

```

import pandas as pd
import math as m
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from xgboost import XGBRegressor
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
import warnings
warnings.filterwarnings('ignore')

```

```
df = pd.read_csv("Salary Data.csv") # store data into df
```

```

print(df.dtypes)
print("\n", df.head())

```

```

➡ Age                float64
   Gender              object
   Education Level     object
   Job Title           object
   Years of Experience float64
   Salary              float64
   dtype: object

```

```

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

      Salary
0   90000.0
1   65000.0
2  150000.0
3   60000.0
4  200000.0

```

```

values = df['Job Title'].value_counts().nlargest(15).values.round(2)
index = df['Job Title'].value_counts().nlargest(15).index

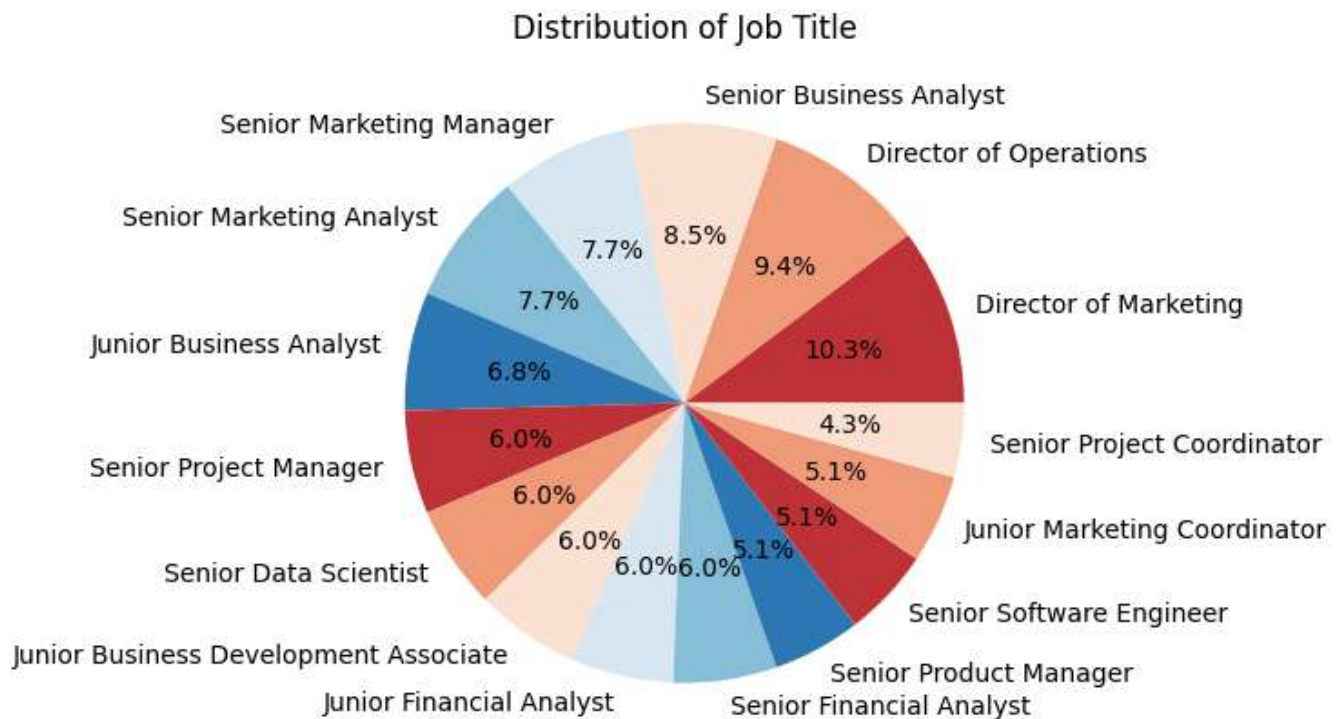
```

```

color = sns.color_palette('RdBu')
plt.figure(figsize=(7,5))

```

```
# pie chart of column
plt.title('Distribution of Job Title')
plt.pie(values, labels=index, autopct='%1.1f%%', colors=color)
plt.show()
```



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



	0
Age	2
Gender	2
Education Level	2
Job Title	2
Years of Experience	2
Salary	2

```
dtype: int64
```

```
df.duplicated().sum()
```



```
np.int64(50)
```

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

```
object_data = df.select_dtypes(include = 'object')
for column in object_data.columns:
    print('column ',df[column].mode()[0])
    df[column].fillna(df[column].mode()[0],inplace = True)
```

```
⇒ column Male
   column Bachelor's
   column Director of Operations
```

```
numeric_data = df.select_dtypes(exclude = 'object')
for column in numeric_data.columns:
    print(column,df[column].mean(),df[column].median())
    df[column] = df[column].fillna(df[column].median())
```

```
⇒ Age 37.382716049382715 36.5
   Years of Experience 10.058641975308642 9.0
   Salary 99985.64814814815 95000.0
```

```
for column in object_data.columns:
    le = LabelEncoder()
    df[column] = le.fit_transform(df[column])
    print(le.classes_)
```

```
⇒ ['Female' 'Male']
["Bachelor's" "Master's" 'PhD']
['Account Manager' 'Accountant' 'Administrative Assistant'
 'Business Analyst' 'Business Development Manager'
 'Business Intelligence Analyst' 'CEO' 'Chief Data Officer'
 'Chief Technology Officer' 'Content Marketing Manager' 'Copywriter'
 'Creative Director' 'Customer Service Manager' 'Customer Service Rep'
 'Customer Service Representative' 'Customer Success Manager'
 'Customer Success Rep' 'Data Analyst' 'Data Entry Clerk' 'Data Scientist'
 'Digital Content Producer' 'Digital Marketing Manager' 'Director'
 'Director of Business Development' 'Director of Engineering'
 'Director of Finance' 'Director of HR' 'Director of Human Capital'
 'Director of Human Resources' 'Director of Marketing'
 'Director of Operations' 'Director of Product Management'
 'Director of Sales' 'Director of Sales and Marketing' 'Event Coordinator'
 'Financial Advisor' 'Financial Analyst' 'Financial Manager'
 'Graphic Designer' 'HR Generalist' 'HR Manager' 'Help Desk Analyst'
 'Human Resources Director' 'IT Manager' 'IT Support'
 'IT Support Specialist' 'Junior Account Manager' 'Junior Accountant'
 'Junior Advertising Coordinator' 'Junior Business Analyst'
 'Junior Business Development Associate'
 'Junior Business Operations Analyst' 'Junior Copywriter'
 'Junior Customer Support Specialist' 'Junior Data Analyst'
 'Junior Data Scientist' 'Junior Designer' 'Junior Developer'
 'Junior Financial Advisor' 'Junior Financial Analyst'
 'Junior HR Coordinator' 'Junior HR Generalist' 'Junior Marketing Analyst'
 'Junior Marketing Coordinator' 'Junior Marketing Manager'
 'Junior Marketing Specialist' 'Junior Operations Analyst'
 'Junior Operations Coordinator' 'Junior Operations Manager'
 'Junior Product Manager' 'Junior Project Manager' 'Junior Recruiter']
```

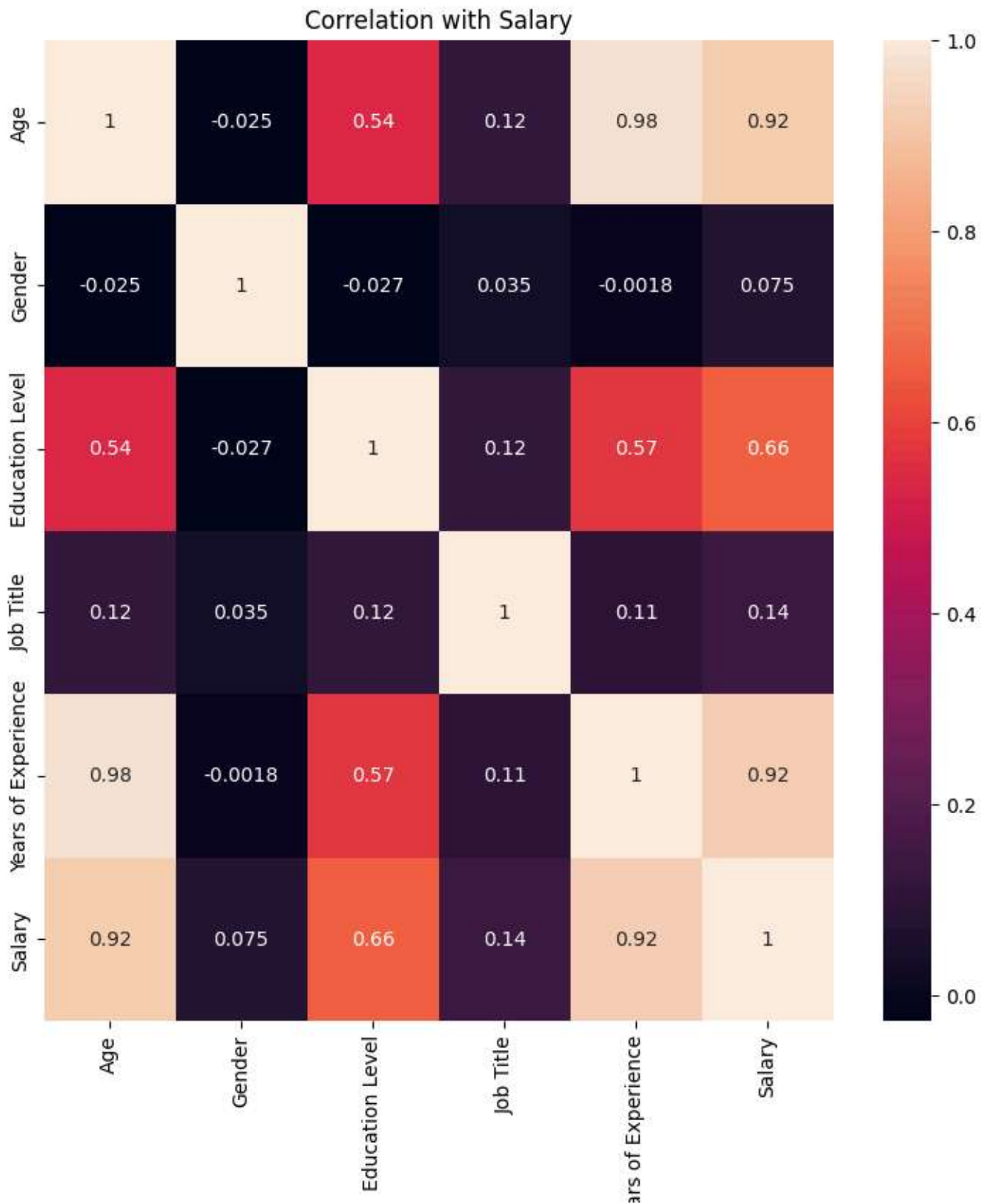
```
'Junior Research Scientist' 'Junior Sales Representative'
'Junior Social Media Manager' 'Junior Social Media Specialist'
'Junior Software Developer' 'Junior Software Engineer'
'Junior UX Designer' 'Junior Web Designer' 'Junior Web Developer'
'Marketing Analyst' 'Marketing Coordinator' 'Marketing Manager'
'Marketing Specialist' 'Network Engineer' 'Office Manager'
'Operations Analyst' 'Operations Director' 'Operations Manager'
'Principal Engineer' 'Principal Scientist' 'Product Designer'
'Product Manager' 'Product Marketing Manager' 'Project Engineer'
'Project Manager' 'Public Relations Manager' 'Recruiter'
'Research Director' 'Research Scientist' 'Sales Associate'
'Sales Director' 'Sales Executive' 'Sales Manager'
'Sales Operations Manager' 'Sales Representative'
'Senior Account Executive' 'Senior Account Manager' 'Senior Accountant'
'Senior Business Analyst' 'Senior Business Development Manager'
'Senior Consultant' 'Senior Data Analyst' 'Senior Data Engineer'
'Senior Data Scientist' 'Senior Engineer' 'Senior Financial Advisor'
'Senior Financial Analyst' 'Senior Financial Manager'
'Senior Graphic Designer' 'Senior HR Generalist' 'Senior HR Manager'
'Senior HR Specialist' 'Senior Human Resources Coordinator'
'Senior Human Resources Manager' 'Senior Human Resources Specialist'
'Senior IT Consultant' 'Senior IT Project Manager'
'Senior IT Support Specialist' 'Senior Manager'
'Senior Marketing Analyst' 'Senior Marketing Coordinator'
'Senior Marketing Director' 'Senior Marketing Manager'
'Senior Marketing Specialist' 'Senior Operations Analyst'
'Senior Operations Coordinator' 'Senior Operations Manager'
'Senior Product Designer' 'Senior Product Development Manager'
```

```
df.corr()
```

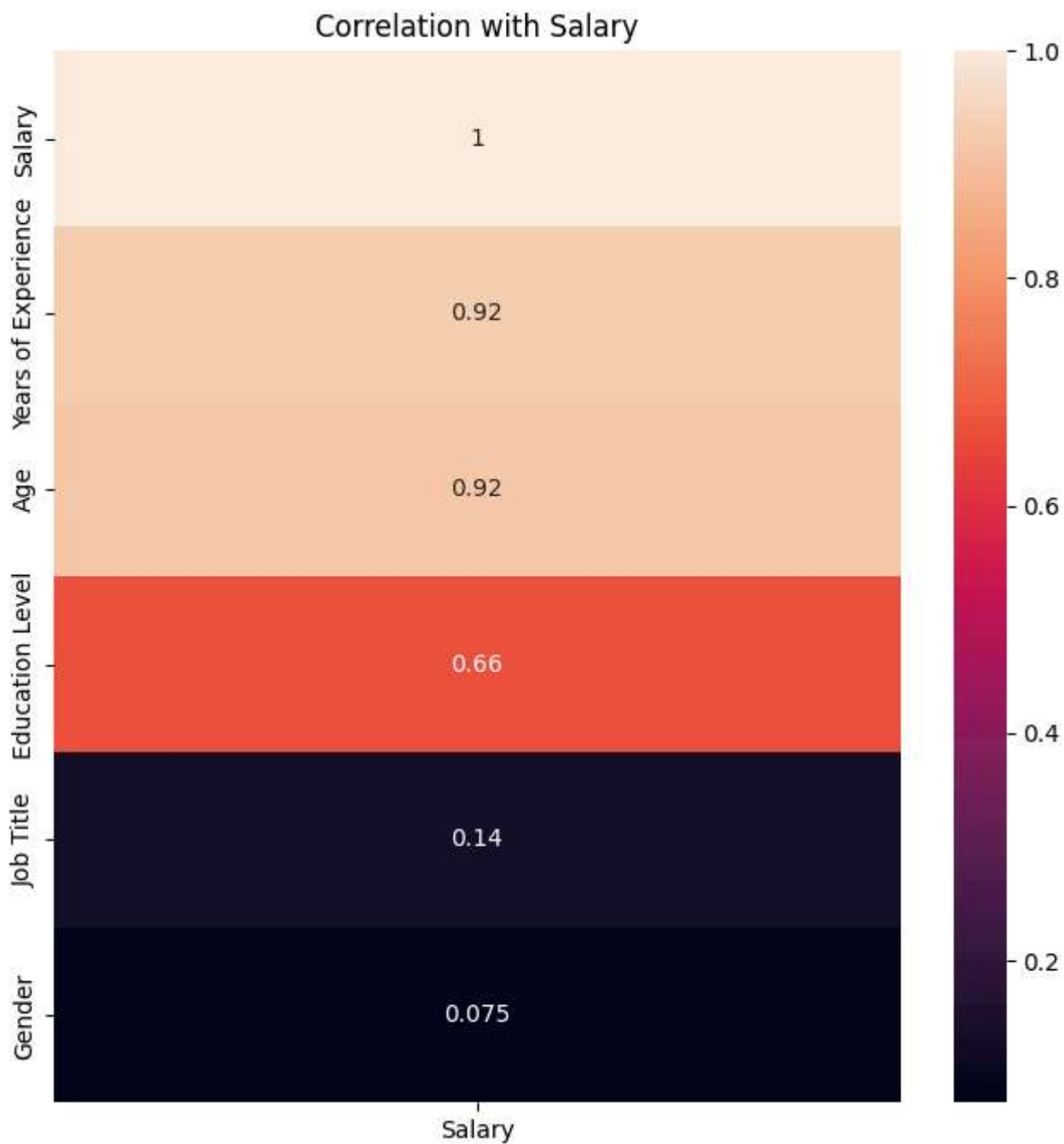


	Age	Gender	Education Level	Job Title	Years of Experience	Salary
Age	1.000000	-0.025358	0.540180	0.118804	0.979191	0.916545
Gender	-0.025358	1.000000	-0.026629	0.035267	-0.001832	0.075013
Education Level	0.540180	-0.026629	1.000000	0.116060	0.573754	0.661544
Job Title	0.118804	0.035267	0.116060	1.000000	0.105868	0.136005
Years of Experience	0.979191	-0.001832	0.573754	0.105868	1.000000	0.924454
Salary	0.916545	0.075013	0.661544	0.136005	0.924454	1.000000

```
plt.figure(figsize=(9,9))
correlation = df.corr()
sns.heatmap(correlation, annot=True)
plt.title("Correlation with Salary")
plt.show()
```



```
plt.figure(figsize=(8,8))
correlation = df.corr()
sns.heatmap(correlation[['Salary']].sort_values(by='Salary', ascending=False), annot=True)
plt.title("Correlation with Salary")
plt.show()
```



✓ Split Data

```
x = df.drop('Salary', axis = 1)
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
```



	Age	Gender	Education Level	Job Title	Years of Experience
172	36.5	1	0	30	9.0
183	27.0	1	0	69	2.0
17	39.0	1	2	116	12.0
24	41.0	1	1	37	13.0
132	40.0	0	1	154	12.0
...
188	50.0	0	2	33	22.0
71	39.0	0	0	98	11.0
106	30.0	0	0	20	3.0
283	29.0	0	0	50	1.5
102	49.0	0	1	122	19.0

260 rows × 5 columns

```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
```

ML Model : Linear Regression

```
model = LinearRegression()
```

```
model.fit(x_train,y_train)
```

```
# Test model
y_predict = model.predict(x_test)
```

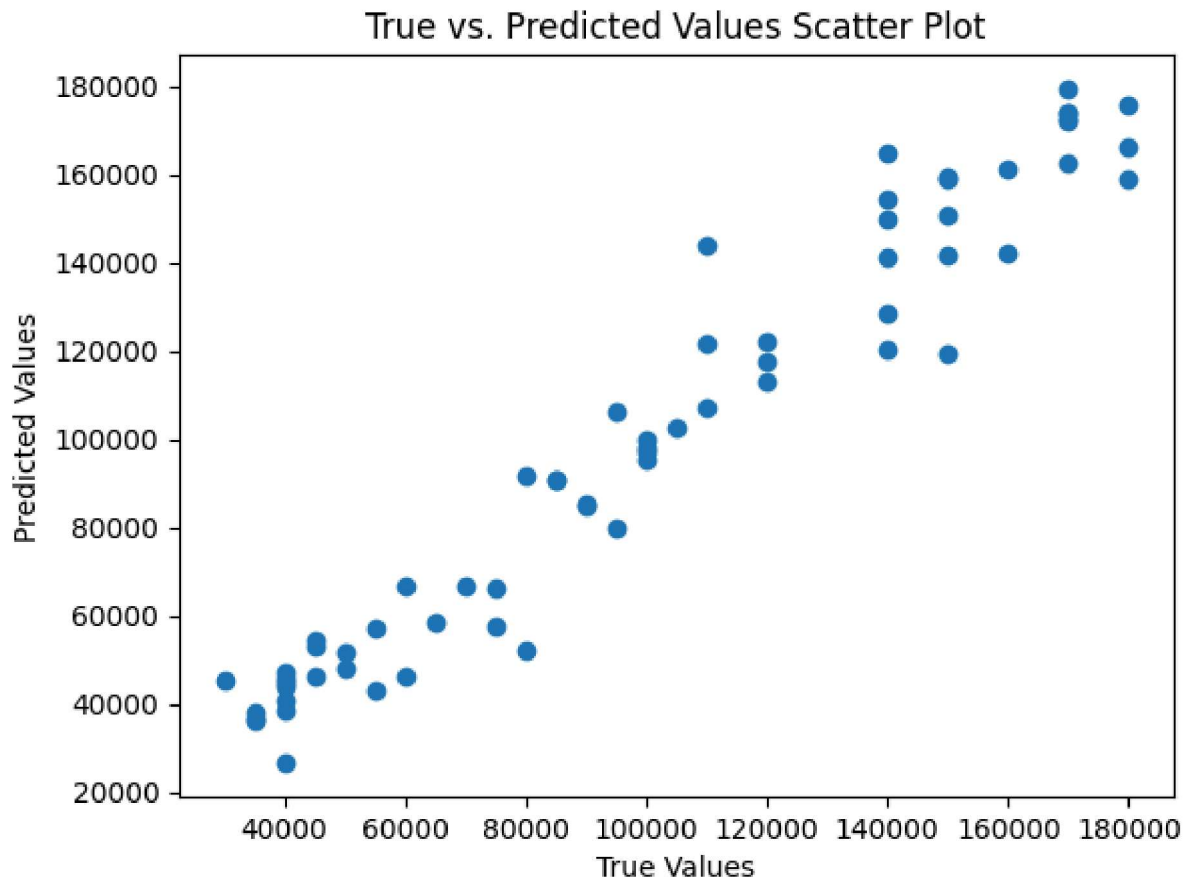
```
#Evaluate Model
print(f"R2 Score = {r2_score(y_test,y_predict) * 100:.2f}%")
print('Mean Absolute Error = ',mean_absolute_error(y_test, y_predict))
print('Mean Squared Error = ',mean_squared_error(y_test, y_predict))
print('Root Mean Squared Error = ',m.sqrt(mean_squared_error(y_test, y_predict)))
```

```
# Scatter plot for real and predict data
plt.scatter(y_test,y_predict)
plt.xlabel("True Values")
plt.ylabel("Predicted Values")
```

```
# Add a title to the plot
plt.title("True vs. Predicted Values Scatter Plot")
```

```
# Display the plot
plt.show()
```

```
➞ R2 Score = 94.58%
Mean Absolute Error = 8255.630408653846
Mean Squared Error = 123815484.41507193
Root Mean Squared Error = 11127.24064694711
```



ML Model = XGBRegressor

```
model_2 = XGBRegressor()
```

```
model_2.fit(x_train,y_train)
```

```
# Test model
y_predict = model_2.predict(x_test)
#Evaluate Model
print(f"R2 Score = {r2_score(y_test,y_predict) * 100:.2f}%")
print('Mean Absolute Error = ',mean_absolute_error(y_test, y_predict))
print('Mean Squared Error = ',mean_squared_error(y_test, y_predict))
print('Root Mean Squared Error = ',m.sqrt(mean_squared_error(y_test, y_predict)))
```

```
# Scatter plot for real and predict data
plt.scatter(y_test,y_predict)
```

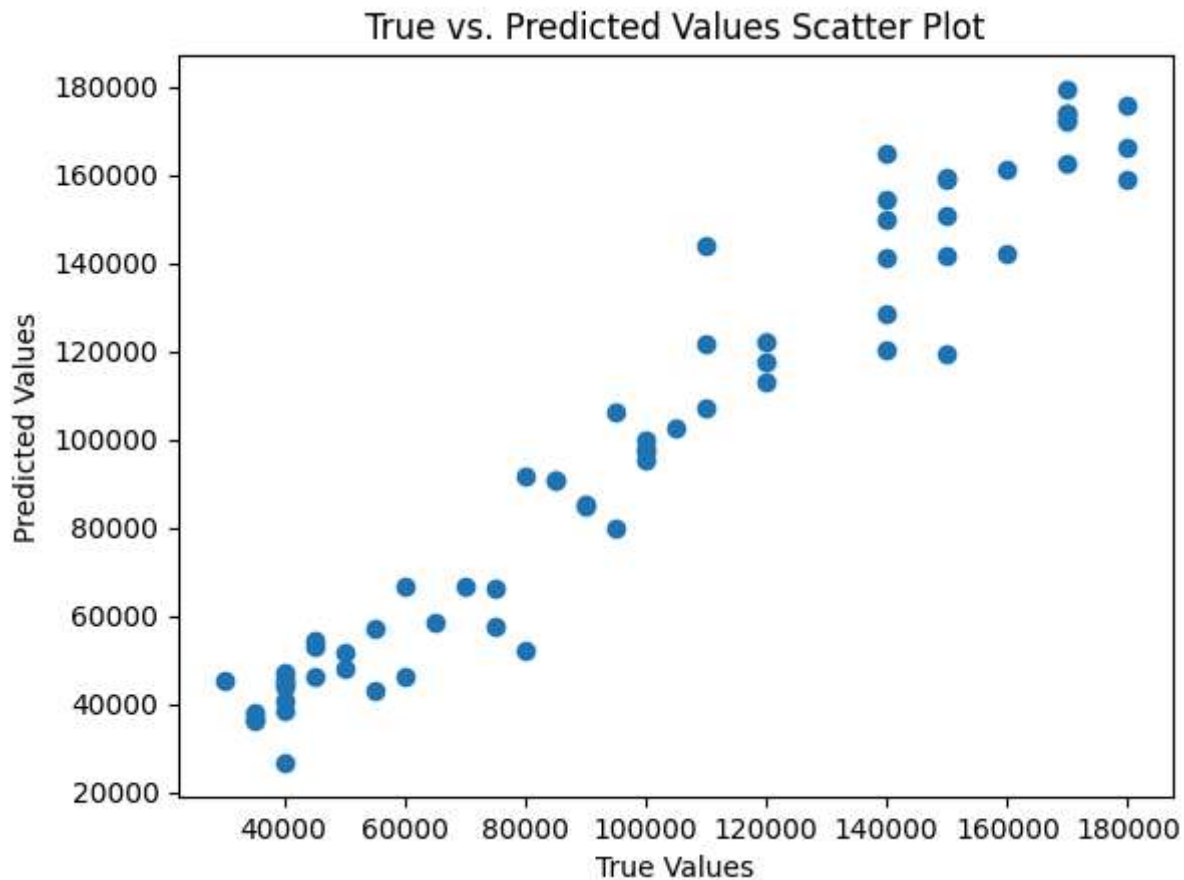


```
plt.xlabel("True Values")
plt.ylabel("Predicted Values")

# Add a title to the plot
plt.title("True vs. Predicted Values Scatter Plot")

# Display the plot
plt.show()
```

```
➞ R2 Score = 94.58%
Mean Absolute Error = 8255.630408653846
Mean Squared Error = 123815484.41507193
Root Mean Squared Error = 11127.24064694711
```



```
model_data = {
    "model": model_2,
    "label_encoders": le,
    "scaler": scaler,
    "feature_names": df.columns.tolist()
}
joblib.dump(model_data, "salary_predictor.pkl")
```

```
➞ ['salary_predictor.pkl']
```

```
model_bundle = joblib.load("salary_predictor.pkl")
print(type(model_bundle))
print(model_bundle.keys()) # if it's a dict
```

```
>>> <class 'dict'>
dict_keys(['model', 'label_encoders', 'scaler', 'feature_names'])
```

```
model_bundle = joblib.load("salary_predictor.pkl")
model = model_bundle["model"]
```

For web app I deploying my model in Streamlit as salary predictor app.

```
!pip install streamlit pyngrok --quiet
```

```
>>> _____ 44.3/44.3 kB 1.7 MB/s eta 0:00:00
_____ 9.9/9.9 MB 65.6 MB/s eta 0:00:00
_____ 6.9/6.9 MB 96.0 MB/s eta 0:00:00
_____ 79.1/79.1 kB 6.9 MB/s eta 0:00:00
```

```
app_code = '''
import streamlit as st
import numpy as np
import joblib

# Load model
model = joblib.load("salary_predictor.pkl")


st.title("Employee Salary Predictor 📁💰")
st.write("Enter employee details to predict their salary.")

# Input fields
age = st.number_input("Age", min_value=18, max_value=70, value=30)
education = st.selectbox("Education Level", ["High School", "Bachelor", "Master", "PhD"])
experience = st.slider("Years of Experience", 0, 40, 5)
position = st.selectbox("Job Position", ["Junior", "Mid-Level", "Senior", "Manager", "Execut
location = st.selectbox("Location", ["Urban", "Suburban", "Rural"])

if st.button("Predict Salary"):
    input_data = [[age, education, experience, position, location]]
    try:
        prediction = model.predict(input_data)
        st.success(f"Estimated Salary: ₹{prediction[0]:.2f}")
    except Exception as e:
        st.error(f"Prediction failed: {e}")
...
'''
```


```
# Write to file
with open('app.py', 'w') as f:
    f.write(app_code)
```

```
!ngrok config add-authtoken 30BxNhXH201e75RqkcS9dEtTrio_5sYVqeUw2jwN6eFDhbNGn
```

 Authtoken saved to configuration file: /root/.config/ngrok/ngrok.yml

```
import threading
import time
from pyngrok import ngrok

# Kill existing tunnels (if any)
ngrok.kill()

# Start the tunnel first
public_url = ngrok.connect(8501)
print(f" Your Streamlit app is available at: {public_url}")

# Run streamlit in a separate thread
def run_app():
    !streamlit run app.py

thread = threading.Thread(target=run_app)
thread.start()

# Give time to spin up
time.sleep(5)
```

  Your Streamlit app is available at: NgrokTunnel: "<https://aac86f2ba69f.ngrok-free.ap>

Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.

You can now view your Streamlit app in your browser.

Local URL: <http://localhost:8502>
Network URL: <http://172.28.0.12:8502>
External URL: <http://34.106.152.4:8502>

