```
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.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_absolute_error

# Load the dataset
file_path = "/content/companies_data.csv"  # Ensure the dataset is in the same directory
df = pd.read_csv(file_path)

# Drop unnecessary columns
df = df.drop(columns=["Unnamed: 0", "reviews"], errors='ignore')

# Handle missing values in 'employee' (fill with 'Unknown')
df["employee"] = df["employee"].fillna("Unknown")

# Encode categorical columns using Label Encoding
categorical_cols = ["name", "type", "hq", "employee"]
label_encoders = {}

for col in categorical_cols:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
    label_encoders[col] = le

# Define features and target
X = df.drop(columns=["rating"])  # Features
y = df["rating"]  # Target variable

# Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predict on test set
y_pred = rf_model.predict(X_test)

# Evaluate performance
mae = mean_absolute_error(y_test, y_pred)
print(f"Mean Absolute Error: {mae:.4f}")
```

```
Mean Absolute Error: 0.0000
```

```
# --- 1. Random Individual Predictions ---
random_samples = X_test.sample(3, random_state=42)
random_predictions = rf_model.predict(random_samples)

# Create DataFrame for display
random_results = random_samples.copy()
random_results["Actual Rating"] = y_test.loc[random_samples.index].values
random_results["Predicted Rating"] = random_predictions

# Decode categorical values
for col in categorical_cols:
    random_results[col] = label_encoders[col].inverse_transform(random_results[col])

# Display Random Predictions
print("Random Predictions for 3 Companies:")
print(random_results)
```

```
Random Predictions for 3 Companies:
            name     type                 hq          employee  \
12547  Capgemini  Private     Paris + 44 more  10000+ employees
69941        IBM  Private  New York + 72 more  10000+ employees
39783  Cognizant  Private   Teaneck + 46 more  10000+ employees

       Actual Rating  Predicted Rating
12547            3.3               3.3
69941            4.0               4.0
39783            3.9               3.9
```

```
# --- 2. Highest-Rated Companies ---
highest_rated = df.nlargest(3, "rating")

print("Top 3 Highest-Rated Companies:")
print(highest_rated[["name", "rating"]])
```

```
Top 3 Highest-Rated Companies:
    name  rating
27    16     4.7
57    16     4.7
87    16     4.7
```

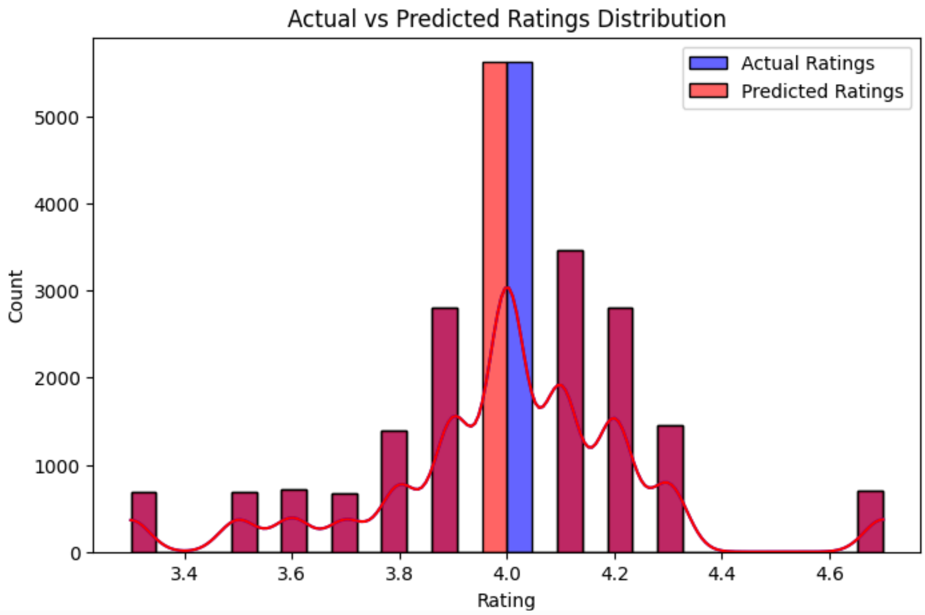
```
# --- 3. Lowest-Rated Companies ---
lowest_rated = df.nsmallest(3, "rating")

print("Top 3 Lowest-Rated Companies:")
print(lowest_rated[["name", "rating"]])
```

```
Top 3 Lowest-Rated Companies:
    name  rating
7      5     3.3
37     5     3.3
67     5     3.3
```

```
# --- 4. Predictions for Public Companies ---
public_companies = X_test[df["type"] == label_encoders["type"].transform(["Public"])[0]].sample(3, random_state=42)
public_predictions = rf_model.predict(public_companies)

public_results = public_companies.copy()
public_results["Actual Rating"] = y_test.loc[public_companies.index].values
public_results["Predicted Rating"] = public_predictions

# Decode categorical values
for col in categorical_cols:
    public_results[col] = label_encoders[col].inverse_transform(public_results[col])

print("Predictions for Public Companies:")
print(public_results)
```

```
Predictions for Public Companies:
                   name    type                      hq          employee  \
49032      Reliance jio  Public  Navi Mumbai + 114 more  10000+ employees
26112      Reliance jio  Public  Navi Mumbai + 114 more  10000+ employees
79419  HCL Technologies  Public        Noida + 107 more  10000+ employees

       Actual Rating  Predicted Rating
49032            4.0               4.0
26112            4.0               4.0
79419            3.7               3.7
<ipython-input-5-299c3d349b5e>:2: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
  public_companies = X_test[df["type"] == label_encoders["type"].transform(["Public"])[0]].sample(3, random_state=42)
```

```
# --- Histogram Plot (Comparison: Actual vs Predicted Ratings) ---
plt.figure(figsize=(8, 5))
sns.histplot(y_test, label="Actual Ratings", kde=True, color="blue", alpha=0.6, bins=30)
sns.histplot(y_pred, label="Predicted Ratings", kde=True, color="red", alpha=0.6, bins=30)
plt.xlabel("Rating")
plt.ylabel("Count")
plt.title("Actual vs Predicted Ratings Distribution")
plt.legend()
plt.show()
```


Actual vs Predicted Ratings Distribution
Actual Ratings
Predicted Ratings
Count
0
1000
2000
3000
4000
5000
3.4
3.6
3.8
4.0
4.2
4.4
4.6
Rating