Employee Salary Prediction

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
import joblib
```

```
df=pd.read_csv("/content/Salary Data.csv")
```

```
df
```

	Age	Gender	Education Level	Job Title	Years of Experience	Salary
0	32.0	Male	Bachelor's	Software Engineer	5.0	90000.0
1	28.0	Female	Master's	Data Analyst	3.0	65000.0
2	45.0	Male	PhD	Senior Manager	15.0	150000.0
3	36.0	Female	Bachelor's	Sales Associate	7.0	60000.0
4	52.0	Male	Master's	Director	20.0	200000.0
...	...	...	...	...	...	...
370	35.0	Female	Bachelor's	Senior Marketing Analyst	8.0	85000.0
371	43.0	Male	Master's	Director of Operations	19.0	170000.0
372	29.0	Female	Bachelor's	Junior Project Manager	2.0	40000.0
373	34.0	Male	Bachelor's	Senior Operations Coordinator	7.0	90000.0
374	44.0	Female	PhD	Senior Business Analyst	15.0	150000.0

375 rows × 6 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.head()
```

	Age	Gender	Education Level	Job Title	Years of Experience	Salary
0	32.0	Male	Bachelor's	Software Engineer	5.0	90000.0
1	28.0	Female	Master's	Data Analyst	3.0	65000.0
2	45.0	Male	PhD	Senior Manager	15.0	150000.0
3	36.0	Female	Bachelor's	Sales Associate	7.0	60000.0
4	52.0	Male	Master's	Director	20.0	200000.0

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df = df[['Age', 'Gender', 'Education Level', 'Job Title', 'Years of Experience', 'Salary']]
```

```
df.columns = ['Age', 'Gender', 'Degree', 'Job_Title', 'Experience_Years', 'Salary']
```

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

```
    Age  Gender      Degree          Job_Title  Experience_Years    Salary
0  32.0    Male  Bachelor's  Software Engineer               5.0   90000.0
1  28.0  Female    Master's       Data Analyst               3.0   65000.0
2  45.0    Male         PhD     Senior Manager              15.0  150000.0
3  36.0  Female  Bachelor's    Sales Associate               7.0   60000.0
4  52.0    Male    Master's           Director              20.0  200000.0
```

```
print("Initial Shape:", df.shape)
```

```
Initial Shape: (375, 6)
```

```
print("Missing Values:\n", df.isnull().sum())
```

```
Missing Values:
 Age                 2
Gender               2
Degree               2
Job_Title            2
Experience_Years     2
Salary               2
dtype: int64
```

```
print("Duplicates:", df.duplicated().sum())
```

```
Duplicates: 50
```

```
print("Data Types:\n", df.dtypes)
print(df.describe(include='all'))
```

```
Data Types:
 Age                 float64
Gender               object
Degree               object
Job_Title            object
Experience_Years    float64
Salary              float64
dtype: object
                Age Gender      Degree              Job_Title  \
count   373.000000    373         373                    373
unique         NaN      2           3                    174
top            NaN   Male  Bachelor's  Director of Marketing
freq           NaN    194         224                     12
mean     37.431635    NaN         NaN                    NaN
std       7.069073    NaN         NaN                    NaN
min      23.000000    NaN         NaN                    NaN
25%      31.000000    NaN         NaN                    NaN
50%      36.000000    NaN         NaN                    NaN
75%      44.000000    NaN         NaN                    NaN
max      53.000000    NaN         NaN                    NaN

        Experience_Years         Salary
count         373.000000     373.000000
unique               NaN            NaN
top                  NaN            NaN
freq                 NaN            NaN
mean           10.030831  100577.345845
std             6.557007   48240.013482
min             0.000000     350.000000
25%             4.000000   55000.000000
50%             9.000000   95000.000000
75%            15.000000  140000.000000
max            25.000000  250000.000000
```

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

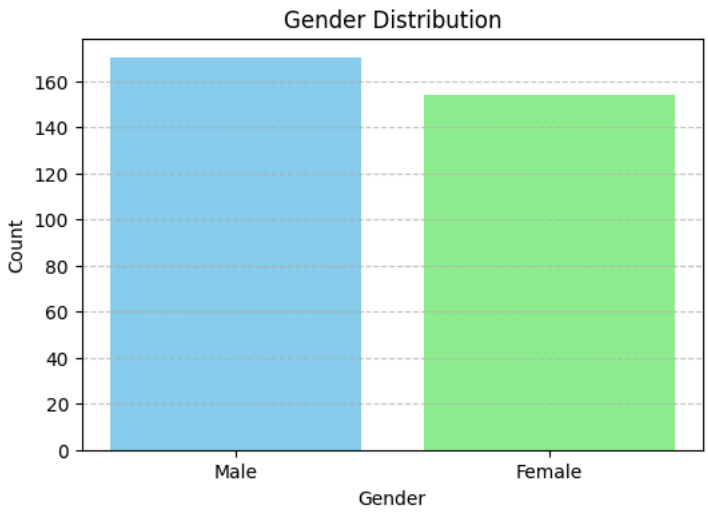
```
df.shape
df.isnull().sum()
```

	0
Age	0
Gender	0
Degree	0
Job_Title	0
Experience_Years	0
Salary	0

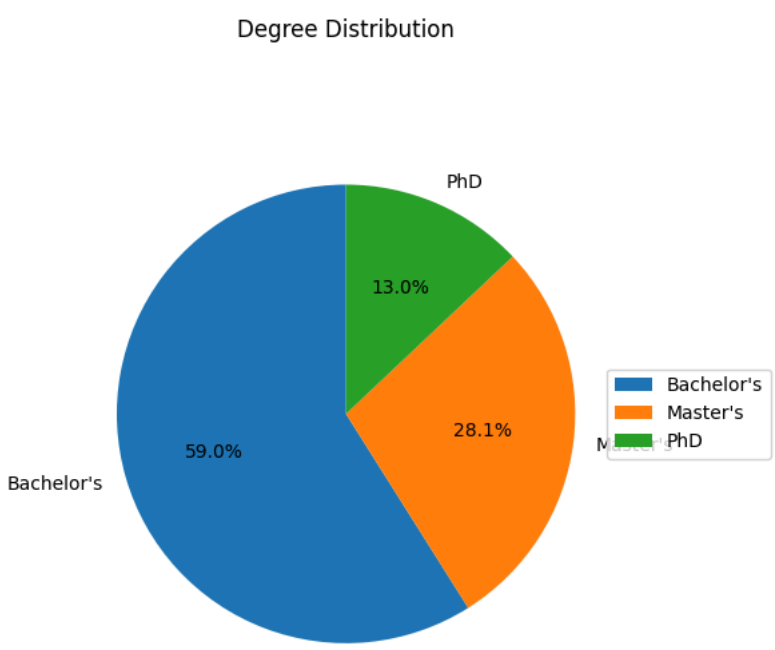
dtype: int64

```
gender_counts = df['Gender'].value_counts()
plt.figure(figsize=(6, 4))
plt.bar(gender_counts.index, gender_counts.values, color=['skyblue', 'lightgreen'])
plt.xlabel('Gender')
plt.ylabel('Count')
plt.title('Gender Distribution')
```

```
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

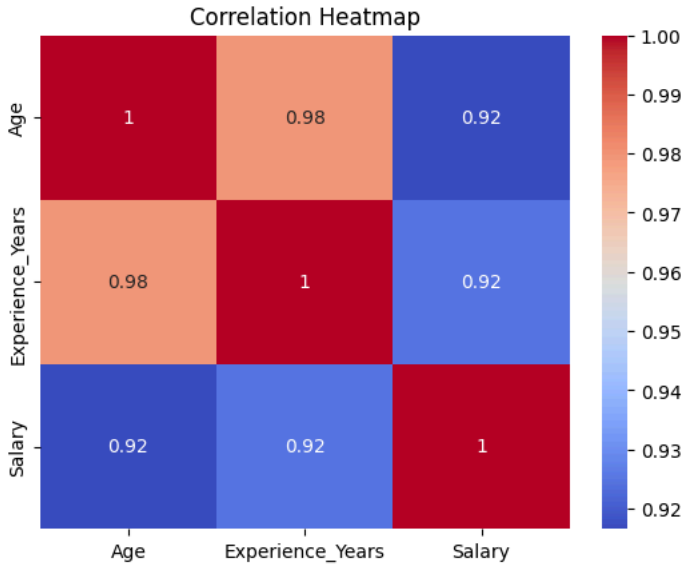



```
degree_col = 'Degree' if 'Degree' in df.columns else 'Education Level'
degree_counts = df[degree_col].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(degree_counts, labels=degree_counts.index, autopct='%1.1f%%', startangle=90)
plt.title('Degree Distribution')
plt.legend(degree_counts.index, loc='center left', bbox_to_anchor=(1, 0.5))
plt.axis('equal')
plt.tight_layout()
plt.show()
```




```
corr = df[['Age', 'Experience_Years', 'Salary']].corr()
print("\nCorrelation Matrix:\n", corr)
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```

```
Correlation Matrix:
                      Age  Experience_Years    Salary
Age              1.000000          0.979192  0.916543
Experience_Years  0.979192          1.000000  0.924455
Salary            0.916543          0.924455  1.000000
```




```
df['Degree'].value_counts()
```

Degree	count
Bachelor's	191
Master's	91
PhD	42

dtype: int64

```
df['Job_Title'].value_counts()
```

Job_Title	count
Director of Operations	9
Director of Marketing	8
Senior Marketing Manager	8
Senior Project Manager	7
Senior Data Scientist	6
...	...
Junior Social Media Specialist	1
Junior Operations Coordinator	1
Senior HR Specialist	1
Director of HR	1
Junior Financial Advisor	1

174 rows × 1 columns

dtype: int64

```
df['Gender'].value_counts()
```

Gender	count
Male	170
Female	154

dtype: int64

```
df['Experience_Years'].value_counts()
```

Experience_Years	count
3.0	27
2.0	26
9.0	19
8.0	17
10.0	16
4.0	16
7.0	16
5.0	16
16.0	15
12.0	14
19.0	13
20.0	13
15.0	12
21.0	11
18.0	11
6.0	11
14.0	11
1.5	11
13.0	10
11.0	9
22.0	8
1.0	7
17.0	5
0.0	3
25.0	3
23.0	2
24.0	1
0.5	1

dtype: int64

```
df['Job_Title'].unique()
```

```
array(['Software Engineer', 'Data Analyst', 'Senior Manager',
       'Sales Associate', 'Director', 'Marketing Analyst',
       'Product Manager', 'Sales Manager', 'Marketing Coordinator',
       'Senior Scientist', 'Software Developer', 'HR Manager',
       'Financial Analyst', 'Project Manager', 'Customer Service Rep',
       'Operations Manager', 'Marketing Manager', 'Senior Engineer',
       'Data Entry Clerk', 'Sales Director', 'Business Analyst',
       'VP of Operations', 'IT Support', 'Recruiter', 'Financial Manager',
       'Social Media Specialist', 'Software Manager', 'Junior Developer',
       'Senior Consultant', 'Product Designer', 'CEO', 'Accountant',
       'Data Scientist', 'Marketing Specialist', 'Technical Writer',
       'HR Generalist', 'Project Engineer', 'Customer Success Rep',
       'Sales Executive', 'UX Designer', 'Operations Director',
       'Network Engineer', 'Administrative Assistant',
       'Strategy Consultant', 'Copywriter', 'Account Manager',
       'Director of Marketing', 'Help Desk Analyst',
```

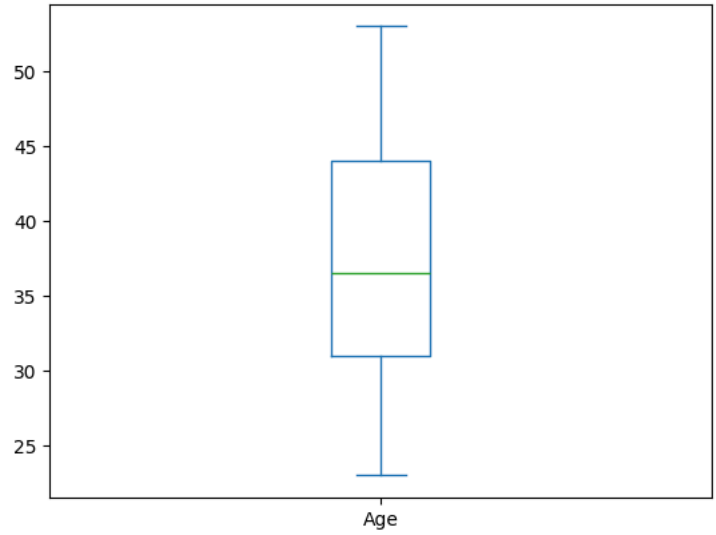
```
'Customer Service Manager', 'Business Intelligence Analyst',
'Event Coordinator', 'VP of Finance', 'Graphic Designer',
'UX Researcher', 'Social Media Manager', 'Director of Operations',
'Senior Data Scientist', 'Junior Accountant',
'Digital Marketing Manager', 'IT Manager',
'Customer Service Representative', 'Business Development Manager',
'Senior Financial Analyst', 'Web Developer', 'Research Director',
'Technical Support Specialist', 'Creative Director',
'Senior Software Engineer', 'Human Resources Director',
'Content Marketing Manager', 'Technical Recruiter',
'Sales Representative', 'Chief Technology Officer',
'Junior Designer', 'Financial Advisor', 'Junior Account Manager',
'Senior Project Manager', 'Principal Scientist',
'Supply Chain Manager', 'Senior Marketing Manager',
'Training Specialist', 'Research Scientist',
'Junior Software Developer', 'Public Relations Manager',
'Operations Analyst', 'Product Marketing Manager',
'Senior HR Manager', 'Junior Web Developer',
'Senior Project Coordinator', 'Chief Data Officer',
'Digital Content Producer', 'IT Support Specialist',
'Senior Marketing Analyst', 'Customer Success Manager',
'Senior Graphic Designer', 'Software Project Manager',
'Supply Chain Analyst', 'Senior Business Analyst',
'Junior Marketing Analyst', 'Office Manager', 'Principal Engineer',
'Junior HR Generalist', 'Senior Product Manager',
'Junior Operations Analyst', 'Senior HR Generalist',
'Sales Operations Manager', 'Senior Software Developer',
'Junior Web Designer', 'Senior Training Specialist',
'Senior Research Scientist', 'Junior Sales Representative',
'Junior Marketing Manager', 'Junior Data Analyst',
'Senior Product Marketing Manager', 'Junior Business Analyst',
'Senior Sales Manager', 'Junior Marketing Specialist',
'Junior Project Manager', 'Senior Accountant', 'Director of Sales',
'Junior Recruiter', 'Senior Business Development Manager',
'Senior Product Designer', 'Junior Customer Support Specialist',
'Senior IT Support Specialist', 'Junior Financial Analyst',
'Senior Operations Manager', 'Director of Human Resources',
'Junior Software Engineer', 'Senior Sales Representative',
'Director of Product Management', 'Junior Copywriter',
'Senior Marketing Coordinator', 'Senior Human Resources Manager',
'Junior Business Development Associate', 'Senior Account Manager',
'Senior Researcher', 'Junior HR Coordinator'
```

```
df['Job_Title'].count()
```

```
np.int64(324)
```

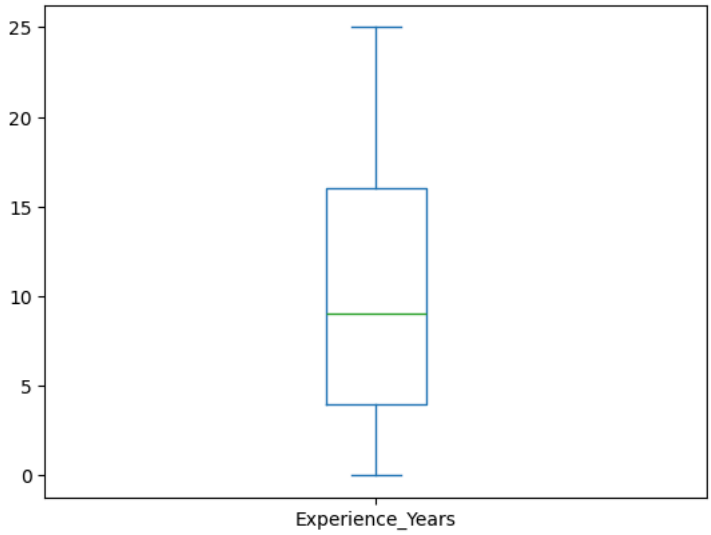
```
df.Age.plot(kind='box')
```

```
<Axes: >
```



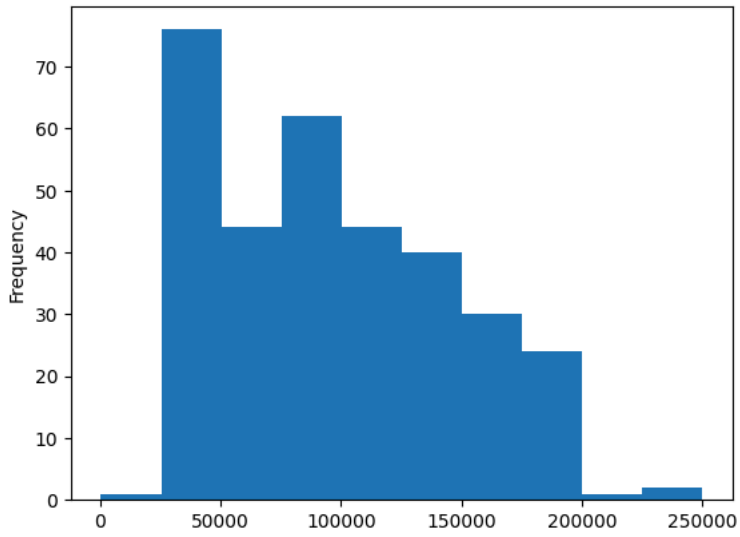

```
df.Experience_Years.plot(kind='box')
```

```
<Axes: >
```




```
df.Salary.plot(kind='hist')
```

```
<Axes: ylabel='Frequency'>
```



Label Encoding

```
le = LabelEncoder()
le_gender = LabelEncoder()
le_degree = LabelEncoder()
le_job = LabelEncoder()
```

```
df['Gender_Encode'] = le_gender.fit_transform(df['Gender'])
```

```
df['Degree_Encode'] = le_degree.fit_transform(df['Degree'])
```

```
df['Job_Title_Encode'] = le_job.fit_transform(df['Job_Title'])
```

```
df.head()
```

	Age	Gender	Degree	Job_Title	Experience_Years	Salary	Gender_Encode	Degree_Encode	Job_Title_Encode
0	32.0	Male	Bachelor's	Software Engineer	5.0	90000.0	1	0	159
1	28.0	Female	Master's	Data Analyst	3.0	65000.0	0	1	17
2	45.0	Male	PhD	Senior Manager	15.0	150000.0	1	2	130
3	36.0	Female	Bachelor's	Sales Associate	7.0	60000.0	0	0	101
4	52.0	Male	Master's	Director	20.0	200000.0	1	1	22

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
scaler = StandardScaler()
```

```
scaler = StandardScaler()
df[['Age', 'Experience_Years']] = scaler.fit_transform(df[['Age', 'Experience_Years']])
```

```
df.head()
```

	Age	Gender	Degree	Job_Title	Experience_Years	Salary	Gender_Encode	Degree_Encode	Job_Title_Encode
0	-0.750231	Male	Bachelor's	Software Engineer	-0.761821	90000.0	1	0	159
1	-1.307742	Female	Master's	Data Analyst	-1.063017	65000.0	0	1	17
2	1.061680	Male	PhD	Senior Manager	0.744158	150000.0	1	2	130
3	-0.192720	Female	Bachelor's	Sales Associate	-0.460625	60000.0	0	0	101
4	2.037324	Male	Master's	Director	1.497148	200000.0	1	1	22

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
X = df[['Age', 'Gender_Encode', 'Degree_Encode', 'Job_Title_Encode', 'Experience_Years']]
y = df['Salary']
```

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
x_train.head()
```

	Age	Gender_Encode	Degree_Encode	Job_Title_Encode	Experience_Years
73	-1.307742	1	0	166	-1.213615
182	0.922302	0	2	155	0.744158
17	0.225413	1	2	116	0.292364
24	0.504169	1	1	37	0.442962
146	0.643547	0	2	115	0.894756

Next steps: Generate code with x_train | View recommended plots | New interactive sheet

```
x_train.shape
```

```
(259, 5)
```

```
models = {
    'Linear Regression': LinearRegression(),
    'Decision Tree': DecisionTreeRegressor(random_state=42),
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Random Forest': RandomForestRegressor(n_estimators=10, max_depth=2, random_state=42)
}
```

```
results = []
```

```
best_r2 = -np.inf
best_model = None
best_model_name = ""
best_eval_df = None
```

```
for name, model in models.items():
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)

    r2 = r2_score(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
```

```
    eval_df = pd.DataFrame({
        'y_Actual': y_test,
        'y_Predicted': y_pred,
    })
    eval_df['Error'] = eval_df['y_Actual'] - eval_df['y_Predicted']
    eval_df['abs_error'] = abs(eval_df['Error'])

    print(f"\n📘 {name} Detailed Evaluation:")
    print(eval_df.head())
    print(f"Mean Absolute Error: {eval_df['abs_error'].mean():.2f}")

    results.append({
        'Model': name,
        'R2 Score': round(r2, 4),
        'MAE': round(mae, 2),
        'MSE': round(mse, 2),
        'RMSE': round(rmse, 2)
    })

    if r2 > best_r2:
        best_r2 = r2
        best_model = model
        best_model_name = name
        best_eval_df = eval_df

metrics_df = pd.DataFrame(results)
```

```
📘 Linear Regression Detailed Evaluation:
     y_Actual    y_Predicted         Error     abs_error
132  100000.0  117415.913446 -17415.913446  17415.913446
108  100000.0  125562.807428 -25562.807428  25562.807428
137   50000.0   48965.153862   1034.846138   1034.846138
9    110000.0  128739.348880 -18739.348880  18739.348880
181  105000.0  106828.499305  -1828.499305   1828.499305
Mean Absolute Error: 10570.79

📘 Decision Tree Detailed Evaluation:
     y_Actual  y_Predicted    Error  abs_error
132  100000.0     100000.0      0.0        0.0
108  100000.0     110000.0 -10000.0    10000.0
137   50000.0      55000.0  -5000.0     5000.0
9    110000.0     120000.0 -10000.0    10000.0
181  105000.0     100000.0   5000.0     5000.0
Mean Absolute Error: 12692.31

📘 Logistic Regression Detailed Evaluation:
     y_Actual  y_Predicted    Error  abs_error
132  100000.0     120000.0 -20000.0    20000.0
108  100000.0     120000.0 -20000.0    20000.0
137   50000.0      45000.0   5000.0     5000.0
9    110000.0     150000.0 -40000.0    40000.0
181  105000.0      90000.0  15000.0    15000.0
Mean Absolute Error: 14307.69

📘 Random Forest Detailed Evaluation:
     y_Actual    y_Predicted         Error     abs_error
132  100000.0   92778.401905   7221.598095   7221.598095
108  100000.0  128341.366469 -28341.366469  28341.366469
137   50000.0   49071.388761    928.611239    928.611239
9    110000.0   92778.401905  17221.598095  17221.598095
181  105000.0   92778.401905  12221.598095  12221.598095
Mean Absolute Error: 13623.34
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (status=
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
print("\n📊 All Model Evaluation Metrics:")
print(metrics_df.to_string(index=False))
```

```
📊 All Model Evaluation Metrics:
              Model  R2 Score      MAE          MSE     RMSE
  Linear Regression    0.8911 10570.79 205754135.72 14344.13
      Decision Tree    0.8325 12692.31 316538461.54 17791.53
Logistic Regression    0.7843 14307.69 407692307.69 20191.39
      Random Forest    0.8402 13623.34 301977413.05 17377.50
```

```
print(f"\n Best Model: {best_model_name} with R2 Score: {best_r2:.4f}")
```

```
 Best Model: Linear Regression with R2 Score: 0.8911
```

```
joblib.dump(best_model, "best_salary_model.pkl")
print(" Best model saved as: best_salary_model.pkl")
```

```
 Best model saved as: best_salary_model.pkl
```

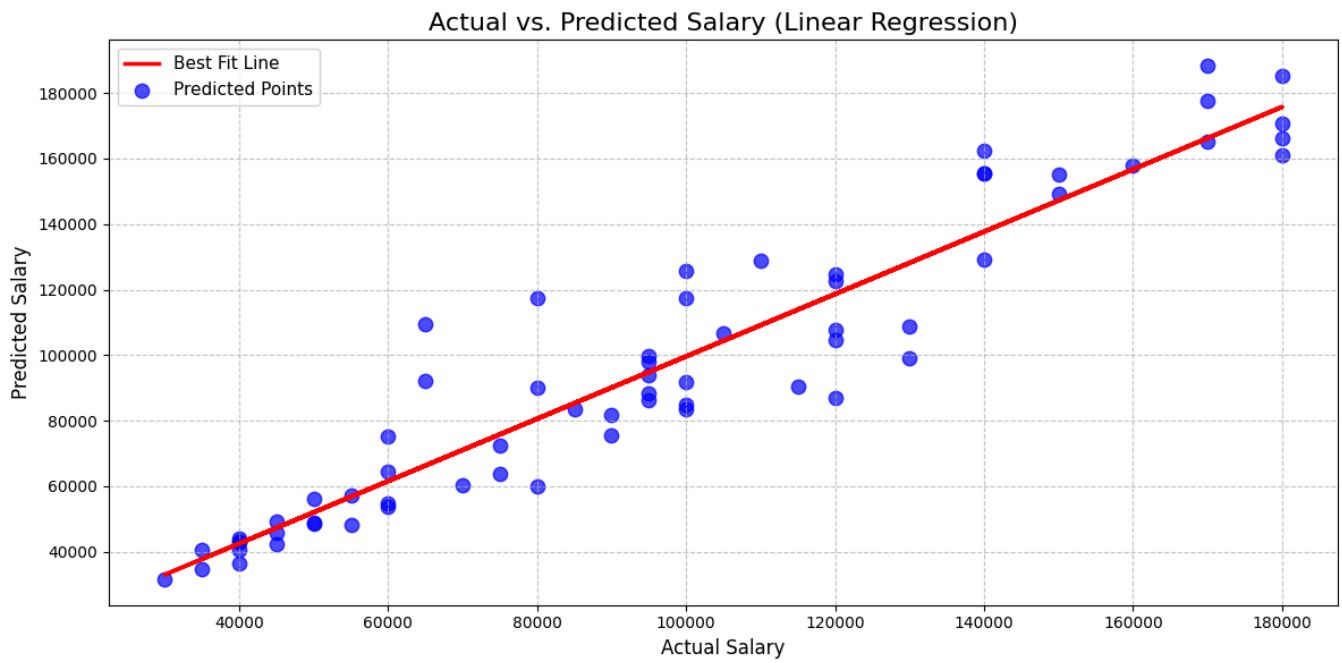
Start coding or generate with AI.

```
y_actual = best_eval_df['y_Actual'].values.reshape(-1, 1)
y_predicted = best_eval_df['y_Predicted'].values.reshape(-1, 1)

reg_model = LinearRegression()
reg_model.fit(y_actual, y_predicted)
reg_line = reg_model.predict(y_actual)

plt.figure(figsize=(12, 6))
plt.plot(y_actual, reg_line, color='red', linestyle='-', linewidth=2.5, label='Best Fit Line')
plt.scatter(y_actual, y_predicted, color='blue', s=80, alpha=0.7, label='Predicted Points')

plt.title(f'Actual vs. Predicted Salary ({best_model_name})', fontsize=16)
plt.xlabel('Actual Salary', fontsize=12)
plt.ylabel('Predicted Salary', fontsize=12)
plt.legend(fontsize=11)
plt.grid(True, linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
plt.close()
```




```
def predict_salary(age, gender, degree, job_title, experience):
    try:
        gender_enc = le_gender.transform([gender])[0]
        degree_enc = le_degree.transform([degree])[0]
        job_title_enc = le_job.transform([job_title])[0]
        scaled_values = scaler.transform([[age, experience]])[0]
        age_scaled = scaled_values[0]
        experience_scaled = scaled_values[1]

        features = [[age_scaled, gender_enc, degree_enc, job_title_enc, experience_scaled]]
        prediction = best_model.predict(features)
        return prediction[0]

    except ValueError as ve:
        raise ValueError(f"Encoding error: {ve}")
```

```
    except Exception as e:
        raise Exception(f"Unexpected roor:{e}")


import warnings
warnings.filterwarnings("ignore", category=UserWarning)




print("\n--- Predict Salary ---")
age = float(input("Enter age: "))
gender = input("Enter gender (e.g., Male/Female): ")
degree = input("Enter degree (e.g., Bachelor's, Master's): ")
job_title = input("Enter job title (e.g., Software Engineer): ")
experience = float(input("Enter years of experience: "))

try:
    predicted_salary = predict_salary(age, gender, degree, job_title, experience)
    print(f"\n💰 Predicted Salary for given input: ${predicted_salary:.2f}")
except ValueError as ve:
    print(f"\n Error during prediction: {ve}")
except Exception as e:
    print(f"\n Unexpected error: {e}")


 ...
```