Gemini

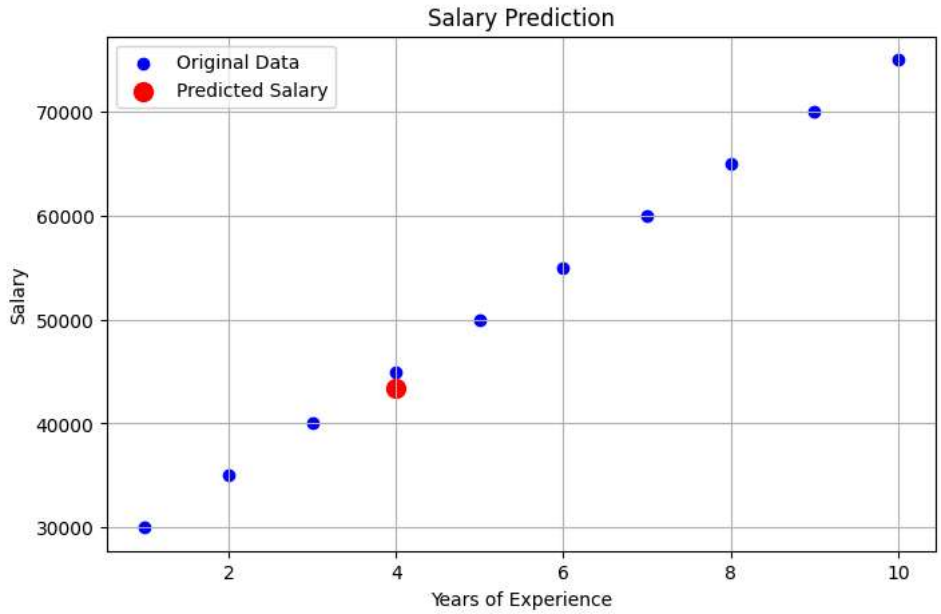
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt  # <-- NEW: For plotting
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, VotingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Sample dataset
def create_sample_data():
    data = {
        'Experience': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
        'Test_Score': [80, 82, 84, 86, 88, 90, 92, 94, 96, 98],
        'Interview_Score': [70, 72, 74, 76, 78, 80, 82, 84, 86, 88],
        'Salary': [30000, 35000, 40000, 45000, 50000, 55000, 60000, 65000, 70000, 75000]
    }
    return pd.DataFrame(data)

# Train the ensemble model
def train_model(df):
    X = df[['Experience', 'Test_Score', 'Interview_Score']]
    y = df['Salary']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    model1 = LinearRegression()
    model2 = RandomForestRegressor(random_state=1)
    model3 = GradientBoostingRegressor(random_state=1)

    ensemble_model = VotingRegressor(estimators=[
        ('lr', model1),
        ('rf', model2),
        ('gb', model3)
    ])

    ensemble_model.fit(X_train, y_train)

    y_pred = ensemble_model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    print(f" Model Trained | Mean Squared Error on Test Data: {mse:.2f}")
    return ensemble_model, df

# Predict salary based on user input and show graph
def predict_salary(model, df):
    try:
        exp = float(input("Enter candidate's years of experience: "))
        test_score = float(input("Enter candidate's test score (out of 100): "))
        interview_score = float(input("Enter candidate's interview score (out of 100): "))

        candidate_df = pd.DataFrame([[exp, test_score, interview_score]],
                                    columns=['Experience', 'Test_Score', 'Interview_Score'])

        predicted_salary = model.predict(candidate_df)
        print(f"\nPredicted Salary: ${predicted_salary[0]:,.2f}")

        # Visualize the prediction
        plt.figure(figsize=(10, 6))
        plt.scatter(df['Experience'], df['Salary'], color='blue', label='Actual Salaries')
        plt.scatter(exp, predicted_salary[0], color='red', marker='*', s=200, label='Predicted Salary')
        plt.xlabel('Experience (Years)')
        plt.ylabel('Salary ($)')
        plt.title('Salary Prediction based on Experience')
        plt.legend()
        plt.grid(True)
        plt.show()

    except ValueError:
        print("Invalid input. Please enter numeric values for experience, test score, and interview score.
    except Exception as e:
        print(f"An error occurred: {e}")

# Main execution
if __name__ == '__main__':
    sample_df = create_sample_data()
    trained_model, full_df = train_model(sample_df)
```

```
    predict_salary(trained_model, full_df)
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt  # <-- NEW: For plotting
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, VotingRegressor
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

# Sample dataset
def create_sample_data():
    data = {
        'Experience': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
        'Test_Score': [80, 82, 84, 86, 88, 90, 92, 94, 96, 98],
        'Interview_Score': [70, 72, 74, 76, 78, 80, 82, 84, 86, 88],
        'Salary': [30000, 35000, 40000, 45000, 50000, 55000, 60000, 65000, 70000, 75000]
    }
    return pd.DataFrame(data)

# Train the ensemble model
def train_model(df):
    X = df[['Experience', 'Test_Score', 'Interview_Score']]
    y = df['Salary']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    model1 = LinearRegression()
    model2 = RandomForestRegressor(random_state=1)
    model3 = GradientBoostingRegressor(random_state=1)

    ensemble_model = VotingRegressor(estimators=[
        ('lr', model1),
        ('rf', model2),
        ('gb', model3)
    ])

    ensemble_model.fit(X_train, y_train)

    y_pred = ensemble_model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    print(f" Model Trained | Mean Squared Error on Test Data: {mse:.2f}")
    return ensemble_model, df

# Predict salary based on user input and show graph
def predict_salary(model, df):
    try:
        exp = float(input("Enter candidate's years of experience: "))
        test_score = float(input("Enter candidate's test score (out of 100): "))
        interview_score = float(input("Enter candidate's interview score (out of 100): "))

        candidate_df = pd.DataFrame([[exp, test_score, interview_score]],
                                    columns=['Experience', 'Test_Score', 'Interview_Score'])

        predicted_salary = model.predict(candidate_df)
        predicted_value = round(predicted_salary[0], 2)
        print(" Predicted Salary for candidate: ₹", predicted_value)

        # Plotting the prediction
        plt.figure(figsize=(8, 5))
        plt.scatter(df['Experience'], df['Salary'], color='blue', label='Original Data')
        plt.scatter(exp, predicted_value, color='red', s=100, label='Predicted Salary')
        plt.xlabel("Years of Experience")
        plt.ylabel("Salary")
        plt.title("Salary Prediction")
        plt.legend()
        plt.grid(True)
        plt.show()

    except ValueError:
        print("❌ Invalid input! Please enter numerical values.")

# Main function
def main():
    df = create_sample_data()
    model, df = train_model(df)
    while True:
        predict_salary(model, df)
```

```
        cont = input("Do you want to predict another salary? (yes/no): ").strip().lower()
        if cont != 'yes':
            print("👋 Exiting Salary Predictor. Goodbye!")
            break

if __name__ == "__main__":
    main()
```

```
••• Model Trained | Mean Squared Error on Test Data: 5961760.37
Enter candidate's years of experience: 4
Enter candidate's test score (out of 100): 60
Enter candidate's interview score (out of 100): 96
 Predicted Salary for candidate: ₹ 43412.59
```



Start coding or generate with AI.