Part A: Mushroom Classification

Prediction Task

The prediction task is to create a machine learning model to **predict a mushroom's edibility**, either edible or poisonous, based on the mushroom's attributes.

Output Variable

The output variable is a **two-class** label - **edible** or **poisonous**. Edible (e) is defined the mushroom would cause no hard when consumed; whereas poisonous (p) refers to the mushroom will cause harmful effects when eaten.

Data Profile

Overview | Warnings 47 | Reproduction

Dataset statistics

Number of variables	23
Number of observations	8124
Missing cells	2480
Missing cells (%)	1.3%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	1.4 MiB
Average record size in memory	184.0 B

Variable types

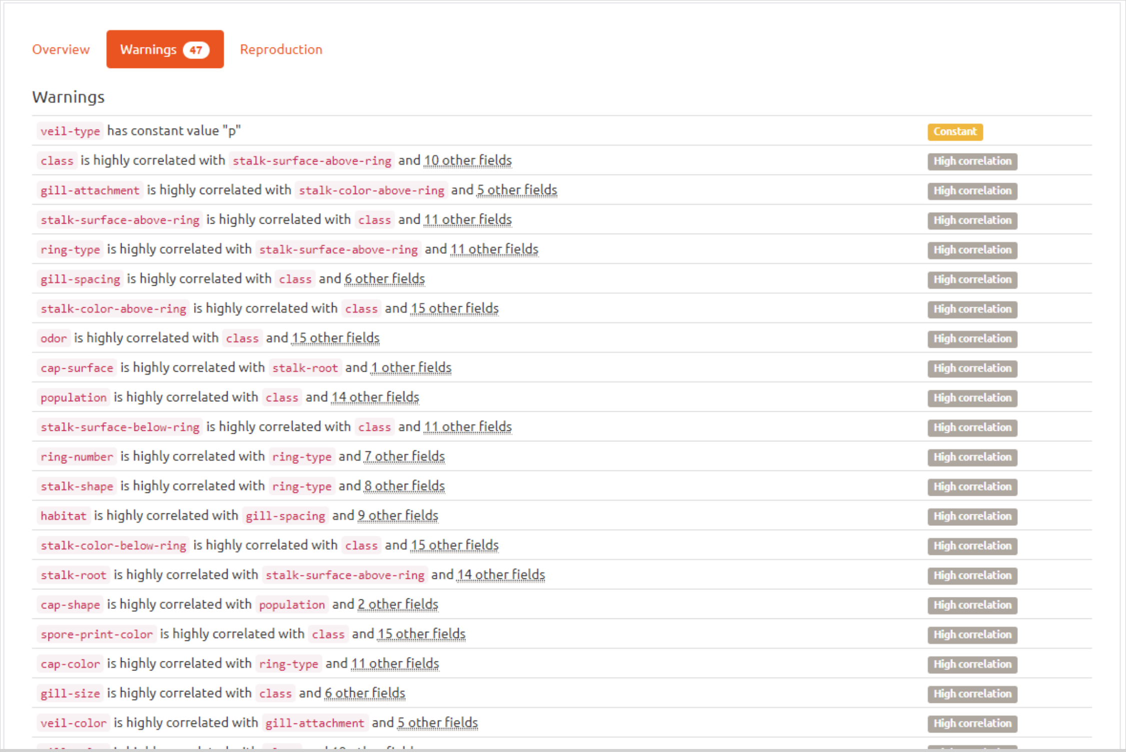
Categorical	22
Boolean	1

Warnings

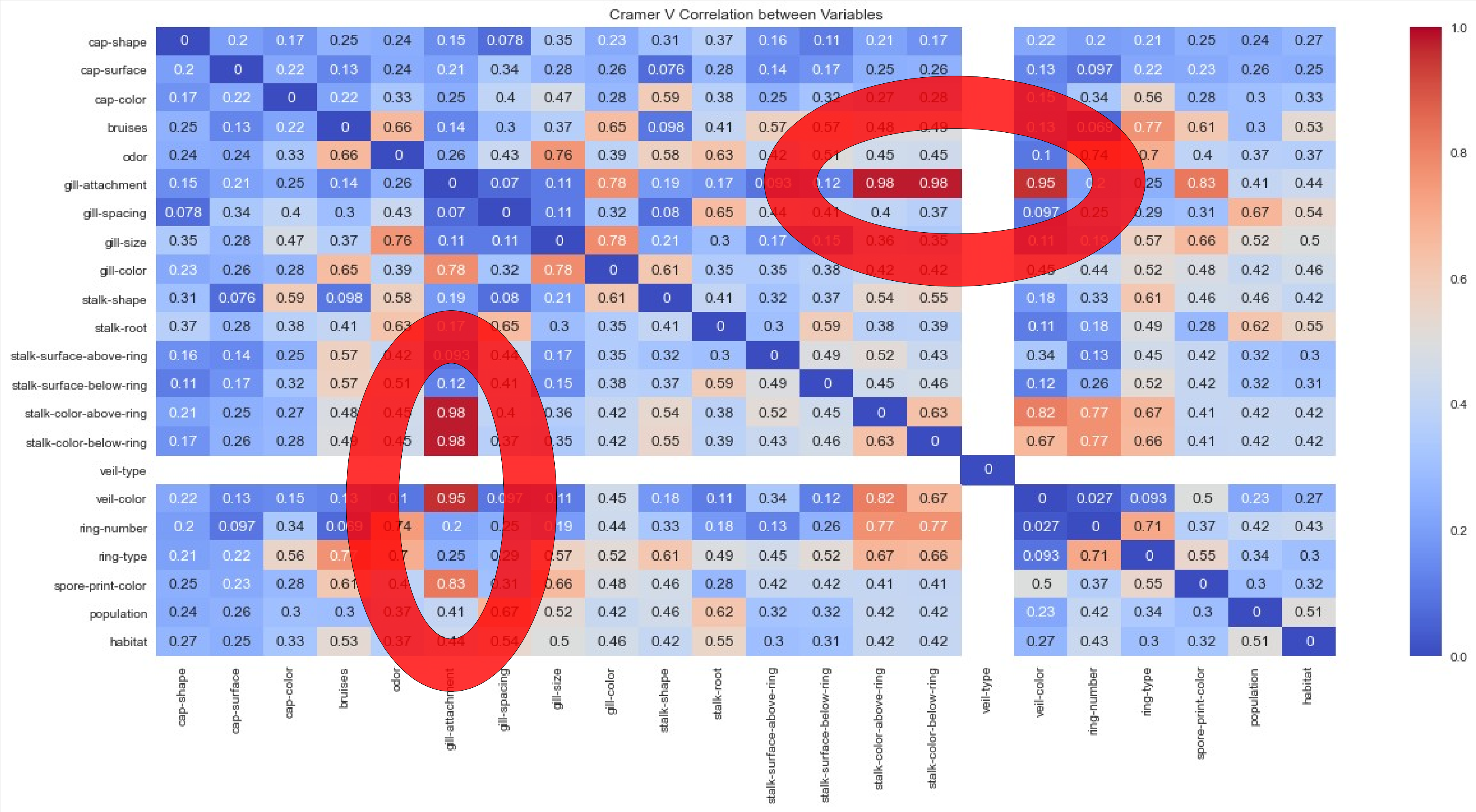
Profile Reports shows **47 Warnings**

Key points:

- All of the columns are categorical data
  - requires encoding
- “veil-type” contains only one class
  - drop column since there is no variance
- “stalk-root” has 30.5% missing data
  - either impute or drop the entire column
- high correlation between features
  - have to check for multicollinearity



Cramer's V Multicollinearity Feature Selection



Theil's U – Univariate Feature Selection

Key Points to take down:

- These columns have low correlation with edibility, where U < 0.05
- cap-shape, cap-surface, cap-color, gill-attachment, stalk-shape, veil-type, veil-color, ring-number

Action Plan

- Drop all these columns show below

	cap-shape	cap-surface	cap-color	gill-attachment	stalk-shape	veil-type	veil-color	ring-number
class	0.048842	0.028617	0.036083	0.014178	0.007524	-1.603208e-16	0.023839	0.038489

Pipeline/Feature Engineering

```
scoring = ["accuracy", "balanced_accuracy", "f1", 'roc_auc']

for name, model in models:
  pipeline = Pipeline(
    steps=[
        ("Imputation", SimpleImputer(strategy="most_frequent")),
        ("One-Hot Encoding", OneHotEncoder(drop="first", sparse=False)),
        (name, model)
    ]
  )
  cross_validate(pipeline, X_train, y_train, scoring=scoring)
```

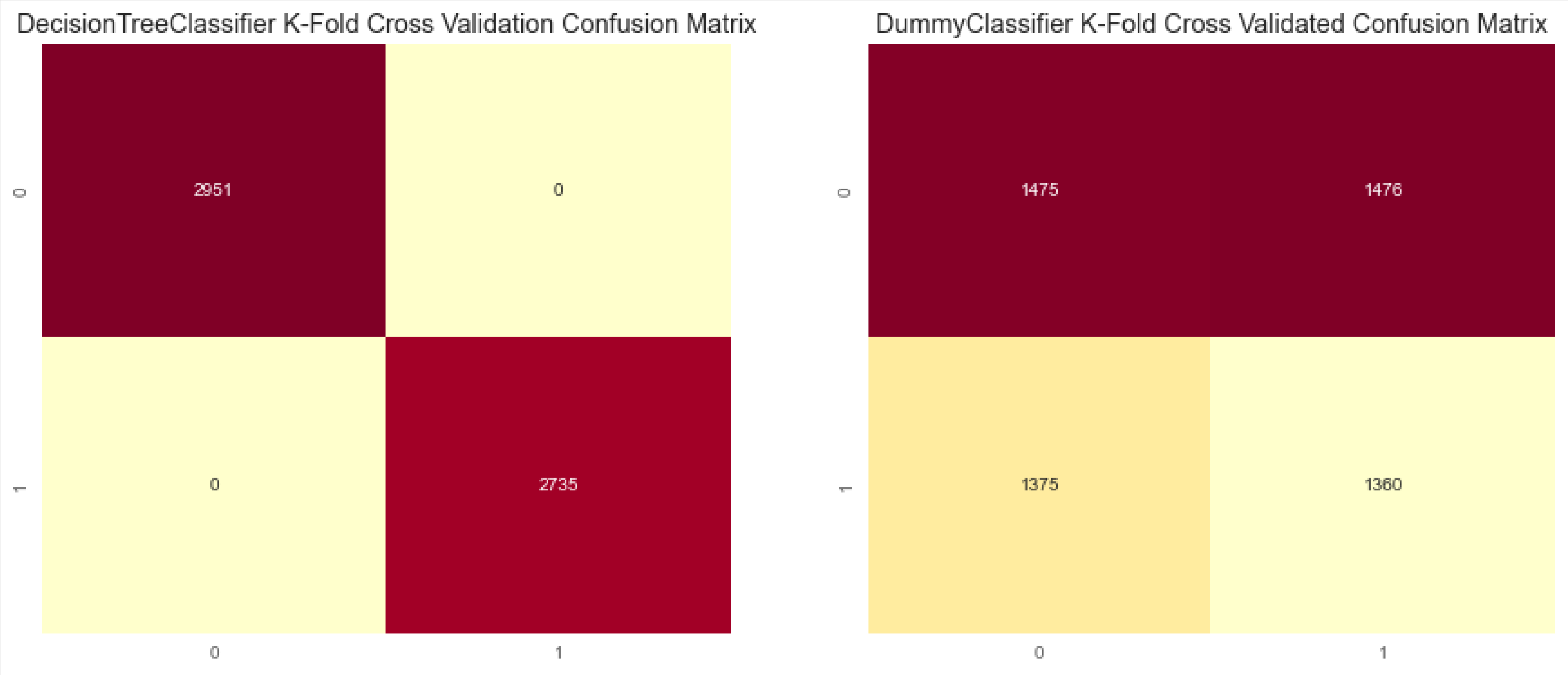
Model Selection

Most of the models I have preselected has 1.0 accuracy.

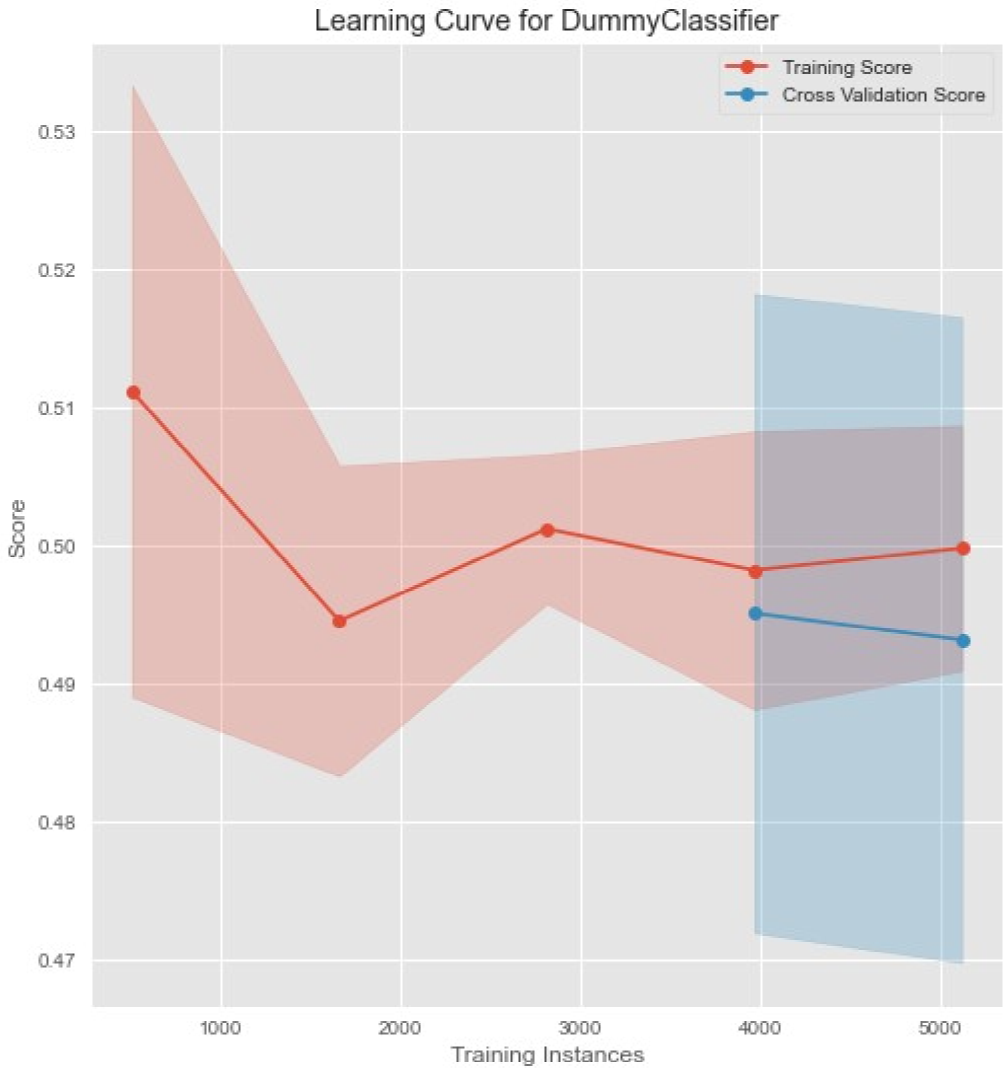
Since there is a dilemma in choosing learning algorithms, I decided to go with the simplest model with the best interpretibility, Decision Tree Classifier.

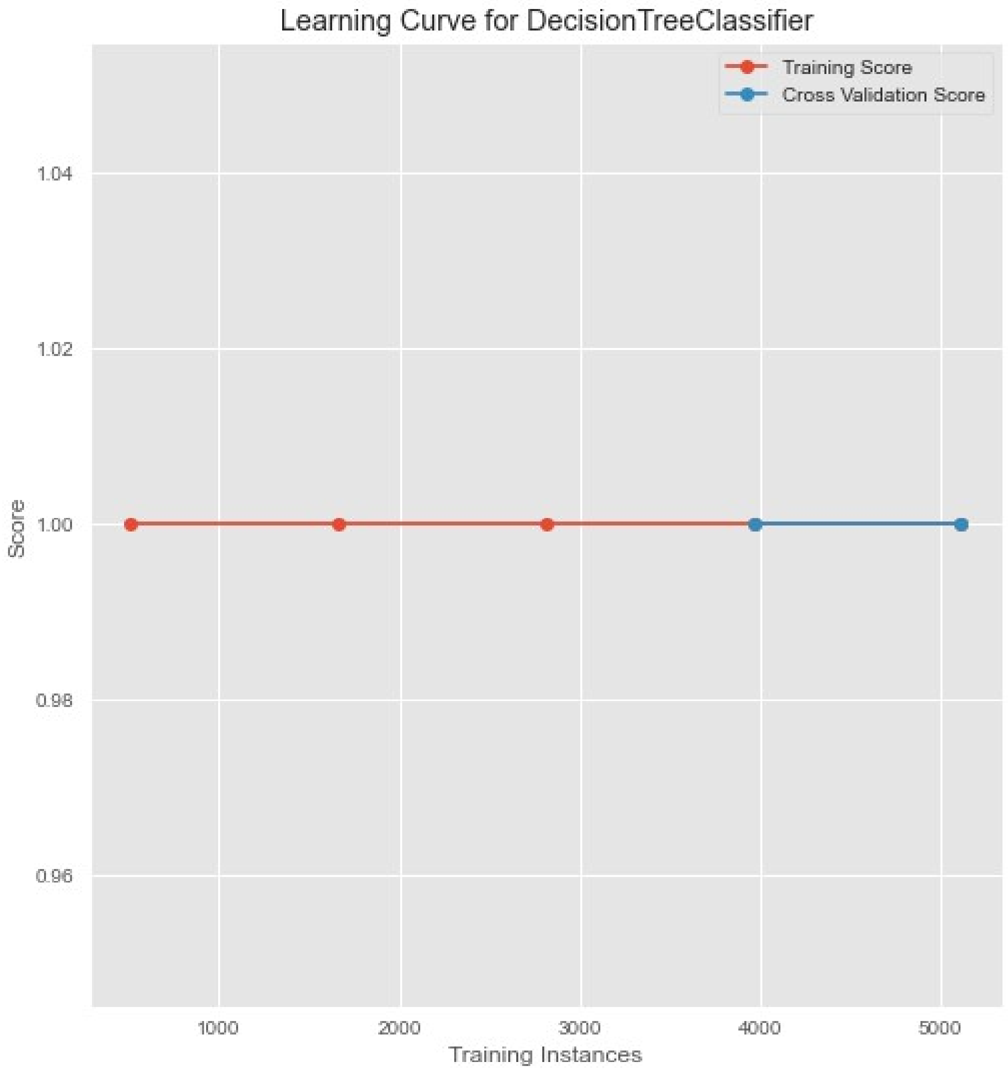
	fit_time	score_time	test_accuracy	test_balanced_accuracy	test_f1	test_roc_auc
DecisionTreeClassifier	0.036299	0.015959	1.000000	1.000000	1.000000	1.000000
RandomForestClassifier	0.294108	0.050652	1.000000	1.000000	1.000000	1.000000
AdaBoostClassifier	0.549287	0.079787	1.000000	1.000000	1.000000	1.000000
GradientBoostingClassifier	1.052442	0.024467	1.000000	1.000000	1.000000	1.000000
Perceptron	0.043483	0.032352	1.000000	1.000000	1.000000	1.000000
Linear SVC	0.139350	0.035562	1.000000	1.000000	1.000000	1.000000
Polynomial SVC	0.359826	0.079387	1.000000	1.000000	1.000000	1.000000
ExtraTreesClassifier	0.372700	0.067840	1.000000	1.000000	1.000000	1.000000
MLPClassifier	1.763953	0.037759	1.000000	1.000000	1.000000	1.000000
CalibratedClassifierCV	0.203737	0.043484	1.000000	1.000000	1.000000	1.000000
SGDClassifier	0.048475	0.029321	1.000000	1.000000	1.000000	1.000000
KNeighborsClassifier	0.045479	0.701378	0.999472	0.999452	0.999450	1.000000
RBF SVC	0.421568	0.208354	0.999472	0.999452	0.999450	1.000000
RidgeClassifier	0.050712	0.033710	0.999120	0.999086	0.999084	1.000000
RidgeClassifierCV	0.109710	0.035540	0.999120	0.999086	0.999084	1.000000
LogisticRegression	0.112658	0.034307	0.999120	0.999086	0.999084	0.999987
GaussianNB	0.034707	0.025941	0.974147	0.975027	0.973790	0.996524
Sigmoid SVC	0.334529	0.079947	0.964649	0.964138	0.962766	0.985698

Comparing to Baseline – Confusion Matrix



Comparing to Baseline – Learning Curve

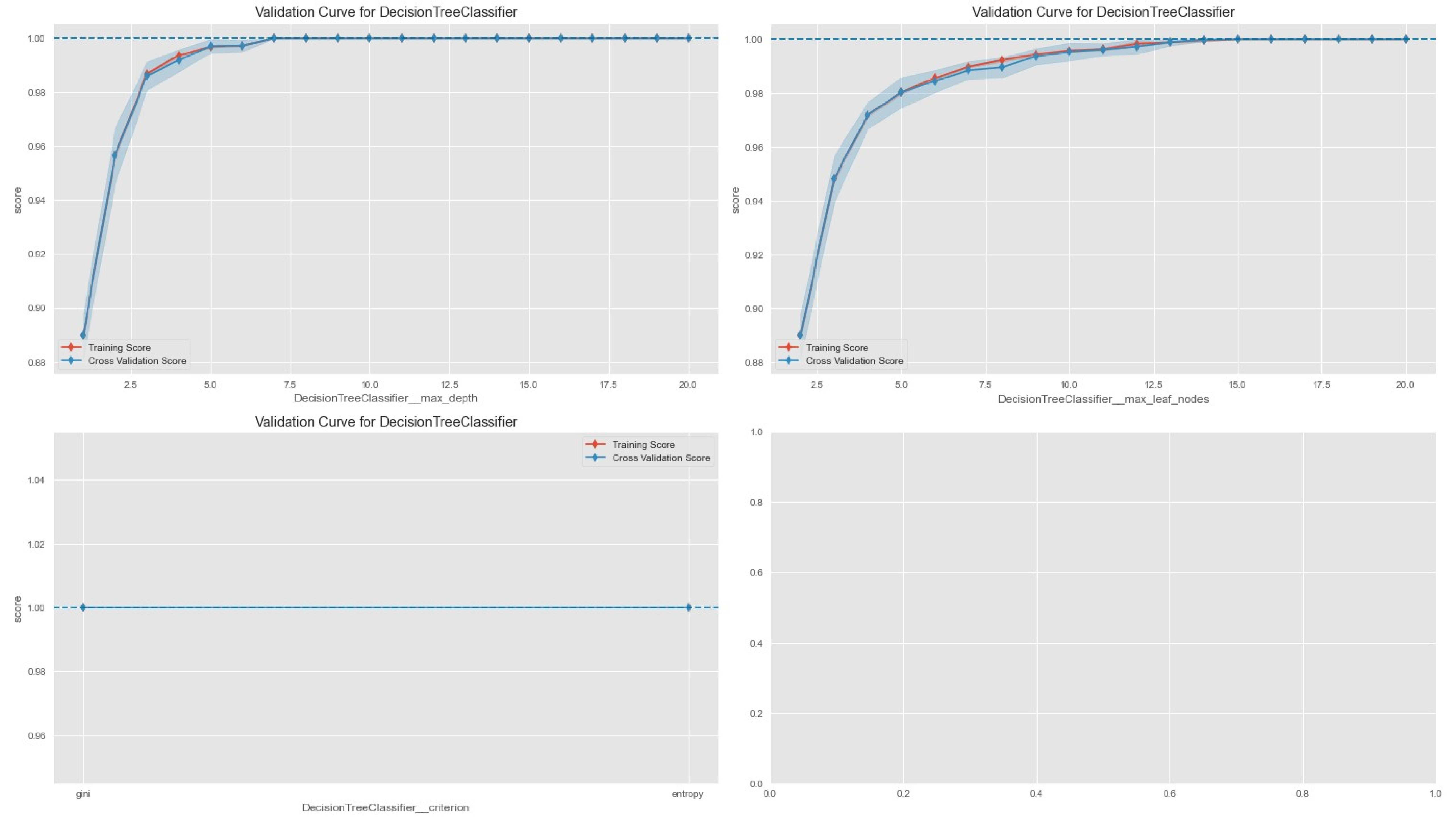




Hyperparameter Tuning

```python
filterwarnings('ignore')
# Create the parameter grid
params_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': np.arange(5, 15),
    'max_leaf_nodes': np.arange(10, 16)
}
# Creating a model based on the pipeline
grid_search = Pipeline(
    steps=[
        ('SimpleImputer', imp),
        ("OneHotEncoder", onehot),
        ("GridSearchCV", GridSearchCV(
            DecisionTreeClassifier(min_samples_split=2, min_samples_leaf=1),
            params_grid,
            cv=5,
            verbose=2,
            n_jobs=4,
            scoring='accuracy'
            )
        )
    ]
)
# Fitting Model
grid_search.fit(X_train, y_train)
print(grid_search.named_steps['GridSearchCV'].best_estimator_)
print(grid_search.named_steps['GridSearchCV'].best_params_)
print(grid_search.named_steps['GridSearchCV'].best_score_)
```

```
Fitting 5 folds for each of 120 candidates, totalling 600 fits
DecisionTreeClassifier(max_depth=7, max_leaf_nodes=14)
{'criterion': 'gini', 'max_depth': 7, 'max_leaf_nodes': 14}
1.0
```



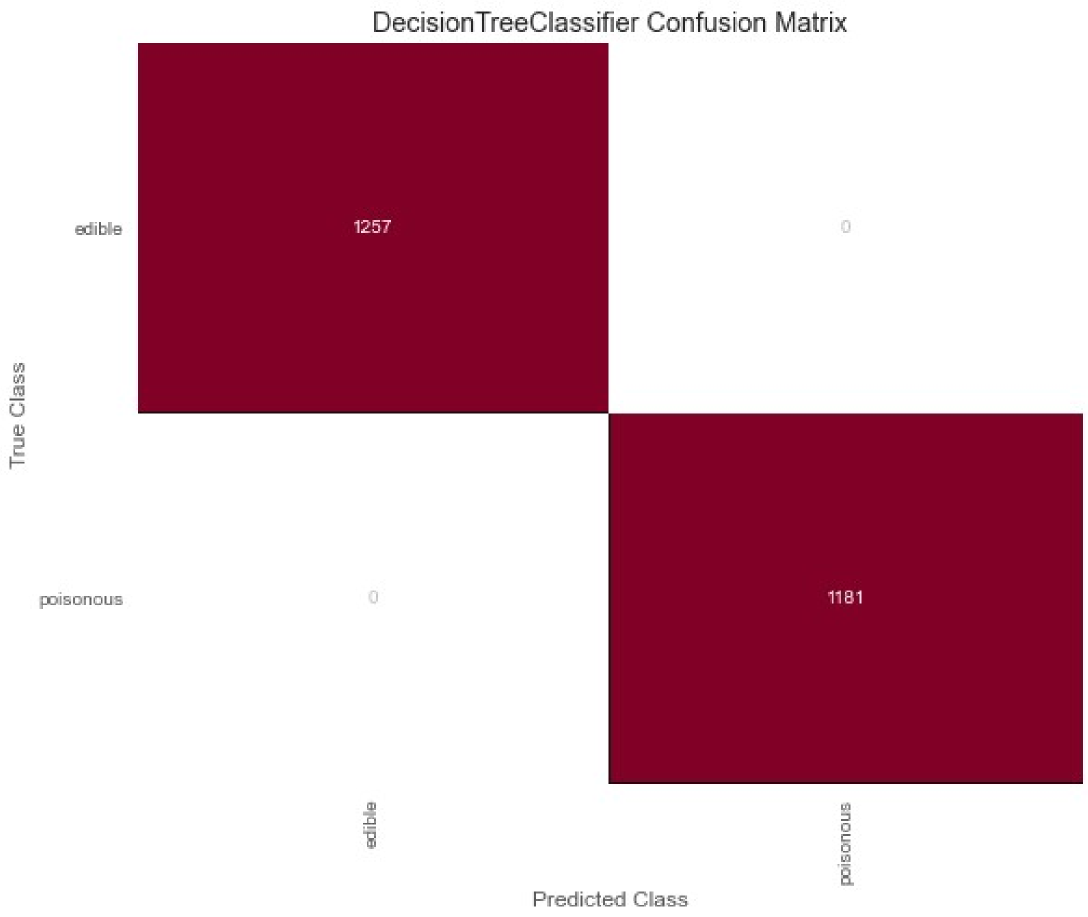
Validation Curve for DecisionTreeClassifier
Training Score
Cross Validation Score
score
DecisionTreeClassifier__max_depth
Validation Curve for DecisionTreeClassifier
Training Score
Cross Validation Score
score
DecisionTreeClassifier__max_leaf_nodes
Validation Curve for DecisionTreeClassifier
Training Score
Cross Validation Score
score
gini
entropy
DecisionTreeClassifier__criterion

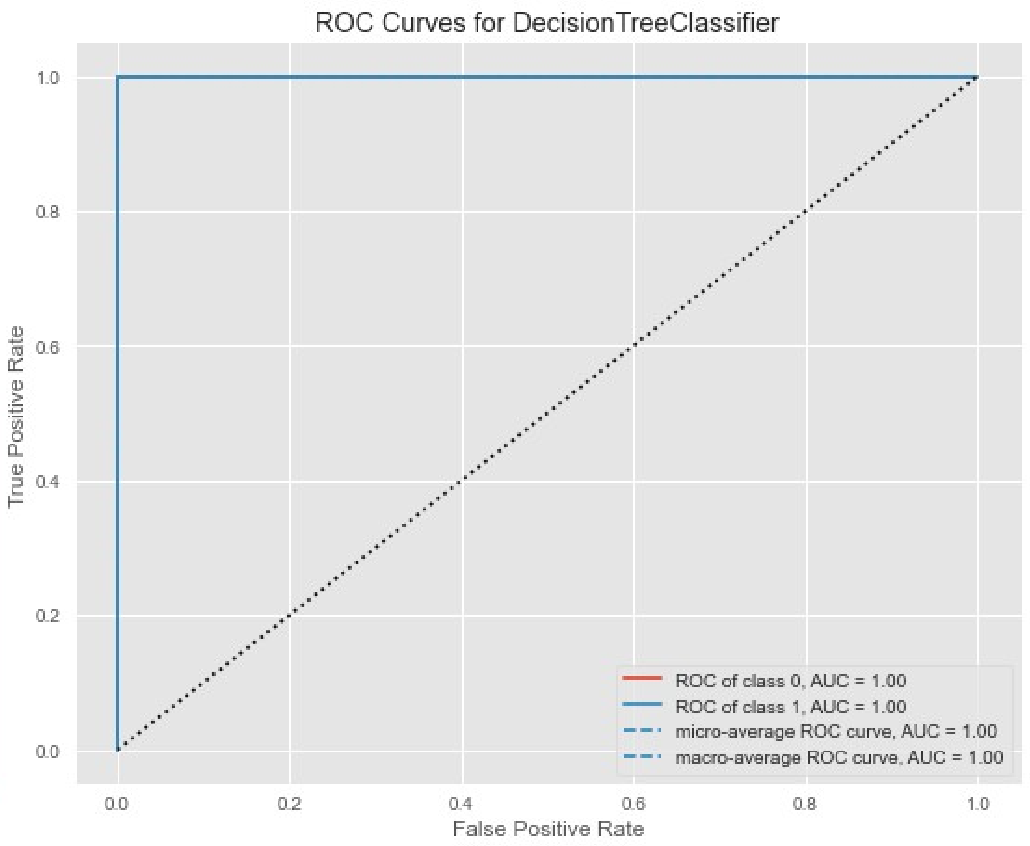
Model Evaluation – Learning Curve

Model Evaluation

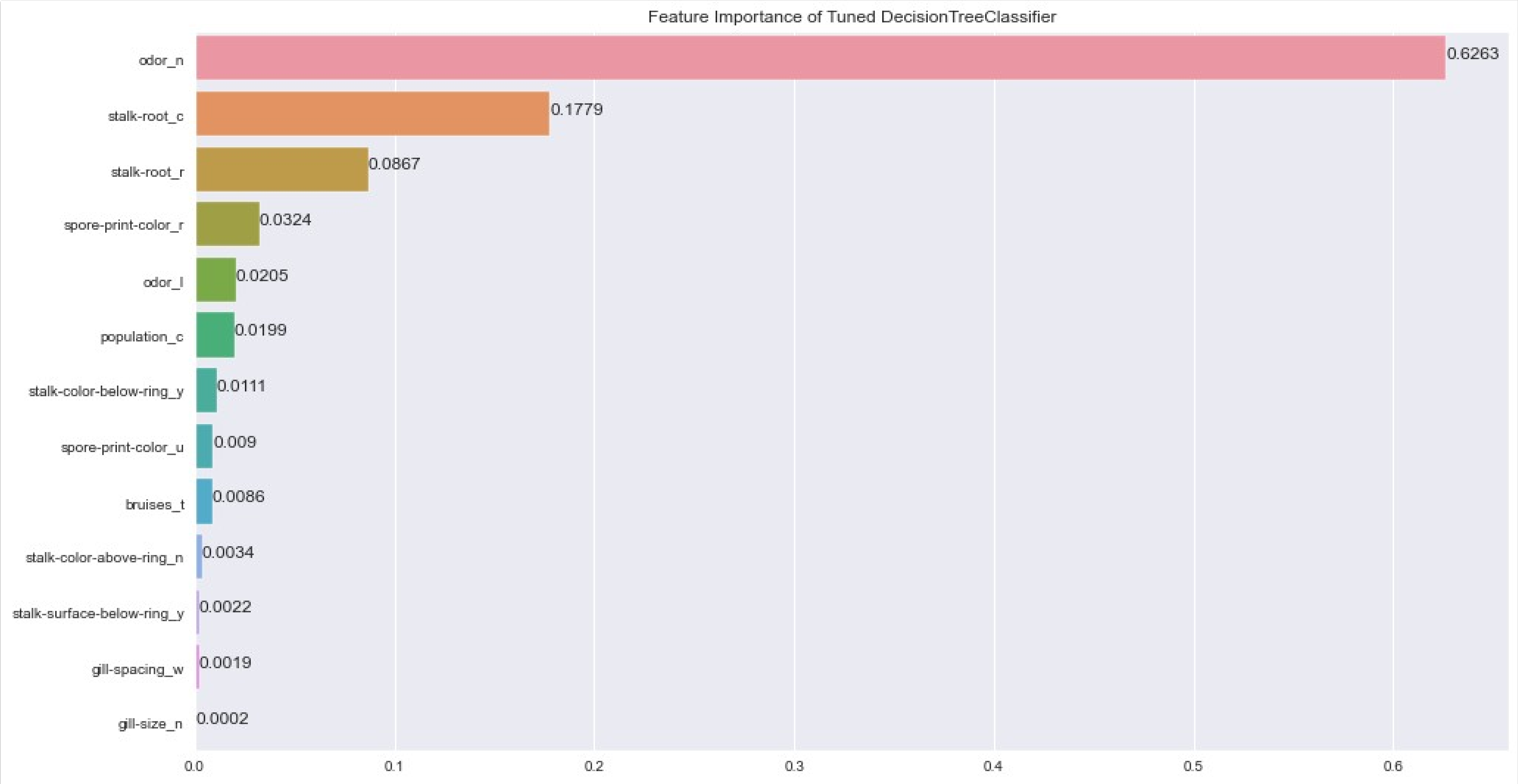
```
              precision    recall  f1-score   support

           0      1.000     1.000     1.000      1257
           1      1.000     1.000     1.000      1181

    accuracy                          1.000      2438
   macro avg      1.000     1.000     1.000      2438
weighted avg      1.000     1.000     1.000      2438

Cross Validation Scores: [1. 1. 1. 1. 1.]
Mean Cross Validation Scores: 1.0
```

Model Evaluation





Model Evaluation – Feature Importance



Part B: Kings County House Price Prediction

Prediction Task
The prediction task is to create a predictive regression model to predict the house prices based of the house sales’ attribute given.

Output Variable
The output variable is `price`, refering to the price of the house sale. The variable is numerical-continuous variable, as such the prediction task requires a regression model.

Data Profile

Overview | Warnings 55 | Reproduction

Dataset statistics

Number of variables	21
Number of observations	21613
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	3.5 MiB
Average record size in memory	168.0 B

Variable types

Numeric	17
DateTime	1
Categorical	3

Warnings

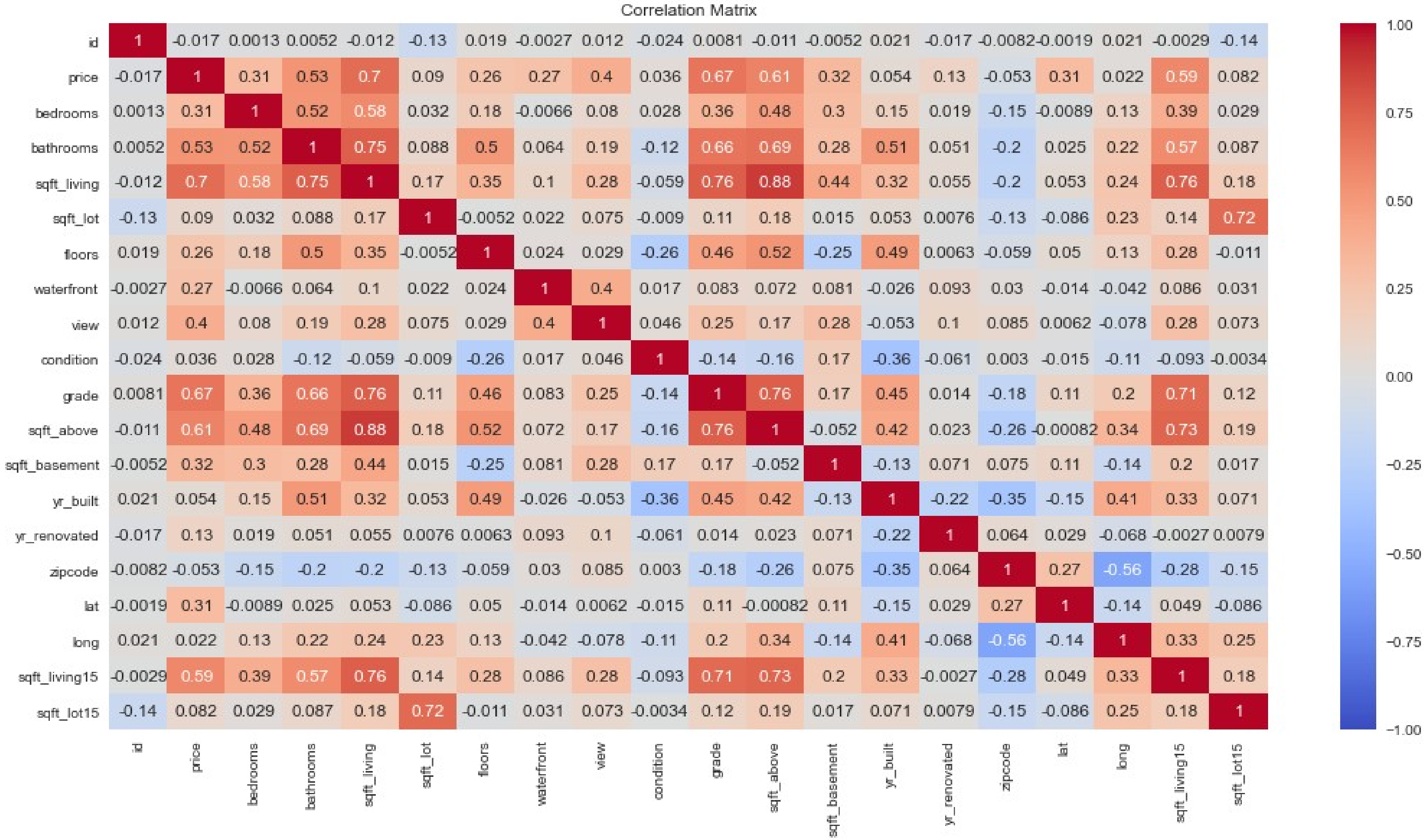
Profile Reports shows **55 Warnings**

Key points:

- high correlation between features
  - have to check for multicollinearity
- ‘sqft_basement’ has 60.7% zeros
- ‘yr_renovated’ has 95.8% zeros

sqft_basement is highly correlated with sqft_above and 3 other fields	High correlation
sqft_above is highly correlated with sqft_living15 and 6 other fields	High correlation
zipcode is highly correlated with lat and 2 other fields	High correlation
sqft_living is highly correlated with sqft_living15 and 6 other fields	High correlation
long is highly correlated with zipcode and 1 other fields	High correlation
grade is highly correlated with sqft_living15 and 4 other fields	High correlation
bathrooms is highly correlated with sqft_living15 and 7 other fields	High correlation
yr_built is highly correlated with condition and 4 other fields	High correlation
bedrooms is highly correlated with sqft_above and 2 other fields	High correlation
sqft_lot15 is highly correlated with sqft_lot	High correlation
price is highly correlated with sqft_living15 and 5 other fields	High correlation
view is highly correlated with waterfront	High correlation
waterfront is highly correlated with view	High correlation
sqft_basement has 13126 (60.7%) zeros	Zeros
yr_renovated has 20699 (95.8%) zeros	Zeros

Feature Selection Pearson Correlation Matrix



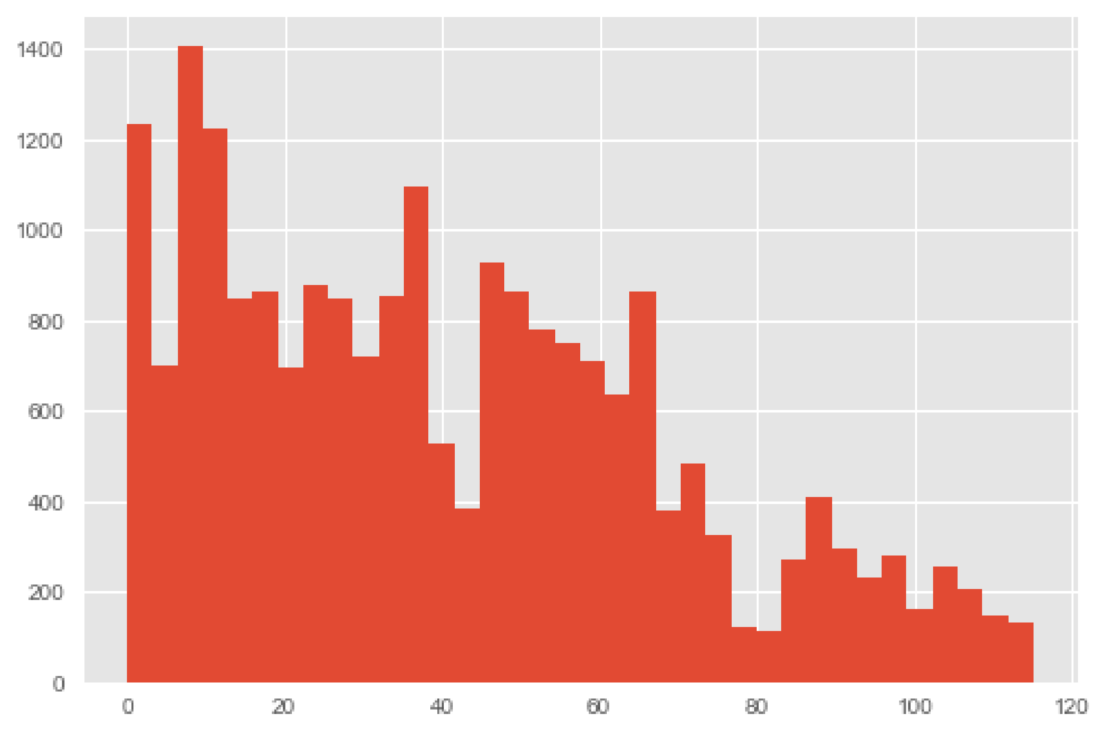
	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15
id	1	-0.017	0.0013	0.0052	-0.012	-0.13	0.019	-0.0027	0.012	-0.024	0.0081	-0.011	-0.0052	0.021	-0.017	-0.0082	-0.0019	0.021	-0.0029	-0.14
price	-0.017	1	0.31	0.53	0.7	0.09	0.26	0.27	0.4	0.036	0.67	0.61	0.32	0.054	0.13	-0.053	0.31	0.022	0.59	0.082
bedrooms	0.0013	0.31	1	0.52	0.58	0.032	0.18	-0.0066	0.08	0.028	0.36	0.48	0.3	0.15	0.019	-0.15	-0.0089	0.13	0.39	0.029
bathrooms	0.0052	0.53	0.52	1	0.75	0.088	0.5	0.064	0.19	-0.12	0.66	0.69	0.28	0.51	0.051	-0.2	0.025	0.22	0.57	0.087
sqft_living	-0.012	0.7	0.58	0.75	1	0.17	0.35	0.1	0.28	-0.059	0.76	0.88	0.44	0.32	0.055	-0.2	0.053	0.24	0.76	0.18
sqft_lot	-0.13	0.09	0.032	0.088	0.17	1	-0.0052	0.022	0.075	-0.009	0.11	0.18	0.015	0.053	0.0076	-0.13	-0.086	0.23	0.14	0.72
floors	0.019	0.26	0.18	0.5	0.35	-0.0052	1	0.024	0.029	-0.26	0.46	0.52	-0.25	0.49	0.0063	-0.059	0.05	0.13	0.28	-0.011
waterfront	-0.0027	0.27	-0.0066	0.064	0.1	0.022	0.024	1	0.4	0.017	0.083	0.072	0.081	-0.026	0.093	0.03	-0.014	-0.042	0.086	0.031
view	0.012	0.4	0.08	0.19	0.28	0.075	0.029	0.4	1	0.046	0.25	0.17	0.28	-0.053	0.1	0.085	0.0062	-0.078	0.28	0.073
condition	-0.024	0.036	0.028	-0.12	-0.059	-0.009	-0.26	0.017	0.046	1	-0.14	-0.16	0.17	-0.36	-0.061	0.003	-0.015	-0.11	-0.093	-0.0034
grade	0.0081	0.67	0.36	0.66	0.76	0.11	0.46	0.083	0.25	-0.14	1	0.76	0.17	0.45	0.014	-0.18	0.11	0.2	0.71	0.12
sqft_above	-0.011	0.61	0.48	0.69	0.88	0.18	0.52	0.072	0.17	-0.16	0.76	1	-0.052	0.42	0.023	-0.26	-0.00082	0.34	0.73	0.19
sqft_basement	-0.0052	0.32	0.3	0.28	0.44	0.015	-0.25	0.081	0.28	0.17	0.17	-0.052	1	-0.13	0.071	0.075	0.11	-0.14	0.2	0.017
yr_built	0.021	0.054	0.15	0.51	0.32	0.053	0.49	-0.026	-0.053	-0.36	0.45	0.42	-0.13	1	-0.22	-0.35	-0.15	0.41	0.33	0.071
yr_renovated	-0.017	0.13	0.019	0.051	0.055	0.0076	0.0063	0.093	0.1	-0.061	0.014	0.023	0.071	-0.22	1	0.064	0.029	-0.068	-0.0027	0.0079
zipcode	-0.0082	-0.053	-0.15	-0.2	-0.2	-0.13	-0.059	0.03	0.085	0.003	-0.18	-0.26	0.075	-0.35	0.064	1	0.27	-0.56	-0.28	-0.15
lat	-0.0019	0.31	-0.0089	0.025	0.053	-0.086	0.05	-0.014	0.0062	-0.015	0.11	-0.00082	0.11	-0.15	0.029	0.27	1	-0.14	0.049	-0.086
long	0.021	0.022	0.13	0.22	0.24	0.23	0.13	-0.042	-0.078	-0.11	0.2	0.34	-0.14	0.41	-0.068	-0.56	-0.14	1	0.33	0.25
sqft_living15	-0.0029	0.59	0.39	0.57	0.76	0.14	0.28	0.086	0.28	-0.093	0.71	0.73	0.2	0.33	-0.0027	-0.28	0.049	0.33	1	0.18
sqft_lot15	-0.14	0.082	0.029	0.087	0.18	0.72	-0.011	0.031	0.073	-0.0034	0.12	0.19	0.017	0.071	0.0079	-0.15	-0.086	0.25	0.18	1

House Age

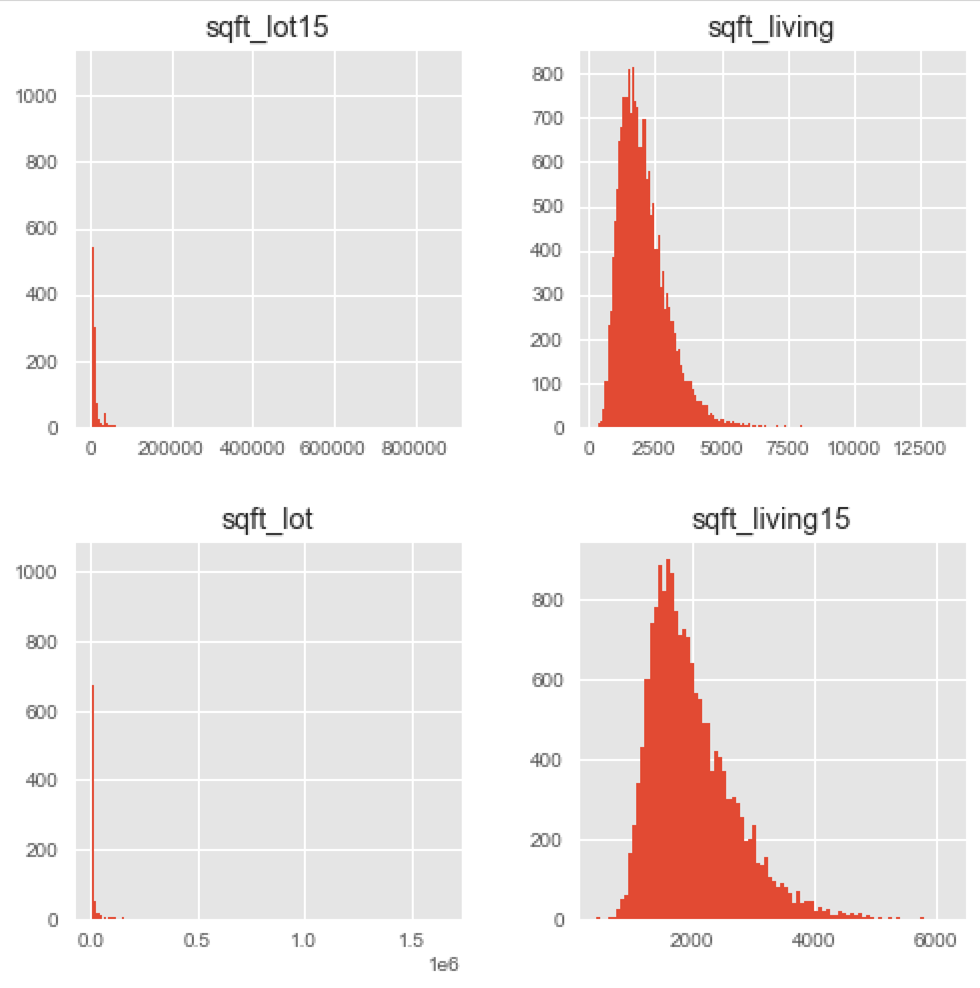
Age referring to the time difference between date and yr_renovated/yr_built

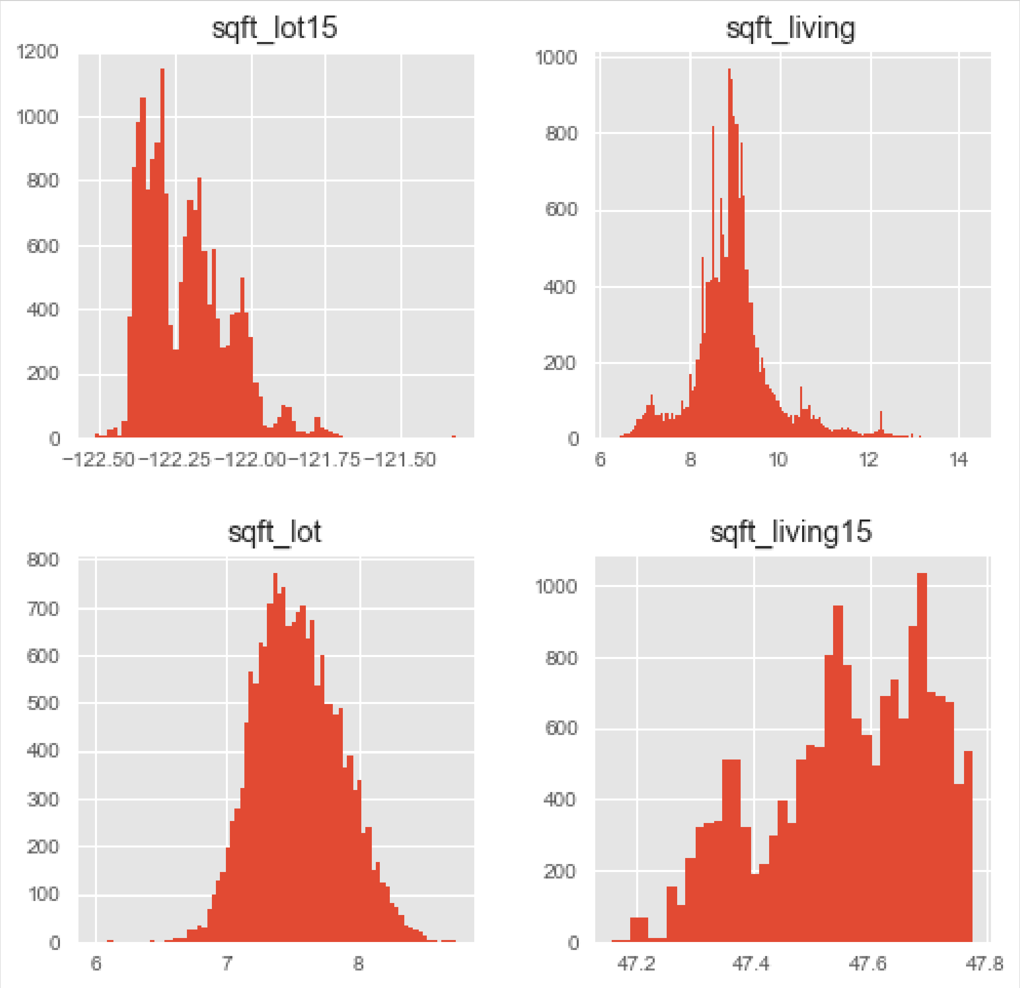
	date	yr_renovated	yr_built	age
0	2014-10-13	0	1955	59.0
1	2014-12-09	1991	1951	23.0
2	2015-02-25	0	1933	82.0
3	2014-12-09	0	1965	49.0
4	2015-02-18	0	1987	28.0
...	...	...	...	...
21608	2014-05-21	0	2009	5.0
21609	2015-02-23	0	2014	1.0
21610	2014-06-23	0	2009	5.0
21611	2015-01-16	0	2004	11.0
21612	2014-10-15	0	2008	6.0

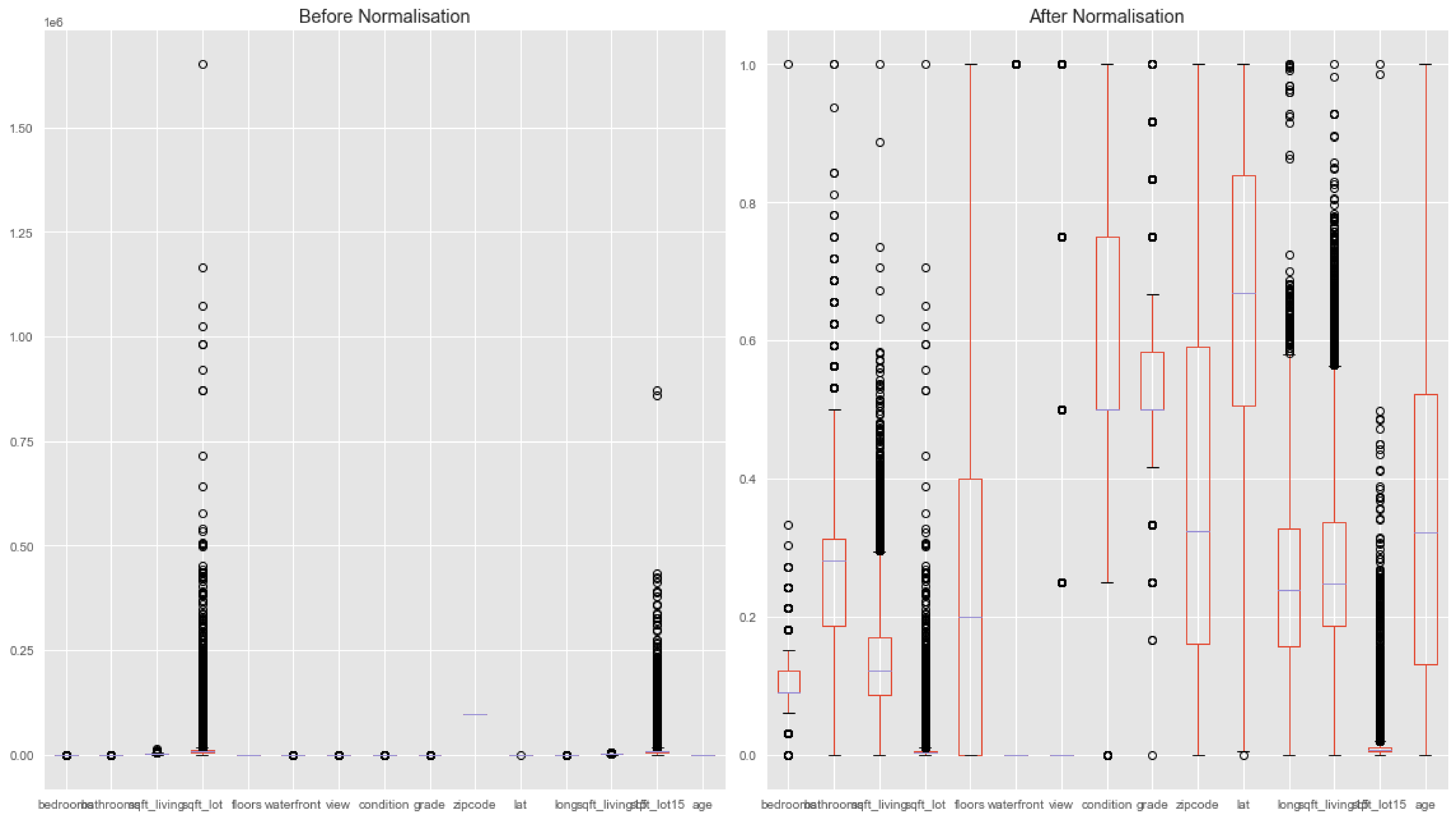
21613 rows × 4 columns

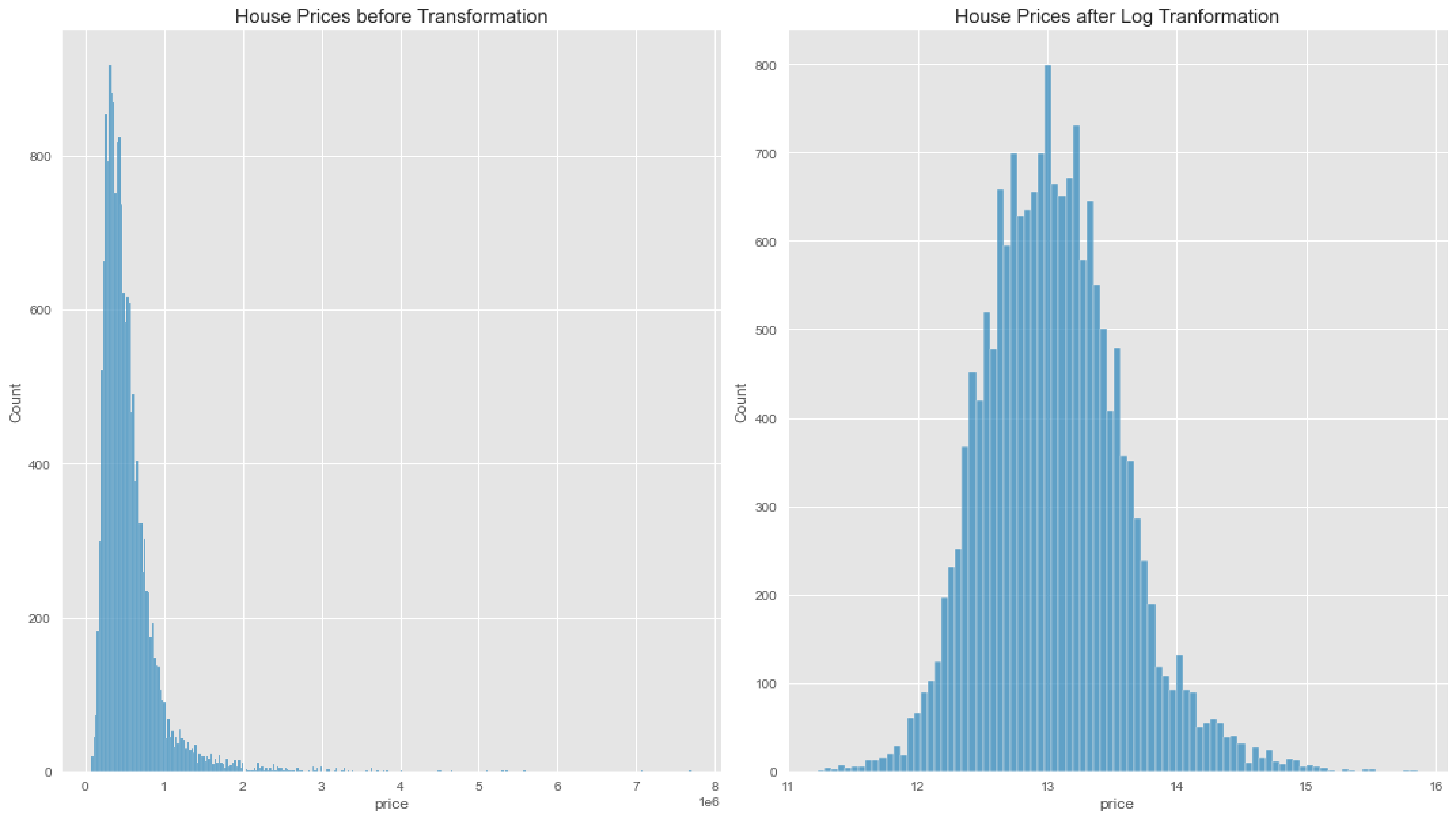


Log Transformation






Before Normalisation
1e6
1.50
1.25
1.00
0.75
0.50
0.25
0.00
bedrooms
bathrooms
sqft_living
sqft_lot
floors
waterfront
view
condition
grade
zipcode
lat
long
sqft_living15
sqft_lot15
age
After Normalisation
1.0
0.8
0.6
0.4
0.2
0.0
bedrooms
bathrooms
sqft_living
sqft_lot
floors
waterfront
view
condition
grade
zipcode
lat
long
sqft_living15
sqft_lot15
age


House Prices before Transformation
Count
price
1e6
0
200
400
600
800
0
1
2
3
4
5
6
7
8
House Prices after Log Tranformation
Count
price
0
100
200
300
400
500
600
700
800
11
12
13
14
15
16

Model Selection

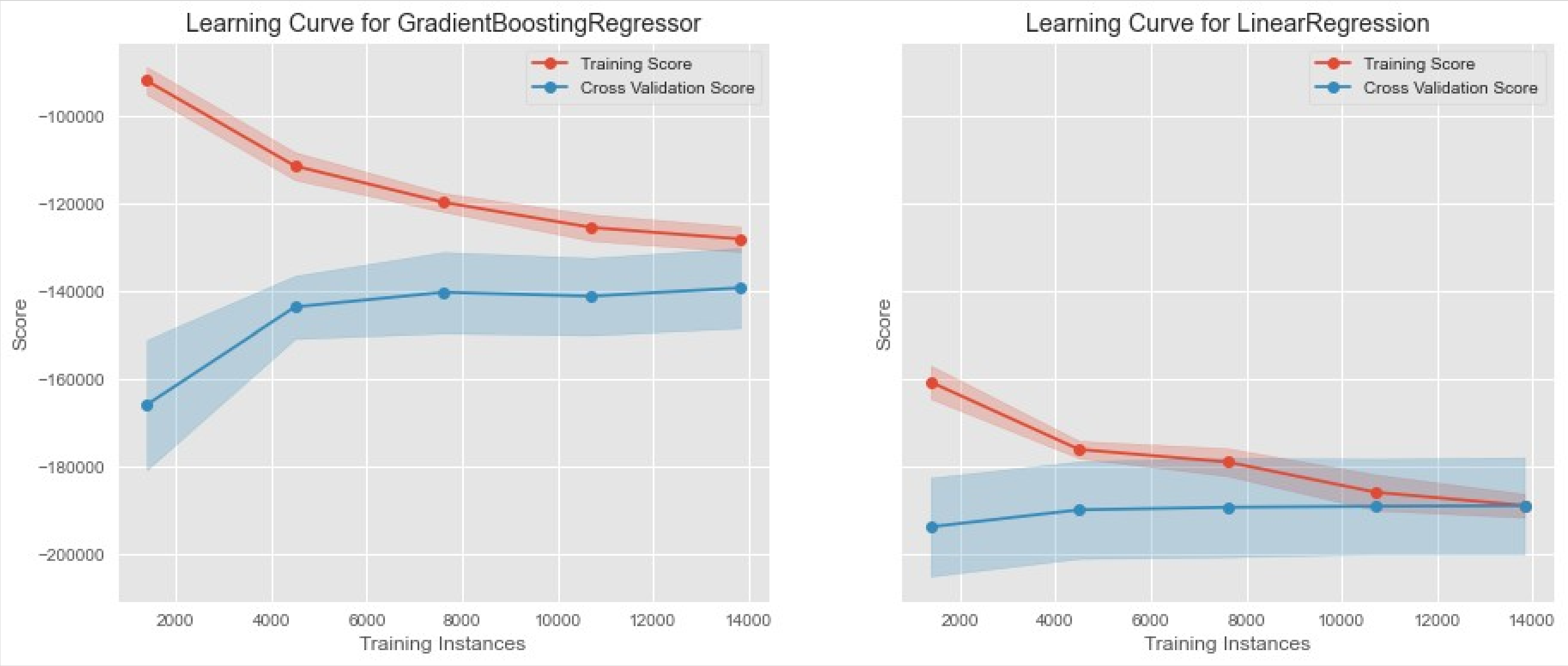
```python
for name, models in models:
    pipeline = Pipeline(
        steps=[
          ('LogTransform', LogTransform),
          ('Normalisation', scaler),
          (name, TransformedTargetRegressor(regressor=model, func=np.log1p, inverse_func=np.expm1))
        ]
    )
    cross_validate(pipeline, X_train, y_train, cv=5, scoring=['RMSE', 'MSE', 'MAE', 'MAPE', 'R2'])
```

Model Selection

	fit_time	score_time	test_rmse	train_rmse	test_mse	train_mse	test_mae	train_mae	test_mape	train_mape	test_r2	train_r2
ExtraTreesRegressor	7.135958	0.200463	-133390.500654	-3068.189155	-17936571347.077385	-9513161.568014	-69219.727082	-123.724826	-0.126745	-0.000425	0.869151	0.999930
RandomForestRegressor	12.393565	0.149601	-135627.651945	-55290.737436	-18525893661.364635	-3058882980.024376	-70347.560765	-26816.512434	-0.128178	-0.047119	0.864615	0.977534
GradientBoostingRegressor	3.347374	0.010773	-139285.123516	-128246.897591	-19486473186.692482	-16455351352.906652	-75522.579562	-71986.147128	-0.137311	-0.131361	0.856904	0.879157
AdaBoostRegressor	1.306047	0.024340	-140606.259676	-64731.613580	-19908085622.554283	-4203027839.285998	-74223.345925	-30492.814420	-0.135643	-0.053883	0.854434	0.969156
BaggingRegressor	1.309004	0.028725	-140874.161656	-63135.779721	-19951648787.887047	-3996438359.867338	-74411.359193	-30475.733821	-0.136075	-0.053843	0.853591	0.970632
SVR	13.303492	4.843579	-145188.994993	-131623.979113	-21304507975.040260	-17330699587.146801	-76776.802559	-72849.505850	-0.139376	-0.134285	0.844945	0.872686
MLPRegressor	15.115033	0.012760	-149213.947738	-141366.393039	-22478985541.641655	-20019688178.613991	-79865.868099	-78126.303800	-0.142699	-0.140868	0.835492	0.852696
KNeighborsRegressor	0.189692	0.985876	-168784.280528	-140807.627813	-28818222820.154930	-19848924340.629433	-84608.095344	-68939.526920	-0.150401	-0.121234	0.790136	0.854273
DecisionTreeRegressor	0.213815	0.008178	-188055.943134	-3068.189155	-35613540755.972511	-9513161.568014	-100716.780634	-123.724826	-0.185233	-0.000425	0.739723	0.999930
LinearRegression	0.624596	0.075797	-188994.705372	-188944.478452	-35841165312.482529	-35707235447.812317	-110814.325837	-110672.289058	-0.201534	-0.201302	0.736630	0.737656
Ridge	0.368858	0.006184	-189350.393436	-189301.885531	-35977847205.701393	-35842467227.082199	-110861.627598	-110719.515040	-0.201544	-0.201316	0.735637	0.736663
SGDRegressor	0.138829	0.007580	-210493.017840	-210375.920353	-44405094031.160751	-44267019889.859184	-125276.401612	-125086.255084	-0.221620	-0.221336	0.672652	0.674762
ElasticNet	0.051257	0.007977	-376119.759146	-376659.424464	-141907449864.942780	-141897913482.805664	-221876.211038	-221875.234014	-0.438628	-0.438620	-0.042641	-0.042515
Lasso	0.021941	0.007579	-376119.759146	-376659.424464	-141907449864.942780	-141897913482.805664	-221876.211038	-221875.234014	-0.438628	-0.438620	-0.042641	-0.042515

Model Selection
Comparing with Baseline

	fit_time	score_time	test_mae	test_mape	test_mse	test_r2	test_rmse
GradientBoostingRegressor	3.382798	0.009982	-75506.759853	-0.137306	-1.943598e+10	0.857226	-139112.218952
LinearRegression	0.021336	0.007188	-110814.325837	-0.201534	-3.584117e+10	0.736630	-188994.705372



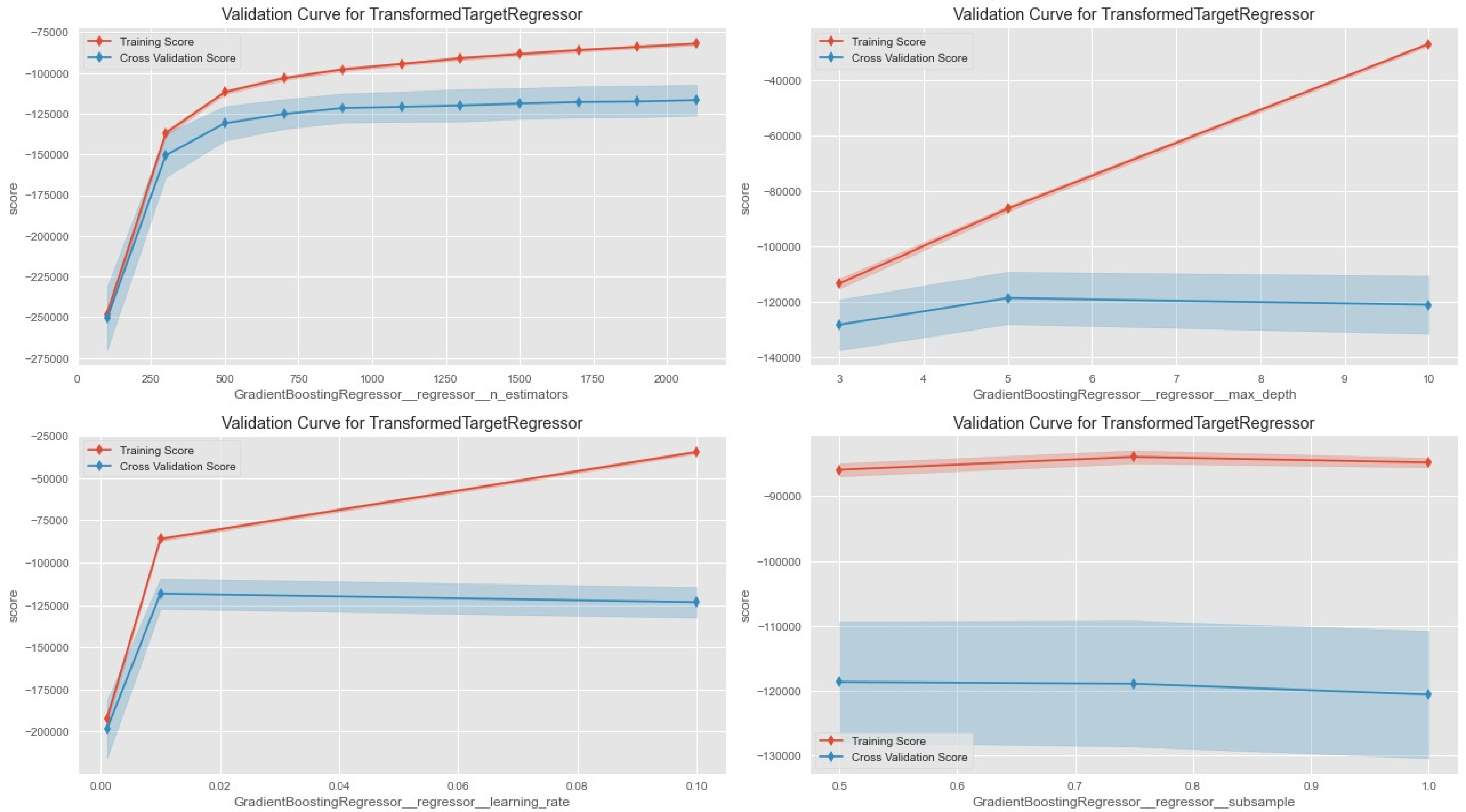
Hyperparameter Tuning – HalvingGridSearchCV

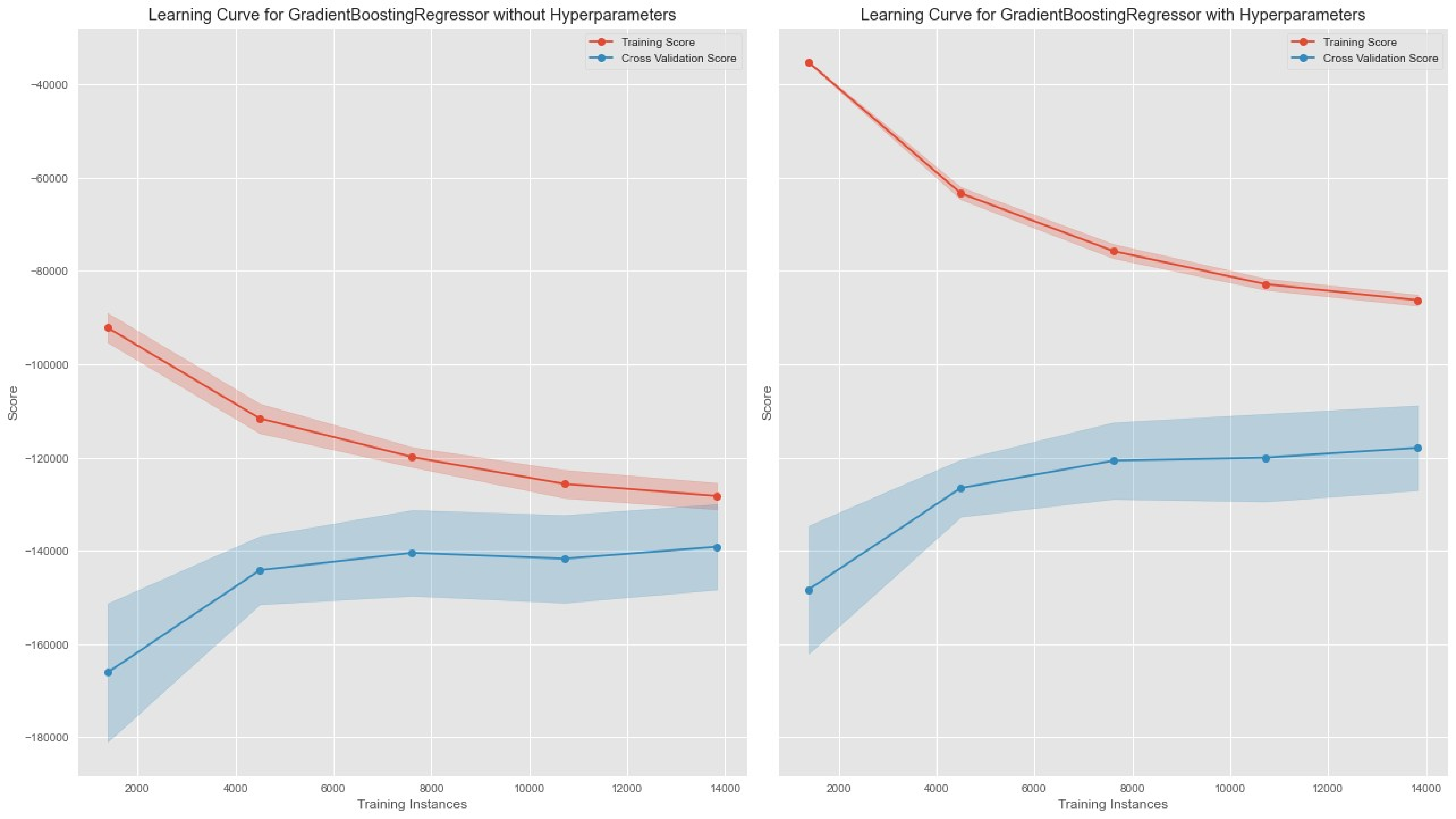
```
grid = {
    'regressor__n_estimators': np.arange(100, 2101, 200),
    'regressor__max_depth': [3, 5, 10],
    'regressor__learning_rate': [.001, .01, .1],
    'regressor__subsample': [.5, .75, 1]
}

model_tuning = Pipeline(
    steps=[
        ('LogTransform', LogTransform),
        ('Normalisation', scaler),
        ('GridSearch', HalvingGridSearchCV(
            TransformedTargetRegressor(
                regressor=GradientBoostingRegressor(),
                func=np.log1p,
                inverse_func=np.expm1
            ),
            grid,
            scoring='neg_root_mean_squared_error',
            n_jobs=4,
            verbose=1,
            cv=5,
            aggressive_elimination=True,
            factor=5
        ))
    ]
)

model_tuning.fit(X_train, y_train)
print('Finish Tuning')
```

Hyperparameter Tuning – HalvingGridSearchCV

```python
print(model_tuning.named_steps['GridSearch'].best_estimator_)
print(model_tuning.named_steps['GridSearch'].best_params_)
print(model_tuning.named_steps['GridSearch'].best_score_)
```

```
TransformedTargetRegressor(func=<ufunc 'log1p'>, inverse_func=<ufunc 'expm1'>,
                           regressor=GradientBoostingRegressor(learning_rate=0.01,
                                                               max_depth=5,
                                                               n_estimators=1700,
                                                               subsample=0.5))
{'regressor__learning_rate': 0.01, 'regressor__max_depth': 5, 'regressor__n_estimators': 1700, 'regressor__subsample': 0.5}
-117994.0405785848
```

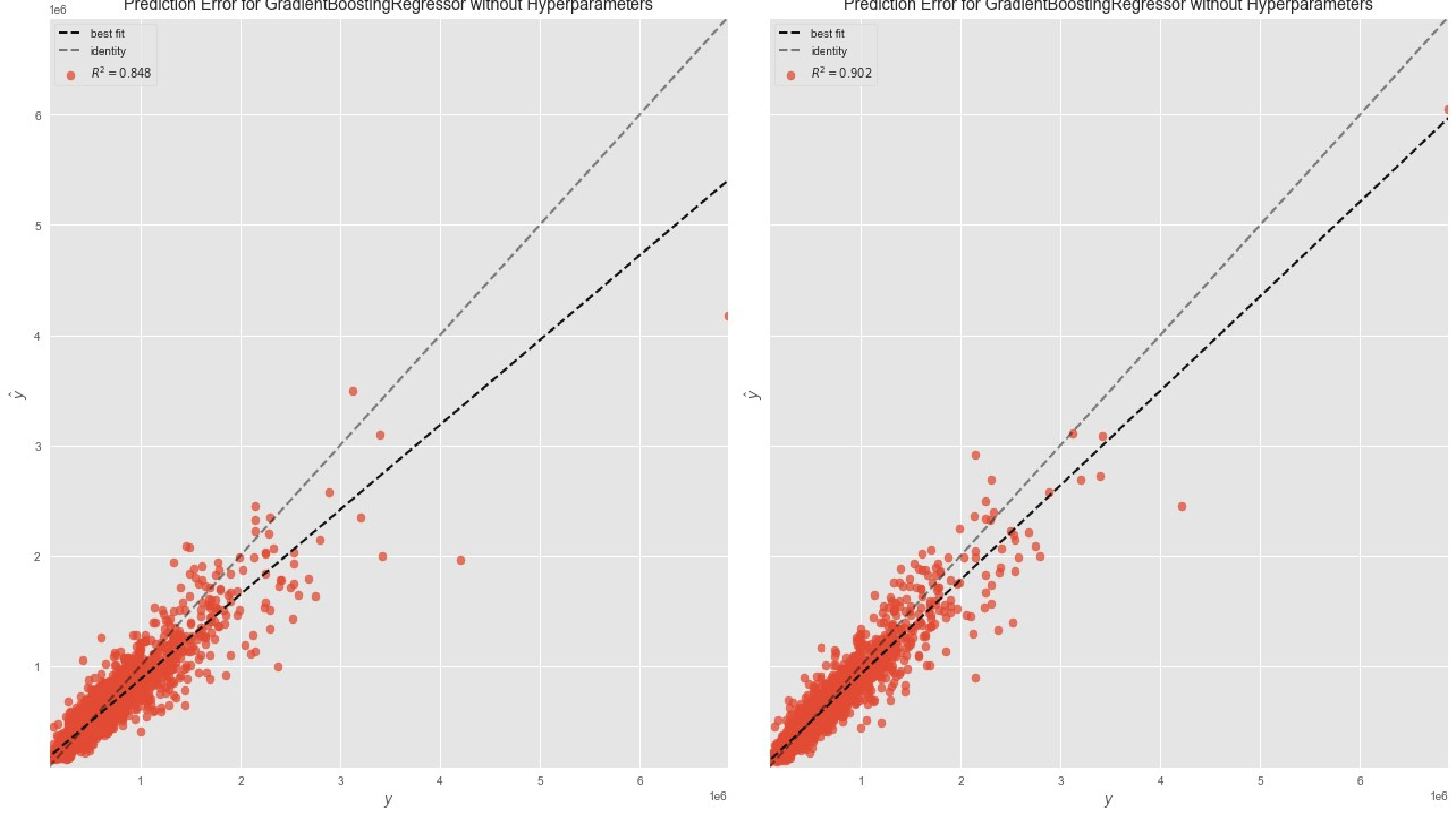

Validation Curve for TransformedTargetRegressor
Training Score
Cross Validation Score
score
GradientBoostingRegressor__regressor__n_estimators
Validation Curve for TransformedTargetRegressor
Training Score
Cross Validation Score
score
GradientBoostingRegressor__regressor__max_depth
Validation Curve for TransformedTargetRegressor
Training Score
Cross Validation Score
score
GradientBoostingRegressor__regressor__learning_rate
Validation Curve for TransformedTargetRegressor
Training Score
Cross Validation Score
score
GradientBoostingRegressor__regressor__subsample

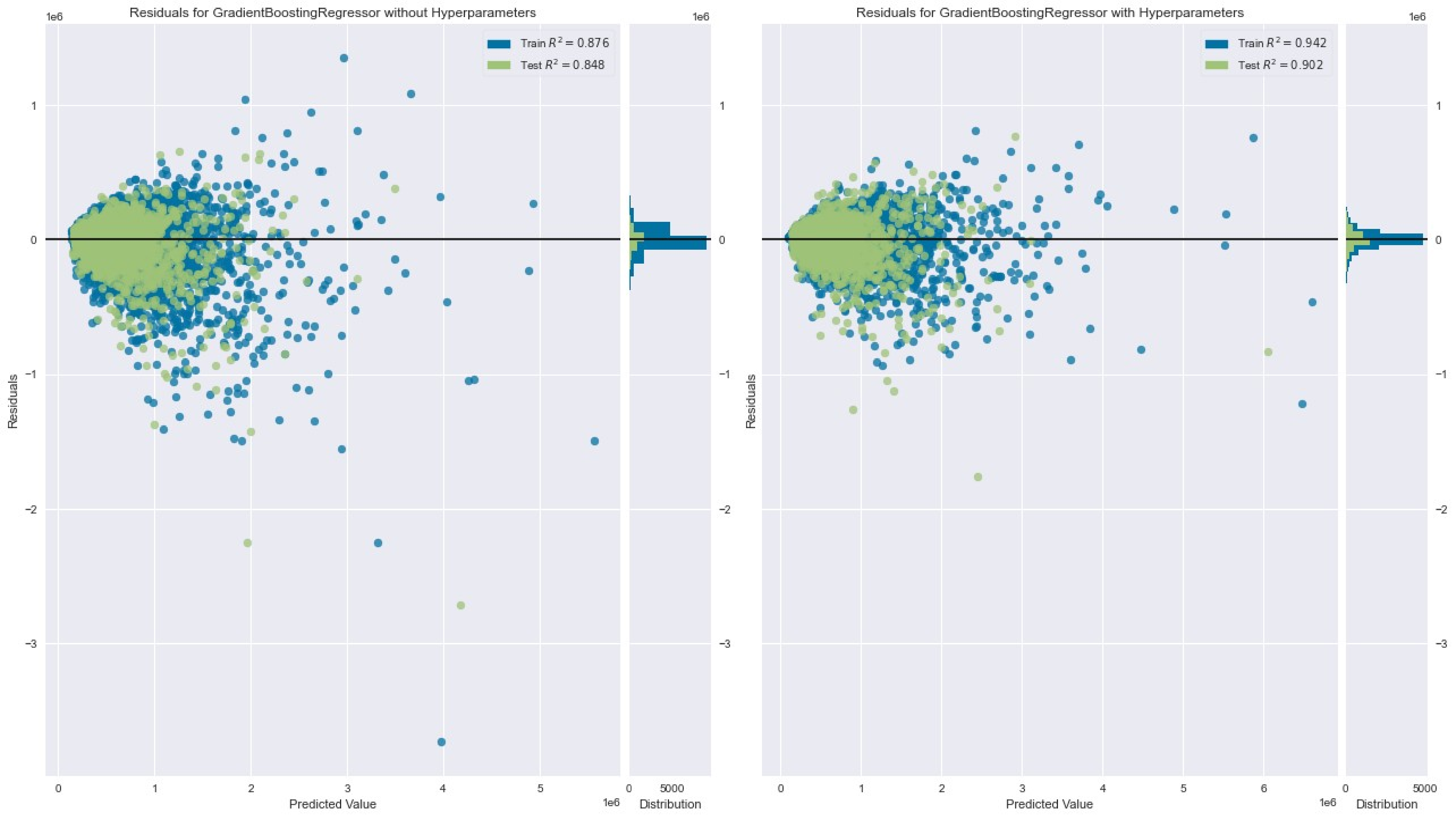


Learning Curve for GradientBoostingRegressor without Hyperparameters
Training Score
Cross Validation Score
-40000
-60000
-80000
-100000
-120000
-140000
-160000
-180000
Score
2000
4000
6000
8000
10000
12000
14000
Training Instances
Learning Curve for GradientBoostingRegressor with Hyperparameters
Training Score
Cross Validation Score
Score
2000
4000
6000
8000
10000
12000
14000
Training Instances

Generating Predictions

	RMSE	MSE	MAE	MAPE	R2
GradientBoostingRegressor	112580.789489	1.267443e+10	64669.690289	0.121199	0.902068


Prediction Error for GradientBoostingRegressor without Hyperparameters
best fit
identity
$R^2 = 0.848$
1e6
$\hat{y}$
y
Prediction Error for GradientBoostingRegressor without Hyperparameters
best fit
identity
$R^2 = 0.902$
1e6
$\hat{y}$
y



Residuals for GradientBoostingRegressor without Hyperparameters
Train $R^2 = 0.876$
Test $R^2 = 0.848$
Residuals
Predicted Value
Distribution
1e6
Residuals for GradientBoostingRegressor with Hyperparameters
Train $R^2 = 0.942$
Test $R^2 = 0.902$
Residuals
Predicted Value
Distribution
1e6

Feature Importance