ML_Assignment_3-Regression

[2]:
```
import pandas as pd
from sklearn.datasets import fetch_california_housing
from sklearn.preprocessing import StandardScaler

# Load the dataset
data = fetch_california_housing()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target, name="MedHouseVal")

# Handle missing values (check if any exist)
print(X.isnull().sum())  # Display any missing values (if any)

# Feature scaling: Standardization
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Convert scaled features into a DataFrame for better readability
X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)

# Print first few rows to inspect
print(X_scaled_df.head())
```

```
MedInc        0
HouseAge      0
AveRooms      0
AveBedrms     0
Population    0
AveOccup      0
```

```
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled_df, y, test_size=0.2, random_state=42)

# Initialize models
models = {
    "Linear Regression": LinearRegression(),
    "Decision Tree": DecisionTreeRegressor(),
    "Random Forest": RandomForestRegressor(),
    "Gradient Boosting": GradientBoostingRegressor(),
    "SVR": SVR()
}

# Train and predict using each model
for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"{name} - Predictions: {y_pred[:5]}")  # Displaying the first 5 predictions
```

```
Linear Regression - Predictions: [0.71912284 1.76401657 2.70965883 2.83892593 2.60465725]
Decision Tree - Predictions: [0.425   1.203   5.00001 2.225   2.257  ]
```

```
Mean Squared Error (MSE)
Mean Absolute Error (MAE)
R-squared Score (R²)
```

[*]:
```python
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Evaluate models
evaluation_metrics = {}

for name, model in models.items():
    y_pred = model.predict(X_test)

    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    evaluation_metrics[name] = {"MSE": mse, "MAE": mae, "R²": r2}

# Convert to DataFrame for easy comparison
metrics_df = pd.DataFrame(evaluation_metrics).T
print(metrics_df)
```