company\_size', y='salary', hue='experience\_level', data=data, palette=['limegreen', 'orange', 'royalblue', 'red'])

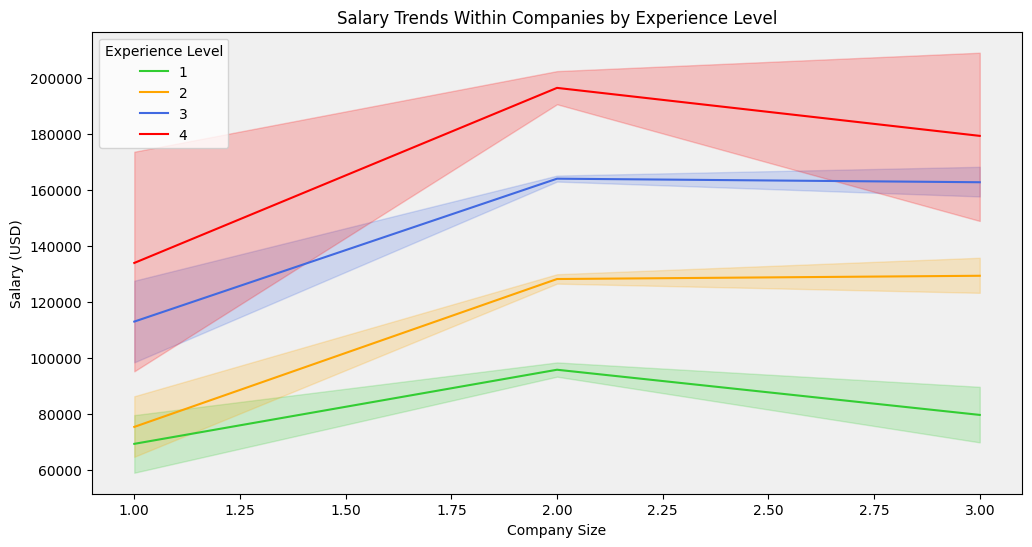
plt.title('Salary Trends Within Companies by Experience Level')

plt.xlabel('Company Size')

plt.ylabel('Salary (USD)')

plt.legend(title='Experience Level', loc='upper left')

plt.show()



plt.figure(figsize=(12, 6))

ax = plt.gca()

ax.set\_facecolor('#f0f0f0')

sns.lineplot(x='work\_year', y='salary', hue='experience\_level', data=data, palette=['limegreen', 'orange', 'royalblue', 'red'])

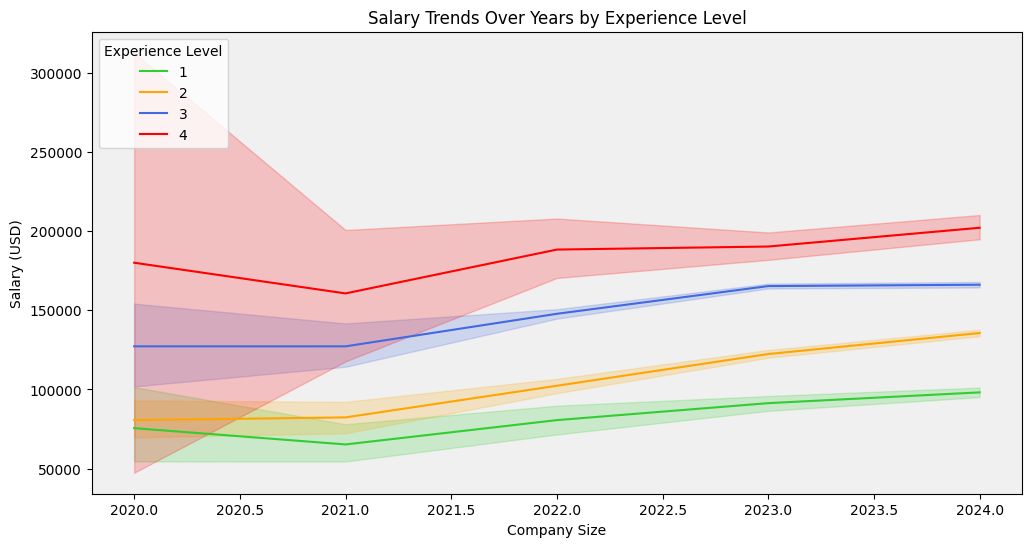
plt.title('Salary Trends Over Years by Experience Level')

plt.xlabel('Company Size')

plt.ylabel('Salary (USD)')

plt.legend(title='Experience Level', loc='upper left')

plt.show()



plt.figure(figsize=(12, 6))

avg\_salary\_by\_title = data2.groupby('job\_title')['salary'].mean().sort\_values(ascending=False)[:10]

avg\_salary\_by\_title.plot(kind='bar', color=['orange','red'], alpha=0.8)

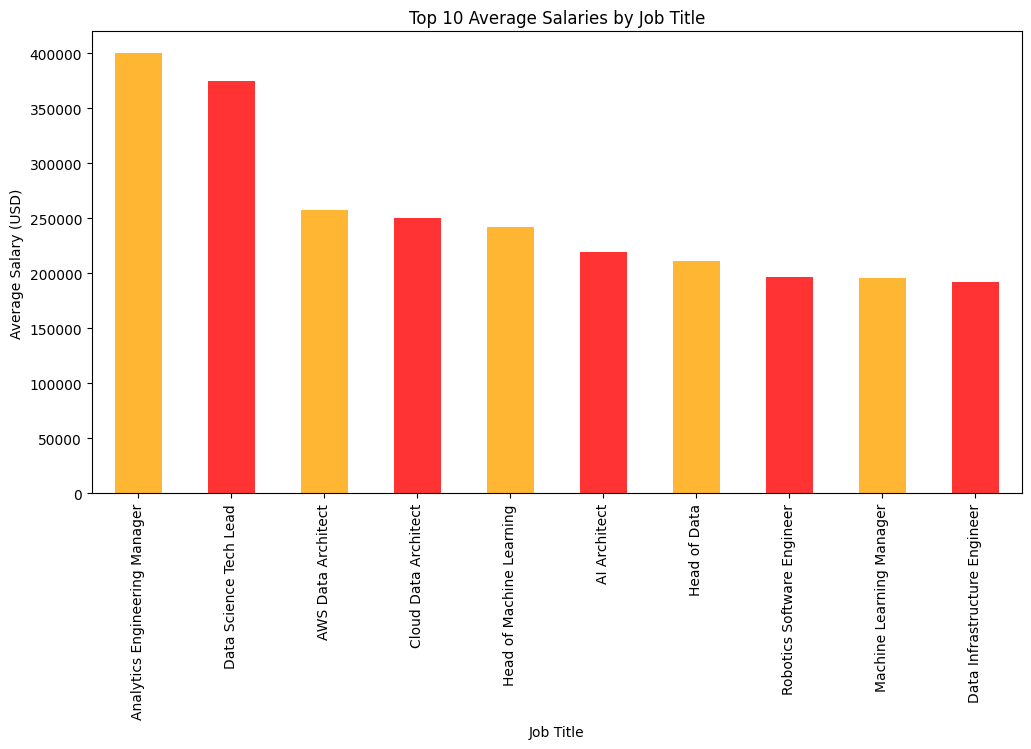
plt.title('Top 10 Average Salaries by Job Title')

plt.xlabel('Job Title')

plt.ylabel('Average Salary (USD)')

plt.xticks(rotation=90)

plt.show()



plt.figure(figsize=(10, 6))

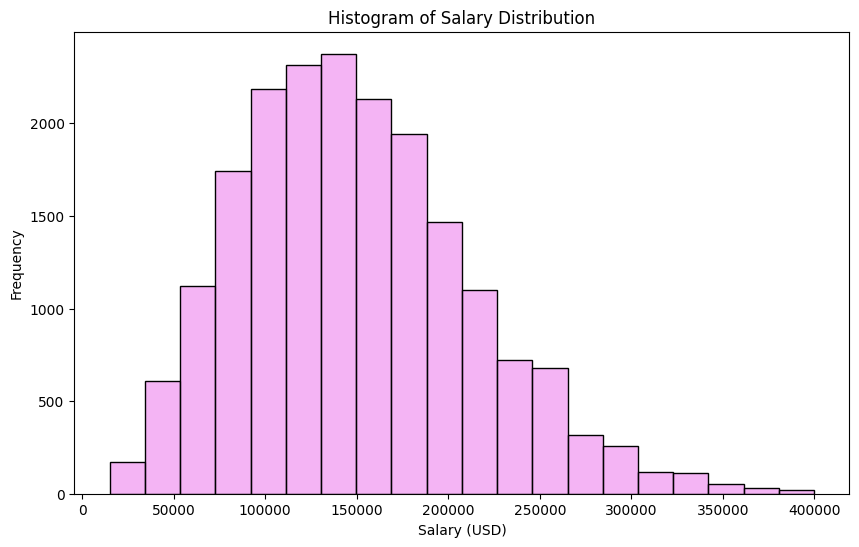
sns.histplot(data['salary'], bins=20, color='violet', alpha=0.6)

plt.title('Histogram of Salary Distribution')

plt.xlabel('Salary (USD)')

plt.ylabel('Frequency')

plt.show()



import plotly.express as px

def draw\_remote\_donut\_chart(remote\_status):

    remote\_counts = remote\_status.value\_counts()

    labels = ['Fully Remote', 'Hybrid', 'In Office']

    values = [remote\_counts.get(100), remote\_counts.get(50), remote\_counts.get(0)]

    fig = px.pie(names=labels, values=values, hole=0.3, title='Remote Status Distribution')

    colors = ['#1f77b4', '#ff7f0e', '#2ca02c']

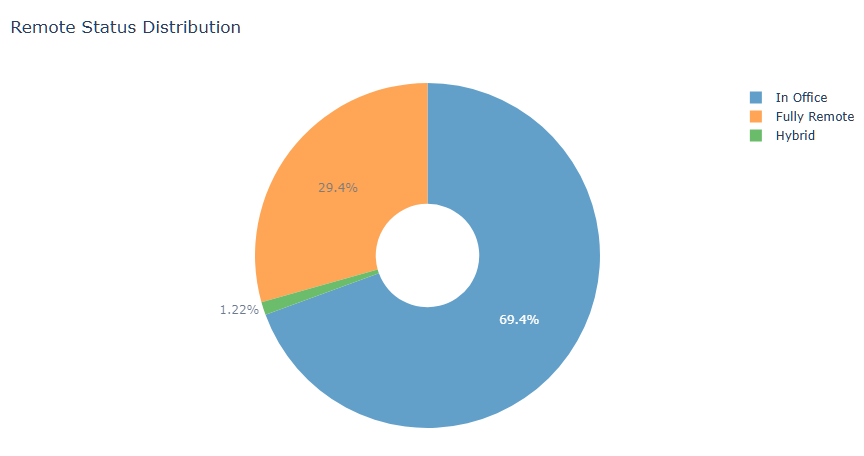
    fig = px.pie(names=labels, values=values, hole=0.3, title='Remote Status Distribution',

                 color\_discrete\_sequence=colors, opacity=0.7)

    fig.show()

remote\_status = data["remote\_ratio"]

draw\_remote\_donut\_chart(remote\_status)



import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestRegressor

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

X = data.drop(columns=["salary"])

y = data["salary"]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25)

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

model = RandomForestRegressor(n\_estimators=1000, max\_depth=10, min\_samples\_split=5, min\_samples\_leaf=2)

model.fit(X\_train\_scaled, y\_train)

y\_test\_pred = model.predict(X\_test\_scaled)

mae = mean\_absolute\_error(y\_test, y\_test\_pred)

print(f"Mean Absolute Error (MAE): {mae}")

mse = mean\_squared\_error(y\_test, y\_test\_pred)

print(f"Mean Squared Error (MSE): {mse}")

rmse = np.sqrt(mse)

print(f"Root Mean Squared Error (RMSE): {rmse}")

r2 = r2\_score(y\_test, y\_test\_pred)

print(f"R-squared (R²): {r2}")

error\_rate = abs(y\_test - y\_test\_pred)/y\_test

accuracy = 1 - error\_rate

X\_test["salary"] = y

X\_test["prediction"] = y\_test\_pred

X\_test["accuracy"] = accuracy

X\_test["error\_rate"] = error\_rate

X\_test

overall\_accuracy = X\_test["accuracy"].mean()

print(f"overall accuracy: {overall\_accuracy}")

feature\_importances = model.feature\_importances\_

features = X.columns

plt.figure(figsize=(12, 6))

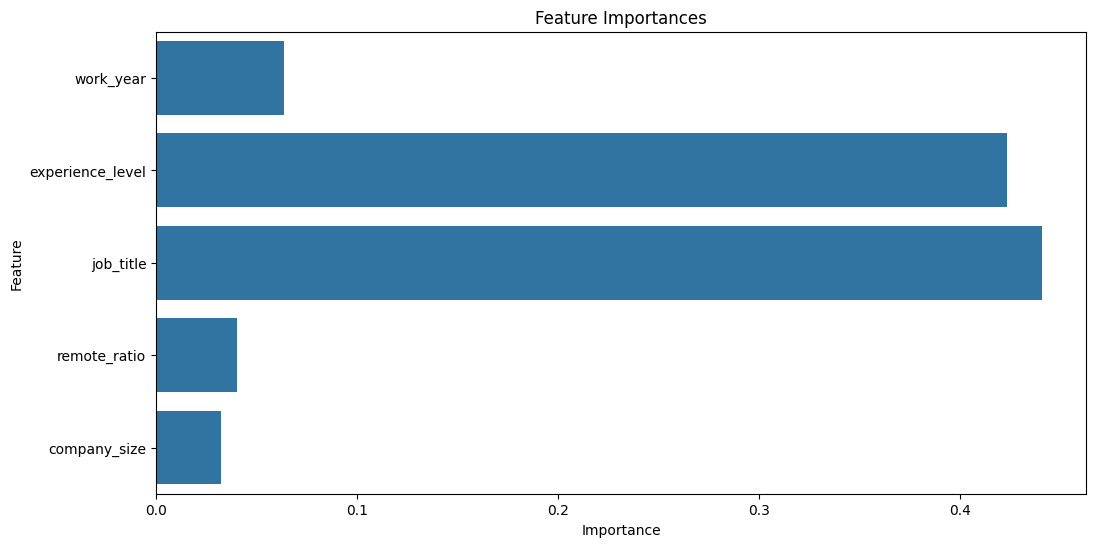
sns.barplot(x=feature\_importances, y=features)

plt.title("Feature Importances")

plt.xlabel("Importance")

plt.ylabel("Feature")

plt.show()



plt.figure(figsize=(10, 6))

plt.scatter(y\_test, y\_test\_pred, alpha=0.3)

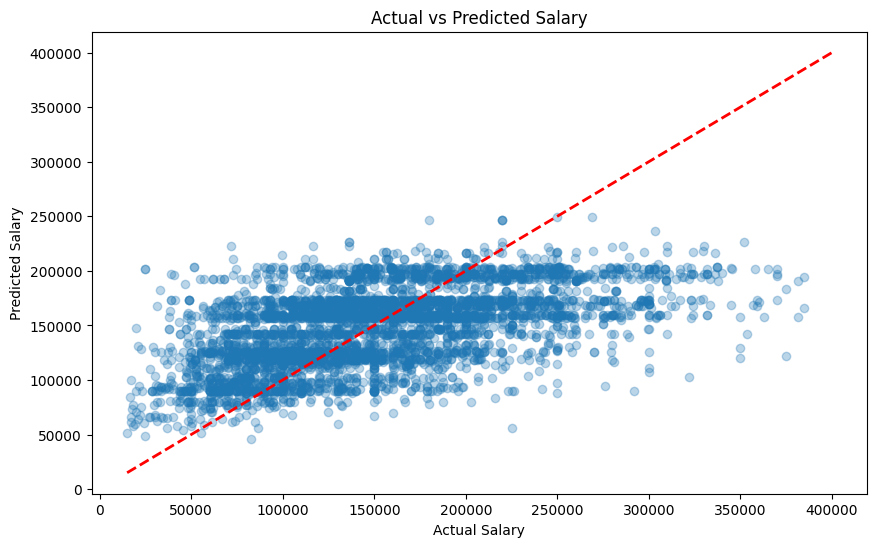
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'r--', lw=2)

plt.xlabel("Actual Salary")

plt.ylabel("Predicted Salary")

plt.title("Actual vs Predicted Salary")

plt.show()



plt.figure(figsize=(10, 6))

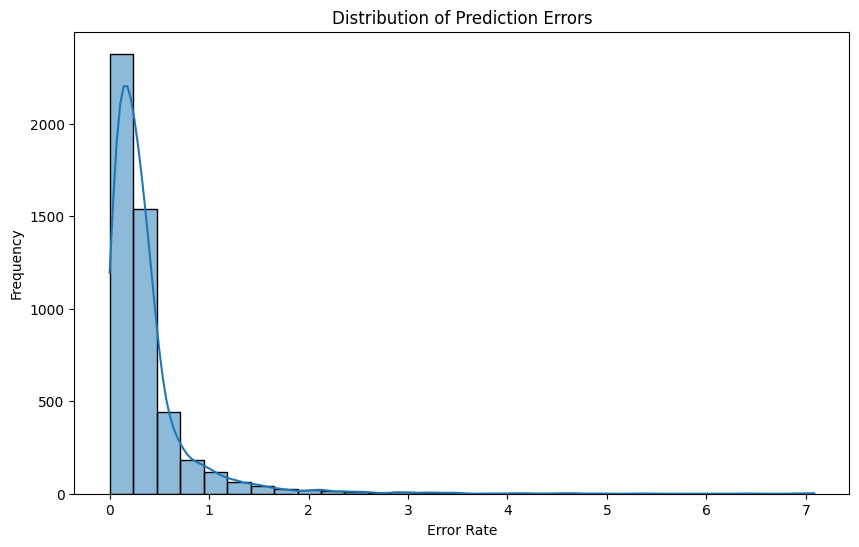
sns.histplot(error\_rate, bins=30, kde=True)

plt.xlabel("Error Rate")

plt.ylabel("Frequency")

plt.title("Distribution of Prediction Errors")

plt.show()



plt.figure(figsize=(10, 6))

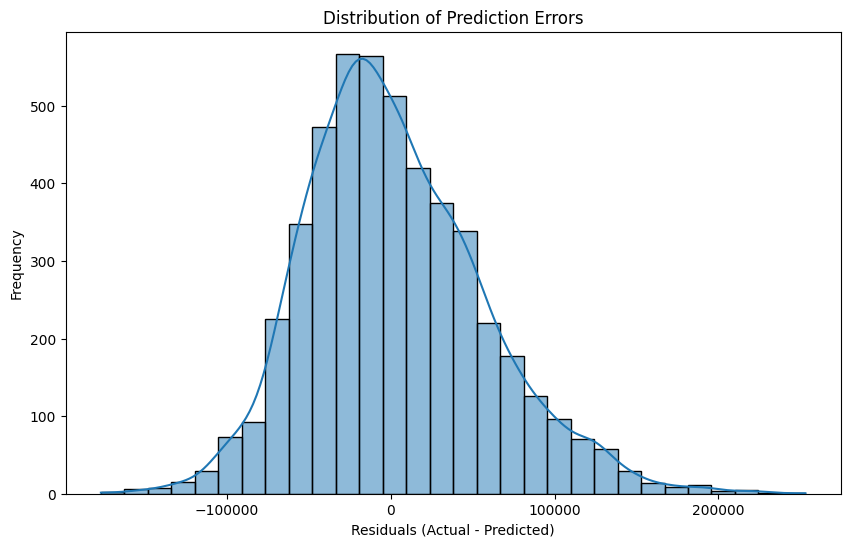
sns.histplot(residuals, bins=30, kde=True)

plt.xlabel("Residuals (Actual - Predicted)")

plt.ylabel("Frequency")

plt.title("Distribution of Prediction Errors")

plt.show()



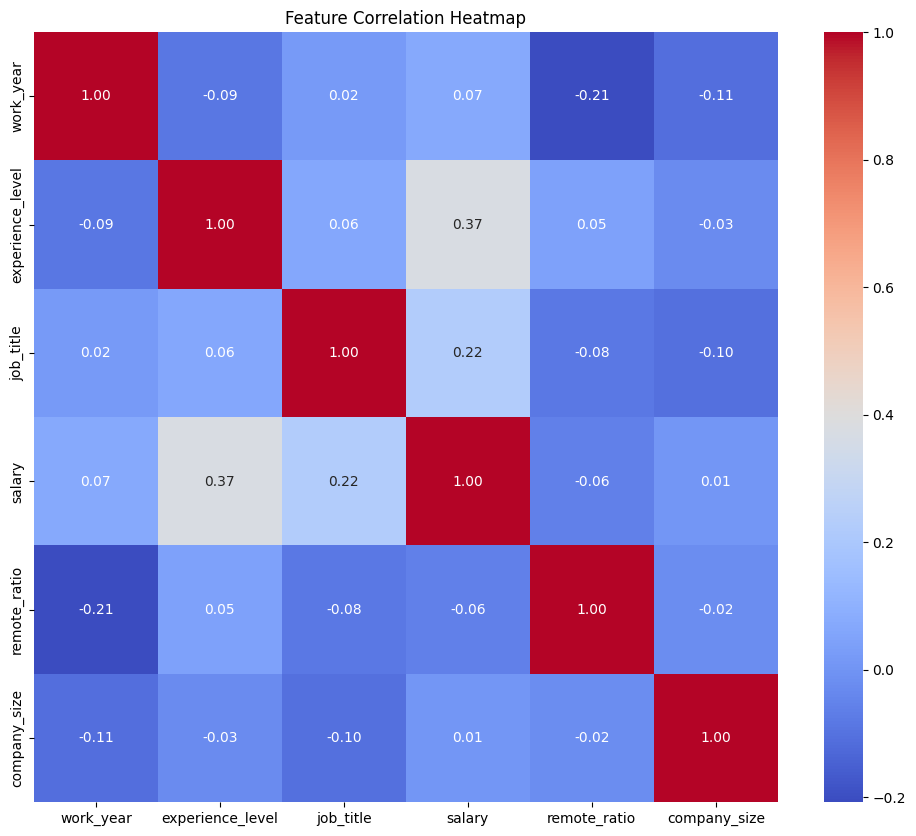
plt.figure(figsize=(12, 10))

correlation\_matrix = data.corr()

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt=".2f")

plt.title("Feature Correlation Heatmap")

plt.show()



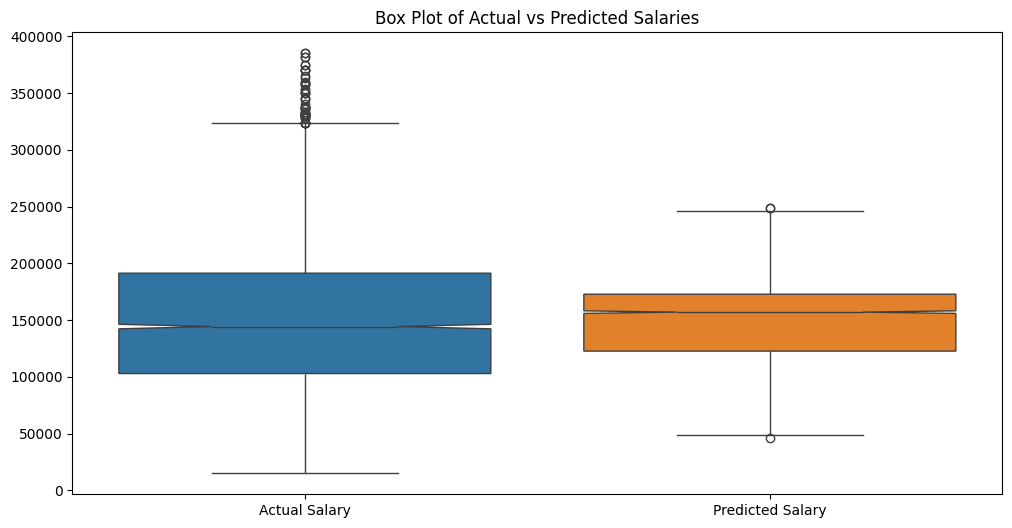
plt.figure(figsize=(12, 6))

sns.boxplot(data=[y\_test, y\_test\_pred], notch=True)

plt.xticks([0, 1], ['Actual Salary', 'Predicted Salary'])

plt.title("Box Plot of Actual vs Predicted Salaries")

plt.show()



top\_10\_jobs = data2['job\_title'].value\_counts().nlargest(10).index

top\_10\_jobs\_array = top\_10\_jobs.values

top\_10\_jobs2 = data['job\_title'].value\_counts().nlargest(10).index

filtered\_data = X\_test[X\_test['job\_title'].isin(top\_10\_jobs2)]

filtered\_data['Error Rate'] = abs(filtered\_data['salary'] - filtered\_data['prediction']) / filtered\_data['salary']

plt.figure(figsize=(12, 6))

ax = sns.boxplot(x=filtered\_data[feature\_to\_plot], y=filtered\_data['Error Rate'])

ax.set\_xticklabels(top\_10\_jobs\_array, rotation=90)

plt.xlabel("Job Title")

plt.ylabel("Error Rate")

plt.title(f"Error Rate by {feature\_to\_plot} (Top 10 Most Common)")

plt.show()

