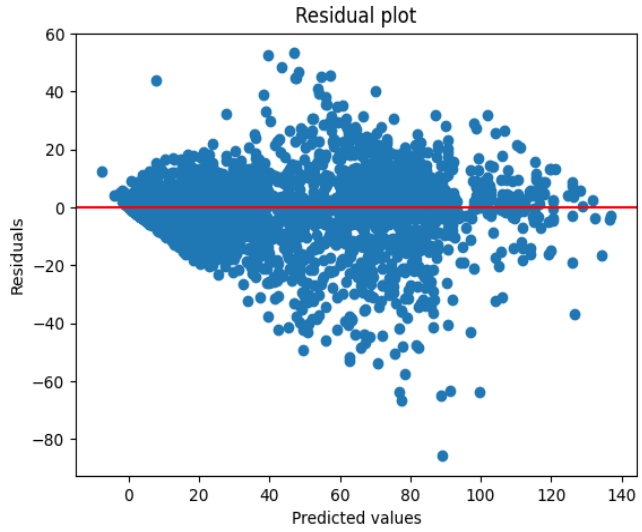
Decision tree result

	Actual Values	Predicted Values
0	6.40	11.530733
1	91.20	84.448205
2	38.00	28.000000
3	19.00	17.375000
4	11.00	11.233333
...	...	...
4248	83.60	82.000000
4249	83.50	30.000000
4250	3.50	3.450000
4251	8.70	10.500000
4252	2.15	4.500000

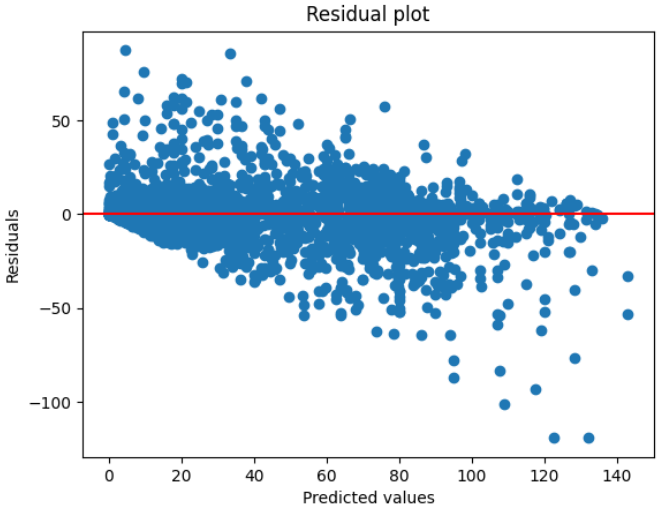
XGBoost result

	Actual Values	Predicted Values
0	6.40	9.724865
1	91.20	85.745049
2	38.00	47.579231
3	19.00	21.558552
4	11.00	6.634473
...	...	...
4248	83.60	81.467155
4249	83.50	72.375893
4250	3.50	3.899378
4251	8.70	8.729270
4252	2.15	4.551018

Residual plot for XGBoost



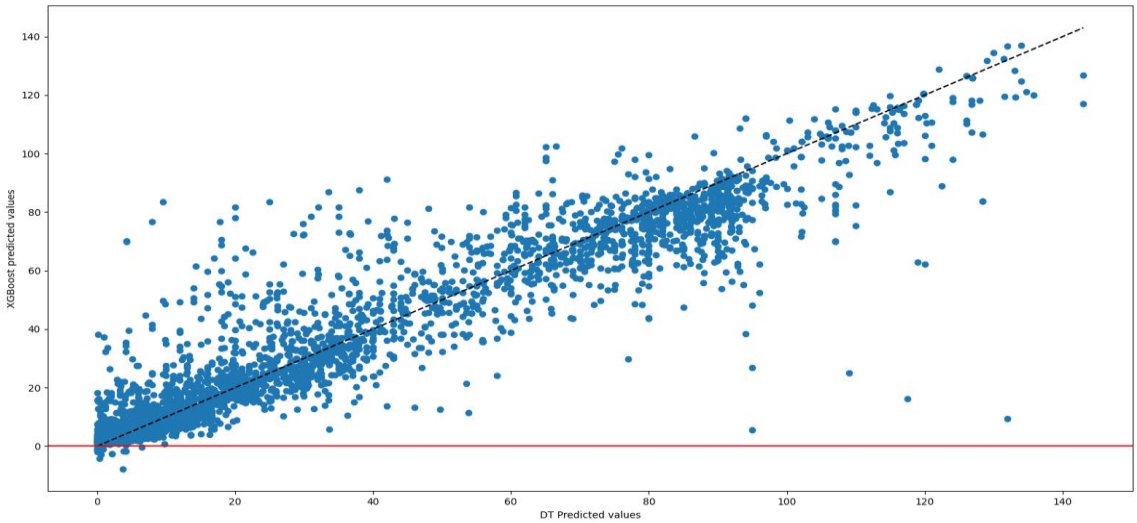
Residual plot for Decision tree



Cross Validation scores(accuracy) to compare XGBoost model and DT regression model

```
Accuracy scores in each fold for XGBoost model: [0.67565979 0.656601   0.85646636 0.74357251 0.6307637 ]
Mean accuracy for XGBoost model: 0.7126126733225717
Standard deviation of accuracy for XGBoost model: 0.08106105254191609
Accuracy scores in each fold for DT regresion model: [0.41554525 0.43576922 0.74430851 0.5322349  0.49191796]
Mean accuracy for DT model: 0.523955167878644
Standard deviation of accuracy for DT model: 0.117626530399644
```

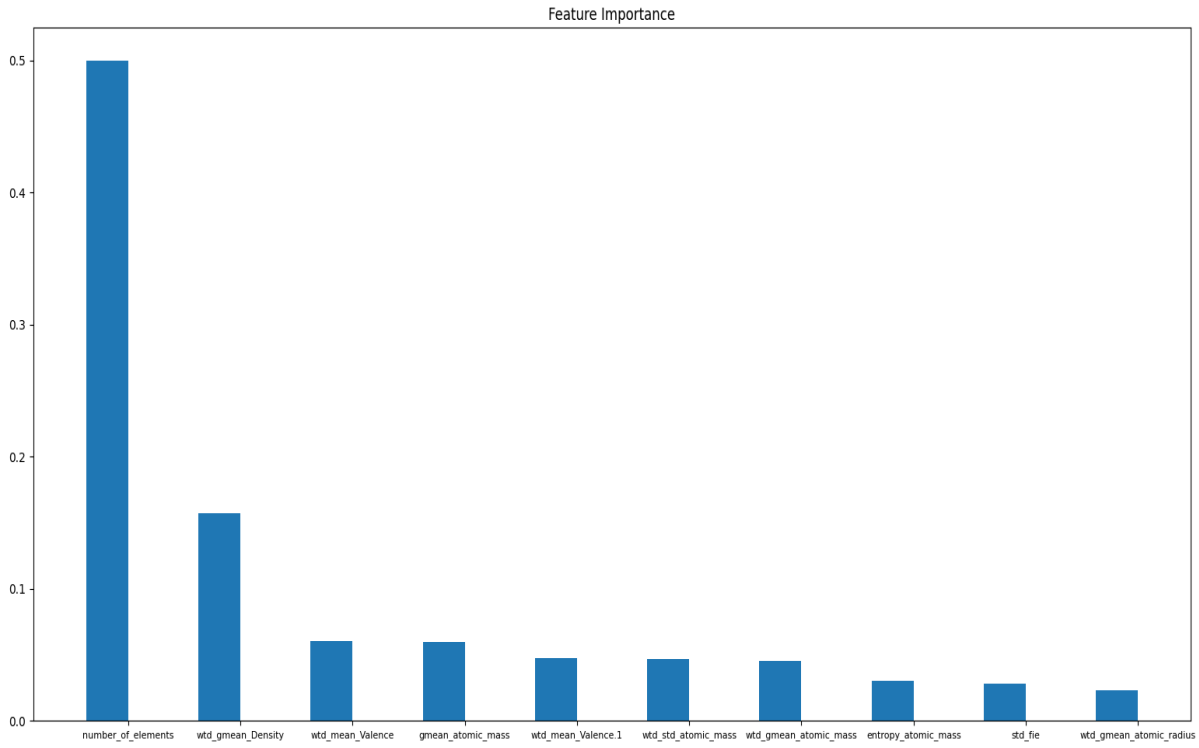
Comparison of Predicted values for XGBoost model and DT model



Top 10 Features selected according to feature importance score from 81 features in the dataset

```
Top 10 Features:
1. number_of_elements (0.499703)
2. wtd_gmean_Density (0.157040)
3. wtd_mean_Valence (0.061633)
4. gmean_atomic_mass (0.057693)
5. wtd_std_atomic_mass (0.048229)
6. wtd_mean_Valence.1 (0.047896)
7. wtd_gmean_atomic_mass (0.045947)
8. entropy_atomic_mass (0.029385)
9. std_fie (0.028169)
```

Bin plot for feature importance



R2_score for DT model and XGBoost model

```
DT R2: 0.860522549949941
XGBoost R2: 0.9108149775654926
```

DT model

```
# Decision tree
dtr = DecisionTreeRegressor(random_state=42)
dtr.fit(X_train, y_train)
dtr_pred = dtr.predict(X_test)
```

XGBoost model

```
# XGBoost
model = xgb.XGBRegressor(objective='reg:squarederror', n_estimators=1000,
                         max_depth=5, learning_rate=0.2, colsample_bytree=0.5)
model.fit(X_train, y_train)

model_pred = model.predict(X_test)
```

Splitting the data into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
```